Analyse des interactions entre flux synchrones et flux asynchrones dans les réseaux temps réel

Hugo Daigmorte

Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Institut Supérieur de l'Aéronautique et de l'Espace

---

Présentée et soutenue par

Hugo DAIGMORTE

Le 21 janvier 2019

Analyse des interactions entre flux synchrones et flux asynchrones dans les réseaux temps réel.

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
ISAE-ONERA MOIS MOdélisation et Ingénierie des Systèmes

Thèse dirigée par
Marc BOYER

Jury

M. Jean-Yves LE BOUDEC, Rapporteur
M. Ye-Qiong SONG, Rapporteur
Mme Anne BOUILLARD, Examinatrice
Mme Sophie QUINTON, Examinatrice
M. Marc BOYER, Directeur de thèse
M. Jean-Luc SCHARBARG, Président

ONERA
THE FRENCH AEROSPACE LAB

Remerciements

Tout d'abord, je souhaite remercier mon directeur de thèse Marc Boyer pour son encadrement, pour sa gentillesse, pour tout le temps qu'il m'a consacré, pour sa disponibilité et pour le soutient qu'il m'a apporté. C'est en grande partie grâce à lui que j'ai pu mener cette thèse à son terme.

Je voudrais aussi remercier mes rapporteurs, Jean-Yves Le Boudec et Ye-Qiong Song, pour leurs analyses attentives et leurs remarques constructives. Je souhaite également remercier les membres du jury, Anne Bouillard, Sophie Quinton et Jean-Luc Scharbarg pour avoir accepté d'évaluer mes travaux de thèse et pour leur intérêt certain concernant cette étude.

Je veux remercier tous les membres du DTIS pour leur aide et leur accueil au cours de ces trois années. Un très grand merci aux doctorants du DTIS pour tous ces bon moments passés ensemble lors des repas et des pauses-café. Merci pour toutes ces discutions autour de films et séries. Cette bonne ambiance a été d'une grande aide lors des moments difficiles.

Je tiens à remercier mes amis qui, même si certains sont à l'autre bout de monde, m'ont toujours soutenu et encouragé. Merci pour tous les bons moments passés ensemble.

Je tiens à remercier très vivement ma famille, merci pour votre soutien et pour vos conseils. En particulier mes parents pour leur éducation et pour m'avoir soutenu tout au long de mes études et aujourd'hui encore, mon frère qui malgré son emploi du temps chargé a pu venir assister à ma soutenance, mon grand père qui m'a donné envie de faire de la recherche et ma grand mère pour son aide indispensable dans la relecture de ce manuscrit. Sans eux ce travail n'aurait pas été possible.

Enfin, je remercie Alice qui me soutient au quotidien et a toujours cru en moi.

Table des matières

Première partie

Introduction

Chapitre 1

Présentation du sujet

1.1 Présentation du contexte

Les systèmes embarqués complexes (avions, satellites, drones...) contiennent de plus en plus de calculateurs. Désormais ce sont des dizaines voire des centaines de calculateurs qui communiquent à travers un réseau partagé. Une fonction est réalisée par la collaboration d'un ensemble de calculateurs qui s'échangent un nombre croissant d'informations. Dans un contexte de temps réel embarqué, il faut non seulement garantir que ces informations échangées sont correctes mais il faut aussi garantir qu'elles vérifient leurs contraintes temporelles. Du point de vue du réseau cela signifie qu'une information doit être échangée en respectant les délais qui lui sont imposés. Ceci implique de pouvoir borner le temps de traversée du réseau de chaque message afin de vérifier qu'il arrive dans les temps. Or les systèmes embarqués étant de plus en plus complexes et le nombre d'informations échangées étant en constante augmentation, cette borne est de plus en plus complexe à calculer. De plus il est important que cette borne soit le moins pessimiste possible afin d'éviter que le système soit surdimensionné.

1.2 Évolution des systèmes avioniques

Dans un premier temps, les échanges entre les calculateurs étaient réalisés à travers une multitude de bus dédiés. Parmi ceux-ci, on peut citer le bus mono-émetteur ARINC 429, dont la première norme date de 1978 et le bus TTP introduit dans les années 80. Le bus ARINC 429 est un bus simple d'utilisation, très fiable et déterministe puisqu'il ne peut pas y avoir de collision. Néanmoins il est limité par son faible débit (débit maximal de 100 kbits/s), le nombre maximum d'utilisateurs (1-20 récepteurs) et l'impossibilité d'avoir plus d'un émetteur qui rend impossible la possibilité de retour entre les calculateurs à moins de mettre en place un second bus.

Pour répondre à ces limitations, Boeing a choisi d'utiliser pour le 777 le bus ARINC 629. Introduit en 1995 il s'agit d'un bus multi-émetteurs avec un débit supérieur (débit maximal de 2 Mbits/s) et autorisant jusqu'à 120 utilisateurs. Plus récemment la norme ARINC 825 cherche à adapter le bus CAN (Controller Area Network), déjà majoritairement utilisé dans l'automobile, pour l'aéronautique.

Le nombre d'informations échangées ne faisant qu'augmenter, l'utilisation d'une mul-

titude de bus devenait de moins en moins satisfaisante d'un point de vue du poids du matériel embarqué et de la maintenance qu'elle impliquait. L'amélioration de bus avionique implique de plus des coûts de développement de composants spécifiques très importants. Une solution a été apportée avec la transition des ces nombreux bus à la technologie de l'Ethernet commuté. On peut citer par exemple le réseau AFDX (technologie utilisée chez Airbus, Boeing et Bombardier) mais aussi TTEthernet, AVB et plus récemment TSN.

Les réseaux embarqués ont donc beaucoup évolué au cours du temps et de nouvelles solutions sont fréquemment proposées. Si l'on s'intéresse aux premiers réseaux utilisés historiquement il est possible de distinguer deux tendances principales :

— Soit les réseaux sont entièrement déterministes et synchrones, les dates d'émission sont connues *a priori* ce qui permet d'éviter les contentions. C'est le cas du bus TTP par exemple.

— Soit les réseaux sont asynchrones, les messages sont envoyés sur le réseau lorsqu'ils sont produits, et ces dates d'émission sont inconnues à la conception. On peut citer par exemple le bus CAN, l'AFDX et AVB.

Actuellement un nouveau type de réseau se développe qui tente de mixer le déterminisme du synchrone et la flexibilité de l'asynchrone en utilisant les deux types de messages. C'est le cas entre autre de TTCAN, TTEthernet et TSN. Ces nouveaux réseaux nécessitent maintenant une modélisation et une méthode d'analyse afin d'être capable de borner le temps de traversée des messages. C'est ce qui a été fait durant cette étude.

1.3 Organisation du mémoire

L'objectif de ce travail est de mettre en place un modèle capable de borner les temps de traversée du réseaux. Afin d'y parvenir nous nous sommes basés sur la méthode d'analyse du Calcul Réseau. Ce travail s'attarde en particulier sur la modélisation des interactions qui existent entre les messages synchrones et les messages asynchrones. Les modèles présentés dans ce manuscrit prennent en compte les dates d'émission sur le réseau des messages synchrones lors du calcul des bornes supérieures de temps de traversée des messages asynchrones.

Notre démarche consiste dans un premier temps à s'intéresser au bus CAN, car simple à modéliser. Dans ce dernier il est possible d'utiliser les dates d'émission à travers un ordonnancement local. Il est aussi possible d'utiliser un ordonnancement global mais cela a un coût. Nous avons tout d'abord proposé une nouvelle solution intermédiaire : une synchronisation faible. Par la suite nous avons développé un modèle capable de borner le temps de traversée lorsqu'un bus faiblement synchronisé est utilisé, ce qui n'avait jamais été fait. Enfin nous avons mené de nombreuses expériences pour évaluer le gain apporté par cette nouvelle approche. Cette partie de l'étude a mené à la publication d'articles [Daigmorte and Boyer, 2016] [Daigmorte et al., 2017] [Daigmorte and Boyer, 2017]. (Chapitre 6)

La démarche a ensuite consisté à considérer le réseau TTEthernet, sachant qu'il est proche du réseau AFDX mais avec en plus de ce dernier l'idée de contrôler les dates d'émission pour les flux prioritaires. L'objectif était double :

— tout d'abord développer une modélisation capable de borner le temps de traversée dans TTEthernet,

— ensuite évaluer finement l'impact des flux synchrones sur le temps de traversée des flux asynchrones.

Cette partie de l'étude a par la suite mené à la publication d'un article [Boyer et al., 2016]. (Chapitre 7)

Enfin l'étude a consisté à considérer le cas du réseau TSN. Ce dernier utilise lui aussi la notion de date d'émission mais y rajoute un concept de crédit d'émission. L'objectif était de proposer une modélisation capable de borner le temps de traversée dans TSN. Pour cela l'idée a été d'analyser le fonctionnement du crédit ainsi que le fonctionnement des messages synchronisés dans TSN. Puis de développer un modèle théorique adapté à TSN, afin de borner le temps de traversée des messages dans TSN. Cette partie de l'étude a mené à l'écriture d'un article [Daigmorte et al., 2018] [soumis pour l'instant]. (Chapitres 8 et 9)

Chapitre 2

Contexte technologique

2.1 Déterminisme, indéterminisme et bornes garanties

Dans le contexte du temps réel embarqué, un nombre très important d'informations est échangé entre les différents calculateurs à travers un réseau embarqué. Il faut non seulement garantir que ces informations échangées sont correctes mais il faut aussi garantir qu'elles vérifient leurs contraintes temporelles. Dans cette étude nous nous sommes concentrés sur la partie "contraintes temporelles".

La traversée du réseau par un message prend un certain temps, on désigne cette durée par le terme de "délai" ou de "latence". Un message respecte sa contrainte temporelle lorsque ce dernier est transmis "dans les temps" c'est-à-dire lorsque son délai est inférieur à une valeur limite la "*deadline*".

Lorsque les dates d'émission et d'arrivée des messages sont connues *a priori* on parle de système "déterministe". Pour un tel système il est donc possible de savoir *a priori* si les contraintes temporelles sont bien respectées.

Au contraire lorsque les dates d'émission et d'arrivée des messages ne sont pas connues *a priori* on parle de système "indéterministe". Il est alors possible que le délai de bout en bout entre des paquets d'un même flux de paquets ne soit pas le même. Cette différence de délai de transmission est désigné par le terme "gigue". Les délais des messages ne sont donc plus connus et pour vérifier que les contraintes temporelles sont bien respectées il est alors nécessaire d'être capable de calculer une borne supérieure du temps de traversée, qu'on désigne par "borne garantie". Si cette borne est inférieure à la "*deadline*", on a alors la garantie que les contraintes temporelles sont bien respectées.

Pour calculer ces bornes supérieures il est non seulement nécessaire d'avoir une méthode de modélisation mais aussi de comprendre le comportement particulier du réseau étudié.

2.2 Les différents types de réseaux

2.2.1 Réseaux en bus ou réseaux commutés

Une topologie de réseau informatique correspond à l'architecture (physique ou logique) de celui-ci, définissant les liaisons entre les équipements du réseau, dans cette étude ces derniers seront en général désignés par "nœuds du réseau". Elle permet de définir la façon dont les équipements sont inter-connectés et la représentation spatiale du réseau. Les

FIGURE 2.1 – Exemple de réseau en bus

topologies peuvent être classées en de très nombreux modes de propagation, nous allons nous concentrer ici sur les deux principaux dans le monde des réseaux embarqués.

Le premier est le *mode de diffusion*. Ce mode de fonctionnement consiste à n'utiliser qu'un seul support de transmission. Les messages sont diffusés à l'ensemble du réseau, chaque nœud reçoit le message et peut ensuite analyser selon l'adresse du destinataire si le message lui est destiné ou non.

Le second est le *mode point à point*. Dans ce mode il y a plusieurs supports physiques, chacun ne reliant qu'une paire de nœuds seulement. Pour que deux stations du réseau communiquent, il y a deux possibilités, soit il existe une liaison directe entre elles, soit il n'en existe pas. Dans ce dernier cas elles doivent obligatoirement passer par un intermédiaire.

Il existe énormément de topologies différentes, les deux que nous présentons par la suite représentent chacune un des deux modes de propagation.

La topologie *Réseau en bus* est représentée par un câblage unique des nœuds du réseau, un exemple est donné Figure 2.1, nous y avons représenté quatre stations (A, B, C et D) connectées à un bus. Elle a l'avantage d'avoir un faible coût de déploiement et la défaillance d'un nœud ne perturbe pas le reste du réseau. Cependant en cas de défaillance du support, c'est l'ensemble du réseau qui ne fonctionne plus. De plus cette solution n'autorise qu'un nombre limité de stations. Enfin le medium étant partagé il est nécessaire de mettre en place un protocole pour l'accès au medium.

La topologie *Réseau commuté* est la topologie la plus courante actuellement, un exemple est donné Figure 2.2. Chaque station est connectée à un nœud particulier : le commutateur (ou *switch*) (noté S sur la Figure 2.2). C'est lui qui recevra tous les messages et les enverra à leur destinataire en fonction de l'adressage. Il est possible de connecter plusieurs commutateurs entre eux. Le message pourra alors traverser un nombre important de liaisons avant d'arriver à son destinataire. Son temps de traversée du réseau est souvent plus complexe à obtenir que dans le cas d'un bus. La panne d'une station ne perturbe pas le fonctionnement global du réseau à moins qu'il ne s'agisse du commutateur, une panne à ce niveau rend une partie du réseau totalement inutilisable.

2.2.2 La synchronisation dans les réseaux

Les réseaux embarqués relèvent de deux paradigmes principaux : leur comportement est dirigé par les événements *Event-Triggered* ou par le temps *Time-Triggered*.

Event-Triggered

Dans le premier cas, *Event-Triggered*, chaque message est transmis suite à un événement déclencheur. Dès que l'événement se produit, le système le prend en compte et transmet le message associé. Ces messages sont asynchrones, ils sont envoyés à une date inconnue. Le protocole doit prévoir un moyen de gérer l'accès au medium afin de gérer les

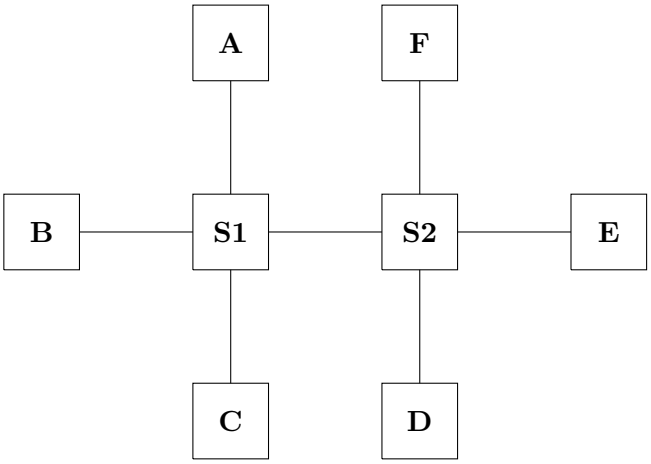


FIGURE 2.2 – Exemple de réseau commuté

contentions et garantir que l'information sera transmise suffisamment rapidement pour ne pas être caduque à sa réception. Cette stratégie est très efficace en matière d'utilisation de bande passante. De plus, elle offre la possibilité de modifier simplement le réseau, que ce soit en ajoutant de nouveaux nœuds et/ou en ajoutant la transmission de nouveaux messages. Néanmoins, pour les applications *temps réel*, il est nécessaire de garantir que tous les messages respectent leurs échéances, le réseau est "indéterminisme", il peut néanmoins offrir des "bornes garanties". Cette vérification n'est pas aisée dans le cas général, différentes méthodes on été développées pour cela qui seront détaillées au Chapitre 3.

Time-Triggered

Dans le second cas, *Time-Triggered*, chaque message est transmis à un instant précis. Ces messages sont synchrones, ils sont envoyés à une date connue, le système est déterministe. Un ordonnancement a été prédéfini *a priori* hors-ligne, chaque message y est associé à un intervalle de temps, appelé créneau (ou *slot*), le réseau transmet chaque message à la date qui correspond. L'allocation du support de communication fonctionne de manière cyclique ainsi l'enchaînement de ces créneaux se répète de façon périodique. Cette stratégie de partage pour l'accès au medium est la méthode *Time Division Multiple Access* (TDMA) [Chan, 2007]. En raison de cet ordonnancement statique des messages, le comportement du réseau est déterministe et donc entièrement prévisible. Il est possible de vérifier très simplement que les contraintes temporelles de chaque message sont bien respectées. Néanmoins, cette méthode a tendance à surcharger le réseau et n'est pas très efficace en matière d'utilisation de bande passante. En effet, un créneau est toujours alloué à un message même dans le cas ou il n'y aurait pas de nouvelles informations à transmettre. De plus, comme l'ordonnancement est prédéfini, toute modification du réseau, que ce soit l'ajout de nouveaux nœuds et/ou de nouveaux messages, nécessite non seulement de recalculer un nouvel ordonnancement mais aussi de configurer tous les nœuds afin qu'ils prennent en compte ces modifications.

La notion de Time-Triggered nécessite l'existence d'horloges et d'une synchronisation à laquelle se référer pour appliquer l'ordonnancement.

Horloge globale On parle de synchronisation globale lorsque chaque nœud du système se réfère à la même horloge. On a alors une synchronisation inter-nœuds, les messages envoyés par chaque élément du réseau sont synchronisés avec tous les autres. Pour cela il est nécessaire de synchroniser les différentes horloges entre elles. Il existe différentes solutions pour un synchroniser des horloges à travers un réseau partagé comme par exemple *Precision Time Protocol* [Lee et al., 2005]. Cette synchronisation peut être réalisée avec des protocoles implémentés au niveau logiciel et nécessitant souvent un support matériel spécifique. Avoir recours à un matériel dédié rajoute un coût supplémentaire important.

Horloges locales On parle de synchronisation locale lorsque chaque nœud du système se réfère à sa propre horloge. On n'est alors plus à proprement parler dans une situation *Time-Triggered* puisque la synchronisation n'est qu'intra-nœuds, les messages envoyés par chaque élément du réseau sont synchronisés avec tous ceux envoyés par le même émetteur. Une synchronisation entre les éléments du réseau n'est donc pas nécessaire. Cependant contrairement au cas d'une horloge globale, des contentions sont possibles entre messages provenant de différents nœuds. Le protocole doit donc prévoir un moyen de gérer l'accès au medium.

De nombreuses études comparatives ont été réalisées sur l'approche *Event-Triggered* et l'approche *Time-Triggered* comme par exemple [Albert, 2004].

Dans la suite de cette section, nous présentons des réseaux obéissant à l'un ou l'autre des paradigmes. Dans les réseaux dirigés par les événements nous détaillerons le bus CAN [Bosch et al., 1991], le réseau Ethernet [Ethernet, 2015], l'AFDX [AFDX, 2005] et AVB [802.1BA, 2011]. Pour ceux dirigés par le temps nous présenterons TTP/C [Kopetz and Grunsteidl, 1993]. Enfin certaines propositions tentent de concilier les avantages des deux approches en minimisant leurs inconvénients respectifs. C'est le cas de FlexRay [FlexRay Consortium, 2005], TTCAN [Leen and Heffernan, 2002], TTEthernet [Kopetz et al., 2005] et TSN [802.1Qbv, 2015].

2.3 Les principaux réseaux en bus

2.3.1 CAN : Controler Area Netwok

Le bus de données CAN [Bosch et al., 1991] (Controller Area Network) est un bus très répandu dans beaucoup d'industries, notamment dans le secteur de l'automobile, mais aussi dans l'aéronautique via des protocoles standardisés comme la norme ARINC 825 [ARINC 825, 2011].

Il existe deux versions du protocole CAN, qui diffèrent par la taille de l'identifiant : CAN 2.0A (le CAN "standard") avec un identifiant de 11 bits et CAN 2.0B (CAN "étendu") avec un identifiant de 29 bits. Les débits du bus CAN se situent entre 125 kb/s et 1 Mb/s. Dans le but d'augmenter la bande passante procurée par le bus CAN, tout en ayant au maximum la plupart de la partie logiciel et de la partie matérielle inchangées, Bosch introduit en 2012 CAN FD [Hartwich et al., 2012] (CAN with Flexible

Data rate). CAN FD modifie le format d'un message CAN en augmentant le nombre maximal d'octets de données par trame CAN. Il passe ainsi de 8 à 64. Cette augmentation est possible car le débit binaire du bus est modifié lors de l'émission de la partie donnée d'un message. Les réseaux CAN FD rendent possibles des utilisations jusqu'à 15Mbits/s.

L'accès au bus se fait de manière asynchrone, l'émission est dirigée par les événements *Event-Triggered*, selon la technique CSMA/CD (*Carrier Sense Multiple Access / Collision Detection*). Chaque message est caractérisé par un identifiant unique. Cet identifiant est utilisé par chaque nœud à la réception pour déterminer si ce message lui est destiné. L'accès au médium se fait en fonction de la priorité des trames. Il n'y a pas de tour de parole prédéterminé entre les différents nœuds, tous les nœuds ne peuvent émettre que si le bus n'est pas actuellement occupé, l'émission est non-préemptive. Enfin si plusieurs nœuds cherchent à émettre simultanément, un arbitrage permet de déterminer lequel est prioritaire : en effet son identifiant sert aussi de priorité, celui qui commence par le plus grand nombre de bits dominants, c'est à dire celui qui a l'identifiant le plus petit, est prioritaire. Les messages de priorité inférieure seront retransmis lorsque le bus sera libre. Ce mécanisme permet de résoudre les conflits d'accès au bus. Cet identifiant est aussi utilisé en réception. Chaque nœud recevant un message regarde l'identifiant pour savoir si ce dernier lui est destiné. Si c'est le cas, il le traite, si au contraire il ne lui est pas destiné, il l'ignore.

Afin de sécuriser la transmission des messages, la méthode du "bit-stuffing" est utilisée. Elle consiste, dans le cas où l'on a émis 5 bits de même polarité d'affilée, d'ajouter à la suite un bit de polarité contraire, pour casser des chaînes trop importantes de bits identiques. En cas de détection d'erreur par un nœud, il émet une trame d'erreur, constituée de 6 bits dominants consécutifs, pour avertir les autres membres du réseau. L'émetteur de la trame erronée doit alors tenter de la ré-envoyer lors de la prochaine étape d'arbitrage.

2.3.2 TTP/C : Time-Triggered Protocol

Le protocole TTP/C [Kopetz and Grunsteidl, 1993] est strictement guidé par le temps, *Time-Triggered*, et non pas par les événements.

Il repose sur la technique du TDMA [Chan, 2007] : le temps est partagé en intervalles entre les différents nœuds (abonnés) du réseau. C'est pendant les intervalles de temps qui lui sont alloués et uniquement pendant ces intervalles que chacun de ces abonnés peut transmettre un message.

Les nœuds ont accès au bus pendant une fenêtre d'émission conformément à un ordonnancement statique effectué *a priori*. Chaque fenêtre est fixe mais les différentes fenêtres ont des tailles différentes, chaque nœud n'a pas le même temps de parole. Un TDMA round est une suite de fenêtres de longueur prédéfinie, où chaque nœud se voit réserver une et une seule fenêtre. Les TDMA round se succèdent et les seules différences sont les données transmises à l'intérieur de chaque message dans une fenêtre donnée.

Toutes les transmissions et réceptions sont commandées par une base de temps globale qui est établie de façon autonome en utilisant un algorithme de synchronisation. Ce protocole est au coeur de TTP/C mais ne sera pas détaillé ici.

2.3.3 Flexray

Historiquement, le protocole FlexRay [FlexRay Consortium, 2005] est né d'un consortium entre BMW, Bosch, Daimler, Chrysler, GM, Motorola, Philips et Volkswagen. Il a été développé pour les besoins spécifiques de l'automobile.

Il concilie des communications *Time-Triggered* et *Event-Triggered*. L'accès au bus est partagé, périodiquement, entre deux segments : le segment statique pour les flux périodiques (*Time-Triggered*) utilisant la technique TDMA [Chan, 2007] et le segment dynamique pour les flux apériodiques (*Event-Triggered*) où l'accès se fait suivant le Flexible TDMA.

Chaque cycle de communication est donc divisé en deux fenêtres. La fenêtre statique est presque équivalente au TDMA round du protocole TTP/C, sauf que les créneaux ont tous la même taille et qu'un même nœud peut obtenir plusieurs créneaux par fenêtre statique.

La fenêtre dynamique permet de transmettre les messages des flux apériodiques. Cependant, pour garantir le déterminisme de la partie statique, la fenêtre dynamique a une longueur prédéfinie, elle peut donc se terminer même si tous les messages apériodiques ne sont pas transmis. Ils seront alors transmis lors de la fenêtre dynamique suivante voire jamais si le bus est trop chargé.

2.3.4 TTCAN : Time-Triggered Controler Area Netwok

Le protocole TTCAN [Leen and Heffernan, 2002] a été mis en place sur la couche physique du bus CAN tout en souhaitant le rendre capable de concilier des communications *Time-Triggered* et *Event-Triggered*.

Pour utiliser un ordonnancement global il faut être capable d'avoir une horloge globale/commune. Dans TTCAN tous les nœuds possèdent une copie de cette horloge sous forme d'un compteur. Pour établir et maintenir cette synchronisation entre les différents nœuds un message de référence est diffusé à tous les nœuds du système de façon périodique. Cela crée un intervalle de temps utilisé par chaque nœud pour réajuster son horloge locale par rapport à l'horloge commune. TTCAN propose deux niveaux de synchronisation : "Level 1" et "Level 2". Le "Level 1" permet une synchronisation d'une précision de l'ordre de la milliseconde et peut être implémenté sur des nœuds CAN classiques. Le "Level 2" permet quant à lui une synchronisation d'une précision de l'ordre de la microseconde cependant il nécessite un matériel dédié.

L'accès au medium dans TTCAN repose sur un mix entre TDMA [Chan, 2007] et CSMA. Un certain nombre de fenêtres est alloué pour les messages *Time-Triggered*, l'accès est alors réservé à ces messages. Le reste du temps l'accès au medium se fait de façon similaire à celui du bus CAN classique.

Des études comparatives entre ces différents types de bus ont été menées par le passé, l'une d'entre elle se trouve dans [Talbot and Ren, 2009].

2.4 Les réseaux commutés basés sur Ethernet

2.4.1 Ethernet commuté

La norme IEEE 802.3 [Ethernet, 2015] définit ce qu'on appelle communément un réseau Ethernet. Dans un réseau Ethernet, l'accès au médium est assuré historiquement par le protocole CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*). CSMA/CD est le moyen par lequel deux (ou plus) nœuds partagent le même support physique. Pour transmettre, une station attend une période d'inactivité sur medium physique, puis transmet ses données. Elle émet pendant un certain temps, si le message entre en collision avec celui d'un autre nœud, les deux machines interrompent leur communication. Ensuite, les deux nœuds attendent un délai aléatoire avant de réessayer de transmettre leur message. Cet algorithme implique un indéterminisme du temps d'accès au medium qui en fait le principal inconvénient du réseau Ethernet CSMA/CD.

Pour diminuer ce risque de collisions, on adopte fréquemment une approche Ethernet commuté point à point. Pour cela on impose des commutateurs dont les ports sont reliés à un et un seul équipement. L'interconnexion avec les autres nœuds est ensuite réalisée en reliant les commutateurs entre eux. Les collisions ne sont donc possibles qu'entre un nœud et son commutateur, ou bien entre deux commutateurs.

Afin d'éviter ces collisions une solution consiste à utiliser des liens bidirectionnels. On appelle couramment ce type de réseau un réseau *Ethernet commuté Full Duplex*. Le réseau Ethernet commuté Full Duplex ne nécessite pas l'utilisation du protocole CSMA/CD et il n'y a donc plus d'indéterminisme quant au temps d'accès au medium. L'indéterminisme est reporté au temps d'attente dans les files des commutateurs.

2.4.2 AFDX : Avionics Full DupleX Switched Ethernet

Le réseau AFDX [AFDX, 2005] (*Avionics Full DupleX switched Ethernet*) est un réseau *Ethernet commuté Full Duplex* qui a été développé pour équiper entre autre l'Airbus A380. Ce type de technologie est donc un réseau Ethernet auquel un certain nombre de modifications ont été appliquées afin de créer une nouvelle norme pour l'avionique. Les débits à ce jour vont jusqu'à 100 Mb/s mais devraient atteindre 1 Gb/s.

La technologie retenue pour l'AFDX repose sur les protocoles *Ethernet commuté Full Duplex*, IP et UDP, et sur l'utilisation de contrats de trafic afin d'assurer une qualité de service aux flux du réseau, appelés liens virtuels ou VL (*virtual link*). Chaque lien virtuel se voit attribuer deux paramètres :

— La taille maximale d'une trame pouvant être transmise sur le VL (**MFS**, *Maximum Frame Size*).

— L'intervalle minimal entre les débuts d'émission de deux trames sur le même VL (**BAG**, *Bandwidth Allocation Gap*).

Les VL sont des canaux de communication qui relient une source émettrice à un ou plusieurs récepteurs. Le rôle d'un nœud est de réguler le trafic des VL en fonction de leurs contrats de trafic. De cette manière, même si une application «s'emballe» et se met à envoyer ses données trop rapidement, le nœud régule son trafic afin de ne pas saturer le reste du réseau.

Les réseaux AFDX apportent une solution partielle aux différents problèmes rencontrés par les industriels de l'aéronautique dans les années 2000 :

— Ils permettent d'alléger les avions et par là même leur consommation de carburant.

— Ils permettent l'échange rapide d'une grande quantité de données.

Malgré ces avantages, les réseaux AFDX nécessitent parfois une solution de secours pour satisfaire les exigences fixées par la criticité des fonctions qui en dépendent. De plus, malgré l'utilisation de câbles full-duplex et, de fait, la suppression des collisions de trames, les temps de traversée de bout en bout sont inconnus car les délais subis par les trames d'un équipement à un autre en traversant le réseau sont indéterministes (au sens de notre définition de la section 2.1). Cet indéterminisme s'entend au sens où ils ne sont pas constants ; en effet, chaque trame d'un VL peut traverser le réseau en plus ou moins de temps, en fonction du trafic. Cela dit, le mécanisme de contraintes de débit des VL en fait des systèmes à "bornes garanties" (section 2.1), il est tout de même possible de borner ces délais et donc d'utiliser ces réseaux dans un environnement embarqué critique. Il existe différentes méthodes, principalement le calcul réseau, qui seront détaillées au Chapitre 3.

2.4.3 TTEthernet : Time-Triggered Ethernet

TTEthernet [Kopetz et al., 2005] vise à étendre le protocole AFDX en conservant un fonctionnement *Event-Triggered* mais en y rajoutant la possibilité d'utiliser une communication *Time-Triggered*.

TTEthernet est une technologie de réseau embarqué, compatible avec IEEE 802.3. Les nœuds échangent des trames au travers de *virtual links* (sauf pour un trafic *best-effort*) déjà définis dans le paragraphe sur l'AFDX. Pour rappel, un *virtual link* définit une connexion unidirectionnelle d'un nœud émetteur à un ou plusieurs nœuds récepteurs : communication *Uni- ou Multicast*. TTEthernet offre plusieurs classes de trafic en parallèle : *Time-Triggered*, *Rate-Constrained*, et *Best-Effort*.

— *Time-Triggered* **TT** : les messages sont envoyés en fonction du temps en suivant un ordonnancement prédéfini sur chaque lien. C'est le trafic le plus prioritaire.

— *Rate-Constrained* **RC** : les messages sont envoyés en fonction d'un événement, mais respectent la durée minimale entre deux trames du même flux à l'entrée du réseau. Ce trafic est moins prioritaire que le trafic TT.

— *Best-Effort* **BE** : les messages correspondent au trafic standard d'une communication Ethernet. Il n'y a aucune garantie sur la transmission des messages. Le trafic *best-effort* utilise la bande passante restante du réseau et a une priorité inférieure aux deux autres types de trafic.

Pour le trafic avec la priorité maximale TTEthernet offre la communication *Time-Triggered* en utilisant la technique TDMA [Chan, 2007]. Dans le cas idéal, les horloges locales des éléments du réseau sont parfaitement synchronisées, les trames sont envoyées et relayées selon un ordonnancement défini *a priori*. Les horloges locales de tous les participants étant synchronisées, si l'ordonnancement a été bien créé, les contentions dans le réseau sont évitées : la possibilité que deux participants ou plus émettent en même temps peut être exclue à la conception. C'est pour cela que les communications *Time Triggered* fournissent d'excellentes garanties temporelles une faible latence et une gigue quasi nulle.

Mais cette communication exige une gestion constante de la synchronisation des horloges locales. En effet cette synchronisation n'est pas parfaite et la maintenir nécessite l'émission de trames prioritaires qui perturbent le reste du trafic.

Pour un trafic avec des exigences de déterminisme moindres, TTEthernet offre le trafic *Rate-Constrained*. Le trafic *Rate-Constrained* est communément connu comme le trafic fourni par AFDX. *Rate-Constrained* est le paradigme de communication réalisé par ARINC 664 part 7. Chaque expéditeur réalise une fonction de mise en forme de trafic pour chaque flux de données *Rate-Constrained*. Cette contrainte du trafic assure qu'il y aura un écart minimum entre l'émission de deux trames successives pour un même flux de données. Cela garantie une limitation de l'utilisation de la bande passante, cependant cela n'assure pas de préserver l'ordre des transmissions et ne donne pas de garanties temporelles.

2.4.4 AVB : Audio Video Bridging

Le réseau AVB (*Audio Video Bridging*) est un réseau Ethernet full duplex conçu pour faciliter l'échange des flux audio et vidéo dans un contexte d'applications temps réel. L'émission des messages dans AVB est dirigée par les événements *Event-Triggered*. AVB est décrit dans plusieurs sous-normes de la norme IEEE 802.1.

La norme IEEE 802.1 BA [802.1BA, 2011] précise les caractéristiques que doivent respecter les émetteurs, récepteurs et commutateurs. Elle explique les différentes configurations possibles.

La norme IEEE 802.1 AS [802.1QAS, 2011] définit le protocole de synchronisation qui doit être utilisé dans AVB, ce dernier est surtout utilisé pour TSN que l'on présente par la suite.

La norme IEEE 802.1 Qat [802.1Qat, 2010] présente le *Stream Reservation Protocol*. Elle définit les protocoles nécessaires à la réservation des ressources pour les flux de données. Ce protocole permet de dynamiquement allouer des ressources à des flux au niveau des commutateurs. À un flux est associé une taille maximale des messages et le nombre maximum de messages qu'un émetteur peut envoyer durant un *class measurement interval*. Cet intervalle permet de contrôler la vitesse d'émission et éviter les problèmes de congestion.

Enfin la norme IEEE 802.1 Qav [802.1Qav, 2010] offre plusieurs types de trafic en parallèle : un trafic temps réel et un trafic best-effort. AVB autorise au plus 8 niveaux de priorité, chacun étant associé une classe. Les deux classes les plus prioritaires utilisent un système de crédit. L'algorithme gérant les variations de ce crédit est le *Credit-Based Shaping* (CBS). Le système de crédit permet d'éviter que ces classes prioritaires n'utilisent toute la bande passante disponible. À chacune de ces classes est associé un crédit. Ce dernier est incrémenté lorsque un message de cette classe est en train d'être émis et décrémenté lorsque c'est le message d'une autre classe qui est émis. Un message ne peut être envoyé que lorsque le crédit de sa classe est positif. Une présentation plus détaillée sera donnée au Chapitre 9.

2.4.5 TSN : Time Sensitive Network

TSN (*Time Sensitive Network*) désigne un groupe de travail de l'IEEE, qui complète Ethernet, et les files d'attente Ethernet (802.1Q), afin de lui permettre de supporter des

flux de données en temps réel. Le mot ≪TSN≫ est aussi le nom utilisé pour un réseau implémentant ces nouvelles fonctionnalités. Ici nous nous concentrerons sur la partie de la norme TSN liée à l'ajout de communication *Time-Triggered*.

Ce groupe est actuellement toujours en cours de travaux, la norme TSN n'est donc pas entièrement définie. De plus, l'architecture TSN admet beaucoup de paramètres, conduisant à de nombreux comportements différents. Dans la situation TSN la plus générique, chaque port de sortie possède 8 files d'attente.

Le réseau TSN que nous considérons est une extension d'AVB ayant pour but de conserver un fonctionnement *Event-Triggered* et le système de crédit mais d'y rajouter la possibilité d'utiliser une communication *Time-Triggered*.

La norme IEEE 802.1 Qbv [802.1Qbu, 2016] précise le fonctionnement de l'aspect *Time-Triggered* dans TSN. Ce dernier est géré à l'aide d'un système de portes. Chaque classe ne peut émettre que durant les fenêtre qui lui sont allouées, lorsque la porte d'une classe est fermée, elle est "gelée", elle ne peut plus émettre.

Ce dernier est le plus prioritaire des trois. Il respecte un ordonnancement défini *a priori*, un certains nombre de fenêtres est alloué pour ce trafic. Durant ces fenêtres les autres trafics sont gelés, ils ne peuvent plus émettre.

Plusieurs politiques d'intégration sont disponibles, elles sont détaillées dans la norme IEEE 802.1 Qbv [802.1Qbu, 2016] qui explique comment mettre en place une préemption des flux moins prioritaires en cas de conflit pour l'accès au medium.

Une explication beaucoup plus précise du fonctionnement de TSN sera donnée dans les Chapitres 8 et 9.

Chapitre 3

État de l'art des différentes méthodes d'analyse

Nous avons vu dans les chapitres précédents que l'évolution de la complexité des systèmes embarqués a conduit à l'utilisation de nouveaux moyens de communication. Cependant, dans un contexte temps réel il faut non seulement garantir que les informations échangées sont correctes mais il faut aussi garantir qu'elles vérifient leurs contraintes temporelles. Or ces nouveaux moyens de communication ne ne sont pas accompagnés de méthodes de calcul de délais permettant d'assurer que le réseau offrira bien la qualité de service requise, qui comprend en particulier une latence bornée.

Dans ce chapitre nous présentons plusieurs approches pour l'étude des délais maximaux de bout en bout. Il ne s'agit pas d'un panorama exhaustif, mais d'une brève présentation des approches que nous avons envisagées.

3.1 L'expérimentation et la simulation

Commençons par évoquer la possibilité d'obtenir les délais maximaux à travers une expérimentation (ou une simulation). Si cette approche semble tentante, car simple à mettre en place, elle ne peut bien souvent pas être appliquée. En effet, excepté dans le cas de réseaux entièrement déterministes où le temps exact de traversée est garanti, la situation de "pire cas" c'est-à-dire celle qui conduira au temps de traversée maximal est non seulement inconnue mais peut de plus avoir une probabilité extrêmement faible de se produire. Ainsi même si au cours de la simulation toutes les contraintes temporelles sont respectées cela ne garantit pas qu'elles le seront toujours. Cette méthode ne peut donc pas être utilisée pour obtenir une borne déterministe du temps de traversée.

Il est cependant possible d'avoir une approche probabiliste comme cela a été proposé dans l'article [Scharbarg et al., 2009]. On obtient alors une borne supérieure qui peut être dépassée avec une certaine probabilité p.

Cependant, même si cette approche peut être intéressante, nous cherchons ici une méthode pour le calcul d'une borne supérieure garantie du délai de bout en bout sur un réseau et il n'est donc pas possible d'utiliser l'expérimentation pour cela.

3.2 Le modèle checking

Le *model checking* désigne une technique de vérification de propriétés sur des systèmes. Cette approche est basée sur une modélisation du système sous forme d'automates et fut introduite par R. Alur et D.L. Dill en 1994 [Alur and Dill, 1994] pour décrire le comportement de systèmes en incluant le temps.

Le *model checking* consiste à exprimer le système considéré au moyen d'un graphe orienté, formé de nœuds et de transitions. Chaque nœud représente un état du système, chaque transition représente une évolution possible du système. À partir d'un état initial de l'automate, celui–ci évolue à l'aide de transitions d'un état donné vers un autre état.

À l'aide du *model checking*, il est possible de vérifier qu'une propriété est vraie quel que soit l'état du système en la vérifiant pour tous les états de son automate.

Cette approche par *model checking* a déjà été explorée sur plusieurs architectures réseau en particulier le bus CAN [Krákora et al., 2004] et Ethernet [Witsch et al., 2006] voire les deux [Ermont et al., 2006].

Ces méthodes à base d'exploration exhaustive des états d'un système sont capables de fournir des résultats exacts et pas seulement des bornes supérieures sur le temps de traversée. Dans la pratique cependant, l'application de l'approche du *model checking* n'est pas possible en général puisqu'elle conduit à une explosion du nombre d'états. Plus généralement il a été montré que le calcul exact du temps de traversée des messages est, en général, incalculable en un temps raisonnable puisque son calcul est NP-difficile [Bouillard and Thierry, 2016].

3.3 La méthode des trajectoires et Forward end-to-end Analysis

La méthodes des trajectoires introduite par S. Martin en 2004 [Martin, 2004] est une méthode analytique qui permet de calculer une borne supérieure du délai de bout en bout d'un paquet en identifiant les paquets d'autres flux que ce paquet rencontre sur sa trajectoire lors de la traversée du réseau. Concrètement, pour déterminer le pire temps de traversée du réseau d'un flux, l'idée est d'identifier l'ensemble des paquets pouvant interférer avec un paquet quelconque de ce flux. Cette information permet de déterminer alors la "pire" date de démarrage sur le dernier noeud traversé. Et ensuite d'en déduire le pire temps de traversée pour ce message.

Si la méthode a été considérablement améliorée depuis sa version d'origine avec entre autre des applications à l'AFDX dans [Bauer et al., 2009] et [Bauer et al., 2010]. Il a par la suite été montré que cette première version pouvait parfois s'avérer optimiste [Kemayo et al., 2013]. La borne supérieure calculée à l'aide de cette méthode pouvait parfois être inférieure au pire cas observé. Une correction a été apportée dans [Li et al., 2014] près de dix ans après les travaux d'origine.

La *Forward end-to-end Analysis* est une méthode d'analyse de bout en bout qui permet de calculer une borne supérieure du délai de bout en bout d'un paquet. Cette méthode est récente puisqu'elle a seulement été introduite en 2004 [Kemayo et al., 2012].

Cette approche analyse de manière itérative les nœuds traversés par un flux depuis son nœud source jusqu'à sa destination. Pour chaque nœud traversé, l'approche *Forward end-to-end Analysis* détermine le délai le plus défavorable que peut subir un paquet de ce flux. Pour cela ce délai est décomposé en deux termes :

— le temps maximum nécessaire pour le paquet avant d'arriver au nœud considéré,

— le temps nécessaire maximum pour émettre tous les paquets actuellement en attente dans le nœud à l'arrivée du paquet.

Dans sa première version cette méthode d'analyse ne gérait pas la sérialisation des flux, l'approche a depuis été complétée afin de la prendre en compte [Kemayo et al., 2015].

Que ce soit la méthode des trajectoires ou la *Forward end-to-end Analysis*, elles ne modélisaient que peu de politique de service (FIFO, SP) au moment où cette thèse a débuté. À l'inverse le Calcul Réseau (présenté en détail dans le Chapitre 4) était beaucoup plus riche et nous semblait donc être une méthode plus appropriée.

3.4 Event Stream

L'idée de base de cette méthode d'analyse consiste à compter des évènements et non pas une quantité de données échangées. L'*Event Stream* repose sur l'article fondateur [Gresser, 1993] qui explique comment définir le nombre maximum d'évènements sur un intervalle de temps. L'idée est alors de reprendre sur chaque nœud une analyse classique d'ordonnancement. Puis de propager le temps de réponse comme une jigue.

En utilisant ce concept il a été montré comment cette notion d'évènement permettait de borner le temps de traversée des messages [Richter and Ernst, 2002][Henia et al., 2005].

Le but de cette méthode était de proposer un modèle simple et proche des modèles d'ordonnancement classiques afin de simplifier son utilisation. Cependant en cherchant à améliorer la méthode, cette dernière a fini par se complexifier grandement.

Il a de plus été montré [Boyer and Roux, 2016] que l'*Event Stream* et le Calcul Réseau avait de nombreux liens et qu'il était possible au moins en partie d'unifier ces deux méthodes d'analyse.

3.5 Le Calcul Réseau

Le Calcul Réseau est une théorie mathématique proposée à l'origine par R.L. Cruz dès 1991 [Cruz, 1991a][Cruz, 1991b]. Cette théorie fut par la suite complétée et formalisée à l'aide de l'algèbre *(min, +)* par C.-S. Chang dans son livre [Chang, 2000] et par J.-Y. Le Boudec et P. Thiran dans [Le Boudec and Thiran, 2001]. Le Calcul Réseau a deux objectifs, premièrement borner les délais de traversée de bout en bout pour les flux et secondement calculer des bornes supérieures sur les tailles des files d'attente.

Le Calcul Réseau utilise les notions de "courbe d'arrivée" pour modéliser un contrat de trafic et de "courbe de service" pour modéliser le serveur.

Le Calcul Réseau est la méthode la plus utilisée actuellement, en particulier c'est cette méthode qui a été utilisée afin de certifier une configuration industrielle du réseau AFDX [Grieu, 2004].

C'est cette méthode qui nous a semblé la plus appropriée pour notre étude puisque elle offre un cadre formel de modélisation des réseaux de communication et a déjà prouvé son efficacité dans le passé. Dans le Chapitre 4 nous présentons avec beaucoup plus de détails cette théorie.

3.6 Le Real Time Calculus

Le *Real Time Calculus* est très souvent présenté comme une alternative au Calcul Réseau. Le *Real Time Calculus* se base globalement sur les même concepts que le Calcul Réseau classique en utilisant les notions d'enveloppes pour modéliser le trafic entrant et de service pour modéliser le serveur mais comporte un certains nombre de spécificités qui lui permettraient d'obtenir une meilleure modélisation du réseau. Une comparaison entre les deux approches a été faite [Bouillard et al., 2009].

Cette comparaison a mis en évidence un certain nombre de différences. Tout d'abord l'espace des fonctions étudiées n'est pas le même ($\mathbb{R}^+$ pour le Calcul Réseau et $\mathbb{R}$ pour le *Real Time Calculus*). De plus les fonctions manipulées en *Real Time Calculus* possèdent deux variables alors qu'elles n'en possèdent qu'une en Calcul Réseau.

Malgré ces différence il a été montré [Bouillard et al., 2009] que les deux méthodes étaient formellement équivalentes.

Chapitre 4

Calcul Réseau

4.1 Introduction

Le Calcul Réseau (ou *Network Calculus*) est une théorie qui a été développée afin d'analyser les comportements critiques dans les réseaux de communication. Le Calcul Réseau s'intéresse pour cela aux performances pire-cas. L'objectif du Calcul Réseau est de calculer des bornes garanties (section 2.1) sur les délais subis par des flux dans des réseaux (temps de trajet de bout en bout, *delays*), et sur la charge utilisée par ces flux dans les éléments du réseau (taille maximale des files d'attente au niveau d'un nœud du réseau, *backlogs*).

R.L. Cruz a écrit les deux articles fondateurs de cette théorie : le premier considère les éléments de réseaux pris isolément [Cruz, 1991a] le second s'intéresse à la mise en réseau de ces différents éléments [Cruz, 1991b].

Par la suite, cette théorie a été complétée et formalisée, jusqu'à la parution dans les années 2000 de deux ouvrages devenus des références dans le domaine : le livre de C.-S. Chang [Chang, 2000] et celui de J.-Y. Le Boudec et P. Thiran [Le Boudec and Thiran, 2001].

La théorie du calcul réseau est basée sur l'algèbre *(min, +)*. Les informations sur le système sont modélisées grâce à des fonctions, telles que les courbes d'arrivée qui modélisent un contrat de trafic et les courbes de service qui quantifient le service garanti en chaque nœud du réseau.

4.2 Modèle mathématique

4.2.1 Dioïde (min-plus)

Un des intérêts du Calcul Réseau est que les fonctions qu'il manipule s'expriment très bien mathématiquement dans le dioïde [Gondran and Minoux, 2001] dont la première loi est le minimum $\wedge$, et la seconde loi la somme $+$.

Définition 1 (Monoïde)**.** *Un Monoïde est un ensemble E muni d'une loi $\oplus$ et d'un élément neutre e, tel que :*

— $\forall\ (a,\ b) \in E^2,\ a \oplus b \in E$ *(stabilité).*

— $\forall\ (a,\ b,\ c) \in E^3,\ a \oplus (b \oplus c) = (a \oplus b) \oplus c$ *(associativité).*

— $\forall\ a \in E,\ a \oplus e = e \oplus a = a$ *(existence d'un élément neutre).*

On dit qu'un monoïde est commutatif si : $\forall$ (a, b) $\in E^2$, a $\oplus$ b = b $\oplus$ a.

Définition 2 (Semi-anneau)**.** *Un semi-anneau est un ensemble E muni de deux lois, la somme $\oplus$ et le produit $\otimes$, et qui contient deux éléments distincts notés e et ε, tel que :*

— $(E, \oplus, e)$ *est un monoïde commutatif.*

— $(E, \otimes, \varepsilon)$ *est un monoïde.*

— $\otimes$ *est distributif par rapport à* $\oplus$ *:* $\forall$ *(a, b, c, d)* $\in E^4$,
(a $\oplus$ *b)* $\otimes$ *(c* $\oplus$ *d) = (a* $\otimes$ *c)* $\oplus$ *(b* $\otimes$ *c)* $\oplus$ *(a* $\otimes$ *d)* $\oplus$ *(b* $\otimes$ *d).*

— *e est un élément absorbant pour* $\otimes$ *:* $\forall\ a \in E,\ a \otimes e = e$.

Un semi-anneau a toutes les propriétés de structure des anneaux, sauf qu'il n'est pas requis que $\oplus$ soit une loi de groupe (dans un anneau, tout élément admet un élément symétrique pour $\oplus$, souvent appelé opposé).

Définition 3 (Dioïde)**.** *Un dioïde est un semi-anneau idempotent, c'est-à-dire que la loi* $\oplus$ *vérifie :*

— $\forall\ a \in E,\ a \oplus a = a$.

On remarquera qu'un anneau ne peut pas être un dioïde. Supposons l'existence d'un dioïde dont chaque élément a admet un opposé $\overline{a}$. On a alors pour tout $a \in E$, en utilisant l'associativité de $\oplus$, l'égalité des deux quantités suivantes :

— $(a \oplus a) \oplus \overline{a} = a \oplus \overline{a} = e$

— $a \oplus (a \oplus \overline{a}) = a \oplus e = a$

Le dioïde devrait donc être réduit au seul élément e, alors qu'il doit comporter au moins deux éléments distincts : e et ε.

Théorème 1 (Le Dioïde MIN-PLUS)**.** *Notons* $\mathbb{R}_{min} = \mathbb{R} \cup \{+\infty\}$*, et* $\wedge$ *l'opérateur minimum. Alors* $(\mathbb{R}_{min}, \wedge, +)$ *est un dioïde commutatif avec* $e = +\infty$ *et* $\varepsilon = 0$.

Démonstration. $\forall$ (a, b, c, d) $\in \mathbb{R}^4_{min}$,

— Le minimum est bien une loi commutative, associative et idempotente.

— $\forall$ a $\in \mathbb{R}_{min}$, a $\wedge$ $+\infty$ = a (élément neutre de $\wedge$).

— L'addition est bien une loi commutative et associative.

— a + 0 = a (élément neutre de +).

— (a $\wedge$ b) + (c $\wedge$ d) = (a + c) $\wedge$ (b + c) $\wedge$ (a + d) $\wedge$ (b + d).

— a + ($+\infty$) = $+\infty$ ($+\infty$ élément absorbant pour +).

□

4.2.2 Opérateurs

Maintenant que l'on a rappelé la définition du dioïde *(min, +)* nous allons présenter les principaux opérateurs du Calcul Réseau.

Définition 4 (Ensembles $\mathcal{F}$ et $\mathcal{F}_0$)**.** *$\mathcal{F}$ désigne l'ensemble des fonctions croissantes de $\mathbb{R}^+$ dans $\mathbb{R}_{min}$. $\mathcal{F}_0$ désigne le sous-ensemble des fonctions de $\mathcal{F}$ nulles en 0.*

Définition 5 (Minimum et Addition)**.** *Soit f et g deux fonctions de $\mathcal{F}$, le minimum $f \wedge g$ et l'addition $f + g$ sont définis par :*

$$\forall t \in \mathbb{R}^+, (f \wedge g)(t) = (f(t) \wedge g(t))$$

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

Définition 6 (Convolution et Déconvolution)**.** *Soit f et g deux fonctions de $\mathcal{F}$, la convolution $f * g$ et la déconvolution $f \oslash g$ sont définies par :*

$$\forall t \in \mathbb{R}^+, (f * g)(t) = \inf_{0 \leq s \leq t} f(t-s) + g(s) \tag{4.1}$$

$$\forall t \in \mathbb{R}^+, (f \oslash g)(t) = \sup_{0 \leq s} f(t+s) - g(s) \tag{4.2}$$

Définition 7 (Pseudo-inverse inférieur)**.** *Soit f une fonction de $\mathcal{F}$, sa pseudo-inverse inférieur f^{-1} est défini par :*

$$\forall x \in \mathbb{R}^+, f^{-1}(t) = \sup\{t | f(t) < x\} = \inf\{t | f(t) \geq x\} \tag{4.3}$$

Définition 8 (Clôture positive croissante)**.** *Soit f une fonction de $\mathbb{R}^+$ dans $\mathbb{R}_{min}$, la clôture positive $[f]^+$ et la clôture positive croissante $[f]^+_\uparrow$ sont définies par :*

$$\begin{aligned} \forall t \in \mathbb{R}^+, [f]^+(t) &= \max(f(t), 0) \\ \forall t \in \mathbb{R}^+, [f]^+_\uparrow(t) &= \sup_{0 \leq s \leq t} \max(f(s), 0) \end{aligned}$$

Il existe aussi une version de ces fonction dans le dioïde *(max, +)* :

Définition 9 (Max-plus Convolution et Max-plus Déconvolution)**.** *Soit f et g deux fonctions de $\mathcal{F}$, la max-plus convolution $f \overline{*} g$ et la max-plus déconvolution $f \overline{\oslash} g$ sont définies par :*

$$\forall t \in \mathbb{R}^+, (f \overline{*} g)(t) = \sup_{0 \leq s \leq t} f(t-s) + g(s) \tag{4.4}$$

$$\forall t \in \mathbb{R}^+, (f \overline{\oslash} g)(t) = \inf_{0 \leq s} f(t+s) - g(s) \tag{4.5}$$

4.2.3 Principales fonctions

Définition 10 (Latence)**.** *Soit $d \in \mathbb{R}$, la fonction représentant une latence de d est définie par :*

$$\forall t \in \mathbb{R}^+, \delta_d(t) = \begin{cases} 0 & \text{si } t \leq d, \\ \infty & \text{sinon.} \end{cases}$$

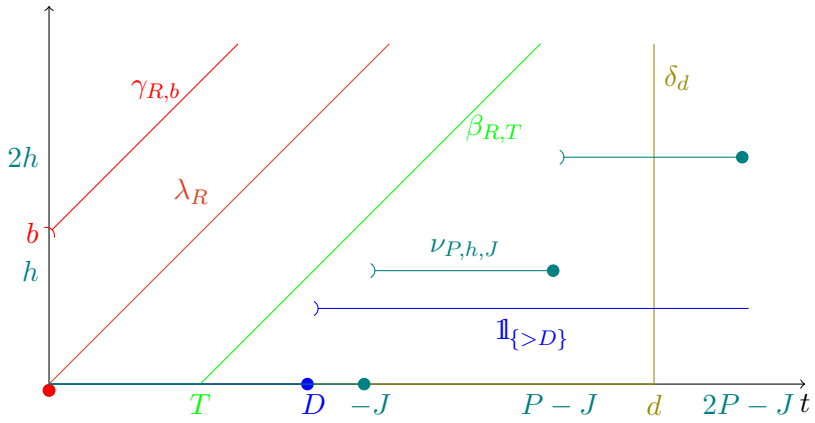


FIGURE 4.1 – Fonctions classiques

Définition 11 (Fonction test)**.** *Soit $T \in \mathbb{R}$, la fonction test est définie par :*

$$\forall t \in \mathbb{R}^+, \mathbb{1}_{\{>T\}}(t) = \begin{cases} 0 & \text{si } t \leq T, \\ 1 & \text{sinon.} \end{cases}$$

Définition 12 (Taux constant)**.** *Soit $R \in \mathbb{R}$, la fonction taux constant est définie par :*

$$\forall t \in \mathbb{R}^+, \lambda_R(t) = Rt.$$

Définition 13 (Taux constant avec latence)**.** *Soit $R \in \mathbb{R}$ et $T \in \mathbb{R}$, la fonction taux constant avec latence est définie par :*

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

Définition 14 (Fonction seau à jetons (*Token bucket*))**.** *Soit $R \in \mathbb{R}$ et $b \in \mathbb{R}$, la fonction seau à jetons est définie par :*

$$\forall t \in \mathbb{R}^+, \gamma_{R,b}(t) = (Rt + b)\,\mathbb{1}_{\{>0\}}(t).$$

Définition 15 (Fonction escalier)**.** *Soit $h, J, P \in \mathbb{R}^+$ la fonction escalier est définie par :*

$$\forall t \in \mathbb{R}^+, \nu_{P,h,J}(t) = \left[h \left\lceil \frac{t+J}{P} \right\rceil\right]^+.$$

4.3 Principe de modélisation d'un réseau

Nous allons maintenant utiliser ces fonctions afin de modéliser les mouvements des données dans le réseau, mais il est important de distinguer deux types d'objets :

— Le mouvement réel des données dans le réseau.

— Les contraintes que ces mouvements vérifient.

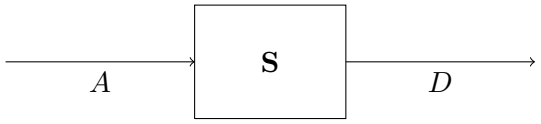


FIGURE 4.2 – Serveur : une relation entrée/sortie

L'idée de base du Calcul Réseau est de représenter un flux de données par sa fonction cumulative, A, où $A(t)$ représente la quantité totale de données envoyées par le flux jusqu'au temps t. Les fonctions étudiées sont donc bien croissantes et nulles à $t = 0$, il n'y a pas eu d'émission avant $t = 0$ (elles appartiennent à l'ensemble $\mathcal{F}_0$). Sauf mention contraire, nous supposerons que les fonctions cumulatives sont continues à gauche (l'impact de cette hypothèse a été détaillé dans [Boyer et al., 2013a]).

Définition 16 (Serveur)**.** *Un serveur est une relation entre la fonction cumulative d'un flux d'arrivée, A, et la fonction cumulative d'un flux de sortie, D qui vérifie : $A \geq D$. On note :*

$$A \xrightarrow{S} D \Longrightarrow A \geq D$$

Le délai d'un serveur ou le temps d'attente peut alors se définir à partir de A et D.

Définition 17 (Délai et occupation mémoire)**.** *Soit $A \xrightarrow{S} D$ un serveur. Le délai du flux A à un instant t est la distance horizontale entre A et D :*

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

De même on peut définir l'occupation mémoire (backlog en anglais) comme la distance verticale entre A et D :

$$b(A, D, t) = D(t) - A(t) = vDev(A, D, t)$$

On peut alors définir le délai maximal pour une entrée sortie $d(A, D)$ et l'occupation mémoire maximale $b(A, D)$ par :

$$d(A, D) = \sup_{t \in \mathbb{R}^+} \{d(A, D, t)\} = hDev(A, D) \tag{4.6}$$

$$b(A, D) = \sup_{t \in \mathbb{R}^+} \{b(A, D, t)\} = vDev(A, D) \tag{4.7}$$

Malheureusement le comportement exact d'un système est inconnu a priori ou trop complexe pour être modélisé. Le Calcul Réseau permet d'approximer le mouvement réel des données dans le réseau grâce aux contraintes qu'il vérifie. Pour cela il modélise un contrat de trafic par une "courbe d'arrivée" et modélise le serveur par une "courbe de service".

4.3.1 Modélisation du flux à travers des contrats

Courbe d'arrivée maximale

La notion de courbe d'arrivée maximale est assez intuitive, elle est utilisée pour borner le trafic sur un intervalle de temps quelconque. On dit qu'une fonction α^u est une courbe d'arrivée maximale de A si et seulement si :

$$\forall (t, s) \in \mathbb{R}_+^2,\ A(t + s) - A(t) \leq \alpha^u(s). \tag{4.8}$$

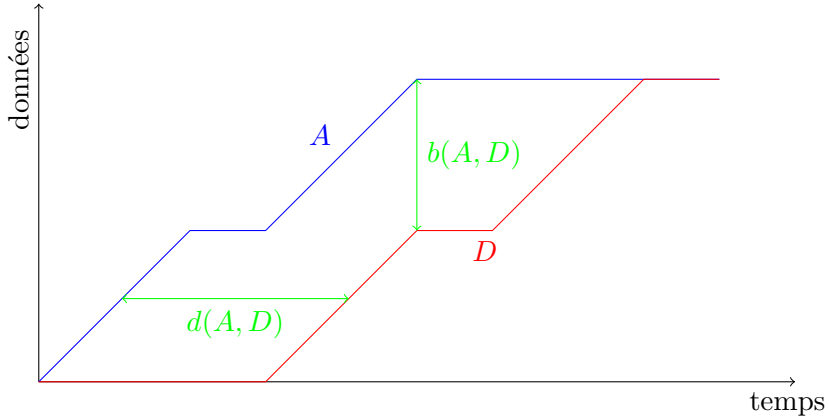


FIGURE 4.3 – Illustration du délai et de l'occupation mémoire

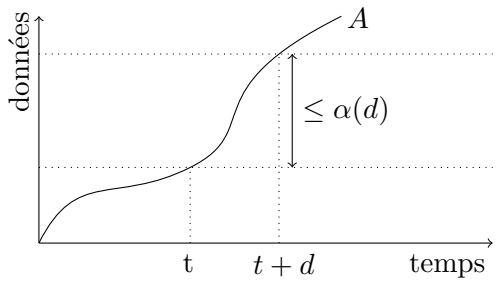


FIGURE 4.4 – Courbe d'arrivée maximale

Il est possible de donner une seconde définition d'une courbe d'arrivée maximale en se basant sur la convolution :

$$A \leq A * \alpha^u. \tag{4.9}$$

Ces deux définitions sont équivalentes, mais la seconde est bien plus pratique pour certains calculs même si elle peut sembler moins intuitive.

Il est évident qu'à un flux donné A il n'existe pas une unique courbe d'arrivée maximale. Cependant l'ensemble des courbes d'arrivée maximale de A admet un minimum qui est lui même une courbe d'arrivée maximale et que l'on peut définir par :

$$\alpha^u = A \oslash A. \tag{4.10}$$

Courbe d'arrivée minimale

De même il est possible de définir une courbe d'arrivée minimale α^l :

$$\forall (t,s) \in \mathbb{R}_+^2,\ A(t+s) - A(t) \geq \alpha^l(s). \tag{4.11}$$

$$A \leq A \overline{*} \alpha^l \tag{4.12}$$

Comme pour la courbe d'arrivée maximale, la courbe d'arrivée minimale admet un maximum qui est lui même une courbe d'arrivée minimale et que l'on peut définir par :

$$\alpha^l = A \overline{\oslash} A. \tag{4.13}$$

La courbe d'arrivée minimale est en pratique moins fréquemment utilisée que la courbe d'arrivée maximale. Ainsi en pratique on se contente de "courbe d'arrivée", noté α, pour désigner la courbe d'arrivée maximale tant que cela ne présente pas d'ambiguïté.

Courbes de service minimal

Contrairement à la notion de courbe d'arrivée qui à elle seule permet de modéliser les contraintes sur le flux, il existe plusieurs définitions de courbes de service minimal, qui ne sont pas équivalentes.

Définition 18 (Courbe de service minimal simple ou courbe de service minimal Min-plus)**.** *Un serveur S offre une courbe de service minimal simple β_{mp}^m si et seulement si :*

$$\forall A, \forall D : A \xrightarrow{S} D \Longrightarrow D \geq A * \beta_{mp}^m. \tag{4.14}$$

Une telle courbe existe toujours, il suffit de considérer $\beta_{mp}^m = 0$.

Pour la seconde définition il va nous falloir introduire la notion de période chargée (*backlogged period*).

Définition 19 (Période chargée)**.** *Soit $A \xrightarrow{S} D$ un serveur. Un intervalle I est une période chargée si et seulement si :*

$$\forall t \in I, A(t) - D(t) > 0.$$

Pour tout $t \in \mathbb{R}_+$, le début de la période chargée, noté start(t), est défini par :

$$start(t) = \sup \{u \leq t | D(u) = A(u)\}.$$

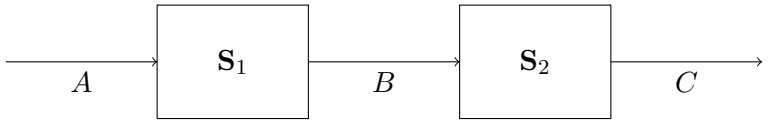


FIGURE 4.5 – Une flux traversant plusieurs serveurs

Définition 20 (Courbe de service minimal strict)**.** *Un serveur S offre une courbe de service minimal strict β^m_{strict} si et seulement si pour toute période chargée (s,t] :*

$$D(t) - D(s) \geq \beta^m_{strict}(t-s). \tag{4.15}$$

Ces deux courbes de service ne sont pas équivalentes mais il existe une relation entre elles :

Théorème 2. *Soit $A \xrightarrow{S} D$ un serveur. Si β^m est une courbe de service strict alors c'est aussi une courbe de service simple.*

Courbes de service maximal et shaper

Tout comme pour les courbes d'arrivée il est possible de définir des courbes de service maximal.

Définition 21 (Courbe de service maximal)**.** *Un serveur S offre une courbe de service maximal β^M si et seulement si :*

$$\forall A, \forall D : A \xrightarrow{S} D \Longrightarrow D \leq A * \beta^M \tag{4.16}$$

Définition 22 (Courbe de shaping)**.** *Un serveur S offre une courbe de shaping σ si et seulement si :*

$$\forall A, \forall D : A \xrightarrow{S} D \Longrightarrow D \leq D * \sigma \tag{4.17}$$

Tout comme pour les courbes d'arrivée, lorsque l'on parle de courbes de service sans donner plus d'indication il s'agit de courbes de service minimal.

Propagation des courbes d'arrivée

Lorsque le réseau n'est pas composé d'un unique lien, chaque message va traverser plusieurs serveurs, comme c'est le cas sur l'exemple de la Figure 4.5.

Il est possible d'étudier chaque section du réseau de façon indépendante mais cela rajoute une dose importante de pessimisme. En effet cela revient à considérer que le pire-cas de l'ensemble du réseau est égal à la succession des pires cas de chaque sous-partie. Ce qui n'est le cas que lorsque ces dernières sont indépendantes or ce n'est en général pas vrai, et il est possible de prendre en compte ces relations.

Ainsi il est possible de définir une courbe d'arrivée maximale est une courbe d'arrivée minimale pour le flux de sortie, en fonction des courbes d'arrivée du flux d'entrée et des courbes de service du serveur traversé.

Définition 23 (Courbes d'arrivée du flux de sortie)**.** *Soit S un serveur offrant une courbe de service minimal min plus β^m, une courbe de service maximal β^M et une courbe de service de shaping σ. Soit A un flux donné entrant ayant une courbe d'arrivée maximale α^u et une courbe d'arrivée minimale α^l. Alors pour tout flux de sortie D tel que $A \xrightarrow{S} D$ soit un serveur, le flux D admet pour courbe d'arrivée maximale η^u et pour courbe d'arrivée minimale η^l :*

$$\eta^u = ((\alpha^u * \beta^M) \oslash \beta^m) \wedge \sigma \tag{4.18}$$

$$\eta^l = \alpha^l * (\beta^m \overline{\oslash} \beta^M) \tag{4.19}$$

4.3.2 Calcul de bornes locales à partir des courbes d'arrivée et de service

L'intérêt du calcul réseau est que, connaissant une courbe d'arrivée maximale et une courbe de service minimal, on peut calculer une borne maximale sur le délai d et l'occupation mémoire b même si le comportement exact du système est inconnu.

Théorème 3 (Bornes sur le délai et l'occupation mémoire)**.** *Soit $A \xrightarrow{S} D$ un serveur tel que A ait pour courbe d'arrivée α et que S offre une courbe de service simple β. Alors le délai et l'occupation mémoire peuvent être majorés par :*

$$d(A, D) \leq d(\alpha, \beta), \tag{4.20}$$

$$b(A, D) \leq b(\alpha, \beta). \tag{4.21}$$

Il a déjà été prouvé que les bornes calculées ainsi sont justes (*tight* en anglais), c'est-à-dire atteignables [Le Boudec and Thiran, 2001]. La qualité de notre borne dépendra donc de l'évaluation de la courbe d'arrivée et de la courbe de service utilisée.

Théorème 4 (Bornes sur la taille d'une période chargée)**.** *Soit $A \xrightarrow{S} D$ un serveur tel que A ait pour courbe d'arrivée α et que S offre une courbe de service strict β. Alors la taille d'une période chargée peut être majorée par :*

$$l_{max} = \inf\{d > 0 | \alpha(d) \leq \beta(d)\} = (\alpha - \beta)^{-1}(0+). \tag{4.22}$$

Où $0+$ désigne la limite à gauche en 0 : $0+ = \lim_{x \to 0, x>0} x$.

4.3.3 Interactions de plusieurs flux

Un réseau n'est en général pas traversé par un unique flux, il est partagé par plusieurs flux. Un réseau traversé par plusieurs flux (Figure 4.6) est appelé MIMO (multiple inputs, multiple output). Dans le contexte du calcul réseau il est toujours décrit par une courbe de service. Celle ci décrit le service offert par le serveur à l'ensemble des flux.

Soit N un nombre fini de flux, on définit la fonction cumulative de l'agrégation des flux par :

$$A = \sum_{i=1}^{N} A_i$$

On peut alors définir le serveur agrégé S ainsi que N serveurs résiduels S_i comme :

$$A \xrightarrow{S} D$$

$$\forall i \leq N, A_i \xrightarrow{S_i} D_i$$

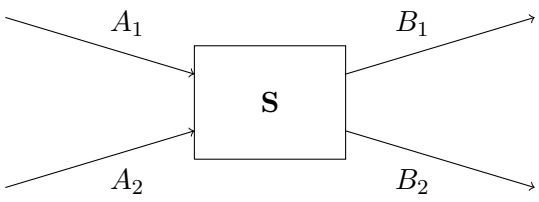


FIGURE 4.6 – Exemple d'un serveur MIMO

Définition 24 (Courbe de service agrégé et courbe de service résiduel)**.**
Un serveur MIMO fournit une courbe de service agrégé min-plus β si et seulement si le serveur agrégé S fournit une courbe de service min-plus β.
Un serveur MIMO fournit une courbe de service résiduel min-plus β_i au i-ième flux si et seulement si le serveur résiduel S_i fournit une courbe de service min-plus β_i.

Cette définition reste vraie pour une courbe de service strict.

4.3.4 Politiques de service

La politique de service, la plupart du temps au niveau d'un serveur, détermine l'ordre dans lequel les paquets présents dans la ou les files d'attente du serveur vont être servis. Il faut donc étudier la politique de service pour connaître le service offert à chaque flux individuel. Par la suite nous allons présenter les principales politiques de service et les résultats du Calcul Réseau qui leur sont associés.

Multiplexage aveugle (multiplexage arbitraire, *blind multiplexing*)

On suppose qu'il n'y a pas d'information sur la politique de service. Il peut y en avoir une de définie, mais on ne la connaît pas. En d'autres mots il faudra prendre toutes les politiques de service possibles en compte.

Théorème 5 (Service résiduel en cas de multiplexage aveugle)**.** *Soit S un serveur offrant une courbe de service strict β. Chaque flux d'arrivée A_i a pour courbe d'arrivée α_i, alors le serveur offre le service résiduel min-plus au flux j :*

$$\beta_j = \left[\beta - \sum_{i \neq j} \alpha_i\right]^+_\uparrow \tag{4.23}$$

FIFO (*First In First Out*)

Premier arrivé, premier servi. Tous les paquets sont servis dans leur ordre de leur arrivée au niveau du serveur. C'est la politique de service la plus classique.

Théorème 6 (Service résiduel en cas de FIFO)**.** *Soit S un serveur FIFO offrant une courbe de service min-plus β. Chaque flux d'arrivée A_i a pour courbe d'arrivée α_i, alors le serveur offre le service résiduel min-plus au flux j :*

$$\beta_j = \delta_{d(\sum_{i=1}^{N} \alpha_i, \beta)} \tag{4.24}$$

Le délai du flux j est alors borné par :

$$d(A_j, D_j) = d(\sum_{i=1}^{N} \alpha_i, \beta) \tag{4.25}$$

Fixed Priority : (ou *static priority*)

Les flux ont chacun une priorité, et c'est le flux le plus prioritaire qui est servi en premier.

Théorème 7 (Service résiduel en cas de *static priority*)**.** *Soit S un serveur offrant une courbe de service strict β. Chaque flux d'arrivée A_i a pour courbe d'arrivée α_i, de plus on considère que si $i < j$ le flux A_i est prioritaire sur le flux A_j. Alors le serveur offre le service résiduel strict au flux j :*

$$\beta_j = \left[\beta - \sum_{i=1}^{j-1} \alpha_i\right]^+_\uparrow \tag{4.26}$$

Ce résultat ne s'applique que dans un cadre préemptif. Pour le cas non-preemptif il faut utiliser le théorème 8.

Théorème 8. *(Non-preemptif static priority , [Bouillard et al., 2009]) Soit S un serveur offrant une courbe de service strict β, traversé par trois flux, A, A_P, A_{NP}, A_H étant plus prioritaire que A, et A_{NP} l'étant moins. Si α_P est une courbe d'arrivée de A_P et si $L_{NP}^{\max}$ est une borne supérieur de la taille des paquet de A_{NP}, alors le flux A reçoit une courbe de service résiduelle simple :*

$$\beta_A = [\beta - \alpha_P - L_{NP}^{\max}]^+_\uparrow \tag{4.27}$$

4.3.5 Calcul des délais de bout en bout

Dans les sections précédentes nous avons vu comment calculer les délais locaux. Lorsqu'il s'agit de calculer les délais de bout en bout de nombreuses réponses ont été apportées, les premiers algorithmes proposés furent *Total Flow Analysis* (TFA) et *Separated Flow Analysis* (SFA), ils sont présentés dans [Schmitt and Zdarsky, 2006]. Dans l'algorithme TFA, le principe est de calculer la somme des délais locaux et, sur chaque lien, de considérer l'ensemble des flux comme un tout. Dans l'algorithme SFA il s'agit d'une analyse bout à bout où l'on considère d'une part le flux d'intérêt et d'autre part l'ensemble des flux qui partagent une partie du réseaux avec lui. Chacun de ces deux algorithmes repose sur le calcul des courbes d'arrivée et des courbes de service que nous avons présenté précédemment.

Jérome Grieu a par la suite montré l'importance de modéliser le shaping dans ces algorithmes [Frances et al., 2006] [Grieu, 2004].

Par la suite un autre algorithme *Shaped Stairs* a été proposé [Boyer et al., 2011], ce dernier est basé sur l'algorithme TFA. De nombreuses variations de TFA et SFA ont été faites par Steffen Bondorf [Bondorf, 2015] [Bondorf and Schmitt, 2016] [Bondorf, 2017].

Plus récemment d'autres algorithmes ont été proposés : *Group Flow Analysis* [Bouillard et al., 2018] et TFA++ [Mifdaoui and Leydier, 2017].

Tous ces algorithmes ont été développés dans le cas de *blind multiplexing* ou de FIFO mais ils peuvent être adaptés pour d'autres politiques [Boyer et al., 2012] [Boyer et al., 2013b]. Mais dans tous les cas, il est nécessaire de calculer un service résiduel.

Le seul algorithme ne se basant pas sur le service résiduel est l'approche LP (*linear programs*) présentée dans [Bouillard et al., 2010] et [Bouillard and Stea, 2014]. Cependant cet algorithme ne fonctionne que dans le cas de *blind multiplexing* ou de FIFO. Dès lors, le calcul des services locaux permet de passer d'un réseau initial avec des politiques données entre files, mais FIFO dans les files, à un réseau fait de serveurs résiduels FIFO.

Dans ce manuscrit nous nous concentrerons donc sur le calcul des courbes (min-plus, strict, maximum, shaping) du service résiduel puisque ce dernier est indispensable quel que soit l'algorithme pour obtenir un délai de bout en bout.

Deuxième partie

Contributions

Chapitre 5

Améliorations des courbes de service lorsque l'émission se fait par paquet à vitesse constante

Dans ce chapitre nous allons présenter de nouveaux résultats en calcul réseau. Ces derniers n'ont pour l'instant pas été publiés.

Dans la majorité des réseaux non-préemptif, l'émission d'un paquet, à partir du moment où elle a débuté, se fait à une vitesse constante C. Or de nombreuses preuves de service résiduel ne prennent pas en compte cet aspect d'émission à vitesse constante C ce qui entraîne un pessimisme dans les résultats obtenus. Nous allons montrer comment il est possible d'améliorer les courbes de service quelle que soit la politique de service utilisée.

5.1 Introduction du problème

Dans la majorité des réseaux non-préemptifs, l'émission d'un paquet, à partir du moment où elle a débuté, se fait à une vitesse constante C. Or cette information n'est parfois pas prise en compte dans les courbes de service.

Le cas le plus classique est le suivant : un flux A se voit réserver une partie de la bande passante, notons la $0 \leq \phi_A \leq 1$. Après un certain temps T, le flux A sera servi en moyenne à la vitesse $C\phi_A$. L'expression du service est donc de la forme :

$$\beta_A(t) = C\phi_A \Big[t - T\Big]^+$$

Comment pouvons nous interpréter ce résultat ? Une façon de le comprendre est de visualiser T comme un temps d'attente avant d'être servi (flux prioritaires ou non préemption). Puis à partir de l'instant T le flux A reçoit une partie du service ; en moyenne il est servi à la vitesse $C\phi_A$. Mais en pratique ce n'est pas vraiment ce qui se passe. Durant cette seconde période le flux A est soit servi à la vitesse C, soit il n'est pas servi du tout. Ce phénomène est illustré Figure 5.1. En bleu a été représenté un exemple de courbe de service que l'on obtient souvent comme c'est le cas par exemple dans [Boyer et al., 2012], [De Azua and Boyer, 2014] mais aussi [Bouillard et al., 2018]. À partir de l'instant T la pente augmente a une vitesse $C\phi_A$. En rouge a été représentée la courbe de service réelle.

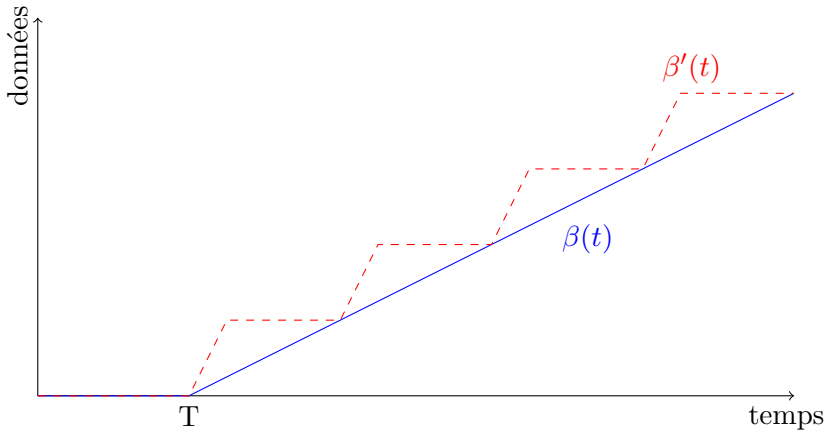


FIGURE 5.1 – Améliorations de la courbe de service lorsque l'émission se fait par paquets à vitesse constante.

À partir de l'instant T elle augmente a la vitesse C, puis reste constante avant de réaugmenter et ainsi de suite. Il est important de noter que la quantité de données envoyées en moyenne reste la même.

Ce type de courbe de service est celui que l'on obtient avec de nombreux algorithmes d'ordonnancement, notamment ceux utilisant la notion de crédit. On peut citer par exemple l'algorithme *deficit round robin* ([Boyer et al., 2012] et [Bouillard et al., 2018]) mais aussi l'algorithme *credit based shaper* utilisé dans AVB ([De Azua and Boyer, 2014]) que nous présentons plus en détail aux Chapitres 8 et 9.

L'amélioration que nous proposons ici a déjà été imaginée dans le passé. Cependant jusqu'à maintenant l'hypothèse de l'émission à vitesse constante C des paquets était soit prise en compte dans le calcul de service résiduel, le résultat dépendait donc de la politique d'intégration utilisée mais en plus cela complexifié grandement la preuve : [Sofack, 2014] et [Bouillard et al., 2018]. Soit cette amélioration restait au stade d'intuition et n'était pas démontrée comme c'est le cas par exemple dans [Boyer et al., 2012] et [Georges et al., 2011].

Dans la suite nous allons démontrer comment il est possible d'améliorer les courbes de service sous différentes hypothèses. Si l'on reprend l'exemple de la Figure 5.1 nous allons montrer comment à partir de $\beta(t)$ il est possible de calculer $\beta'(t)$.

5.2 Courbe de service strict

Nous allons commencer par calculer les nouvelles courbes de service strict. Pour cela nous distinguons deux cas. Dans le premier cas la taille des paquets est constante, l'émission se fait donc toujours sur la même durée à la vitesse C. Dans le second cas la taille des paquets est variable mais bornée.

Théorème 9 (Courbe de service strict à taille de paquets constante)**.** *Soit S un serveur et $(A, D) \in S$ avec A fait de paquets de taille fixe l, tel que lorsqu'un paquet débute son*

émission, il est émis jusqu'à la fin à la vitesse C. Si le serveur S offre à A une courbe de service strict β alors il offre aussi à A une courbe de service strict :

$$\beta' = l \left\lceil \frac{\beta}{l} \right\rceil * \lambda_C \tag{5.1}$$

Démonstration. Soit $[t, t+d]$ une période chargée et considérons la quantité de données $D(t+d) - D(t)$.

Maintenant définissons x. Si un paquet est en cours d'émission à t alors x est défini comme la date de fin d'émission de ce paquet. Nous savons que chaque paquet est émis jusqu'à la fin à vitesse constant C donc cet instant existe. S'il n'y a pas d'émission de paquet en cours à la date t alors $x = t$.

De même définissons y. Si un paquet est en cours d'émission à $t+d$ alors y est défini comme la date de début d'émission de ce paquet, sinon $y = t+d$.

x et y sont tels que $t \leq x$ et $y \leq t+d$.

— Premièrement considérons le cas particulier $y < x$:
Cela signifie que c'est le même paquet qui est émis à la date t et à la date $t+d$. Ce paquet est émis à la vitesse C donc :

$$D(t+d) - D(t) = \lambda_C(d) \geq \left(l \left\lceil \frac{\beta}{l} \right\rceil * \lambda_C \right)(d) = \beta'(d) \tag{5.2}$$

— Maintenant supposons que $x \leq y$:
Sur les intervalles $[t, x]$ et $[y, t+d]$, une partie de paquet est émise, à vitesse constante C, donc :

$$D(t+d) - D(t) = D(y) - D(x) + C(x-t) + C(d+t-y) \tag{5.3}$$

S offre à A une courbe de service strict β, donc $D(y) - D(x) \geq \beta(y-x)$.
Mais par construction, il y a $n \in \mathbb{N}$ paquets de taille l émis durant $[x, y]$ c'est-à-dire $D(y) - D(x) = nl$, donc $nl \geq \beta(y-x) \Rightarrow \lceil n \rceil = n \geq \left\lceil \frac{\beta(y-x)}{l} \right\rceil$ et

$$D(t+d) - D(t) \geq l \left\lceil \frac{\beta(y-x)}{l} \right\rceil + C(x-y+d) \tag{5.4}$$

Notons $s = d + x - y$

$$\begin{aligned} D(t+d) - D(t) &\geq l \left\lceil \frac{\beta(d-s)}{l} \right\rceil + C(s) && (5.5) \\ &\geq \inf_{0 \leq s \leq d} \left(l \left\lceil \frac{\beta(d-s)}{l} \right\rceil + \lambda_C(s) \right) && (5.6) \\ &\geq \left(l \left\lceil \frac{\beta}{l} \right\rceil * \lambda_C \right)(d) = \beta'(d) && (5.7) \end{aligned}$$

□

Théorème 10 (Courbe de service strict à taille de paquets variable)**.** *Soit S un serveur et $(A, D) \in S$ avec A fait de paquets de tailles variables $l_{min} \leq l \leq l_{max}$, tel que lorsqu'un paquet débute son émission, il est émis jusqu'à la fin à la vitesse C. Si le serveur S offre à A une courbe de service strict β alors il offre aussi à A une courbe de service strict :*

$$\beta' = l_{min} \left\lceil \frac{\beta}{l_{max}} \right\rceil * \lambda_C \tag{5.8}$$

Démonstration. Soit $[t, t+d]$ une période chargée et considérons la quantité de données $D(t+d) - D(t)$.

Maintenant définissons x. Si un paquet est en cours d'émission à t alors x est défini comme la date de fin d'émission de ce paquet, nous savons que chaque paquet est émis jusqu'à la fin à vitesse constant C donc cet instant existe. S'il n'y a pas d'émission de paquet en cours à la date t alors $x = t$.

De même définissons y. Si un paquet est en cours d'émission à $t + d$ alors y est défini comme la date de début d'émission de ce paquet, sinon $y = t + d$.

x et y sont tels que $t \leq x$ et $y \leq t + d$.

— Premièrement considérons le cas particulier $y < x$:
Cela signifie que c'est le même paquet qui est émis à la date t et à la date $t + d$. Ce paquet est émis à la vitesse C donc :

$$D(t+d) - D(t) = \lambda_C(d) \geq \left(l_{min} \left\lceil \frac{\beta}{l_{max}} \right\rceil * \lambda_C \right)(d) = \beta'(d) \tag{5.9}$$

— Maintenant supposons que $x \leq y$:
Sur les intervalles $[t, x]$ et $[y, t+d]$, une partie de paquet est émise, à vitesse constante C, donc :

$$D(t+d) - D(t) = D(y) - D(x) + C(x - t) + C(d + t - y) \tag{5.10}$$

S offre à A une courbe de service strict β, donc $D(y) - D(x) \geq \beta(y - x)$.
Mais par construction, il y a $n \in \mathbb{N}$ paquets émis durant $[x, y]$. Pour la preuve introduisons $l^{x,y}_{mean}$, la taille moyenne de ces n paquets. Il est possible d'en déduire que $D(y) - D(x) = n l^{x,y}_{mean} \geq n l_{min}$, donc $n l^{x,y}_{mean} \geq \beta(y-x) \Rightarrow \lceil n \rceil = n \geq \left\lceil \frac{\beta(y-x)}{l^{x,y}_{mean}} \right\rceil \geq \left\lceil \frac{\beta(y-x)}{l_{max}} \right\rceil$ et

$$D(t+d) - D(t) \geq l_{min} \left\lceil \frac{\beta(y-x)}{l_{max}} \right\rceil + C(x - y + d) \tag{5.11}$$

Notons $s = d + x - y$

$$D(t+d) - D(t) \geq l_{min} \left\lceil \frac{\beta(d-s)}{l_{max}} \right\rceil + C(s) \tag{5.12}$$

$$\geq \inf_{0 \leq s \leq d} \left(l_{min} \left\lceil \frac{\beta(d-s)}{l_{max}} \right\rceil + \lambda_C(s) \right) \tag{5.13}$$

$$\geq \left(l_{min} \left\lceil \frac{\beta}{l_{max}} \right\rceil * \lambda_C \right)(d) = \beta'(d) \tag{5.14}$$

□

Corolaire 1 (Courbe de service strict à taille de paquet variable)**.** *Soit S un serveur et $(A, D) \in S$ avec A fait de paquets de tailles variables $l_{min} \leq l \leq l_{max}$, tel que lorsqu'un paquet débute son émission, il est émis jusqu'à la fin à la vitesse C. Si le serveur S offre à A une courbe de service strict β alors il offre aussi à A une courbe de service strict :*

$$\beta' = \max(\beta, l_{min} \left\lceil \frac{\beta}{l_{max}} \right\rceil * \lambda_C) \tag{5.15}$$

Démonstration. Le maximum de deux courbes de service est une courbe de service d'où le résultat. □

5.3 Courbe de service simple

Nous allons maintenant montrer comment les courbes de service simple peuvent elles aussi être améliorées. Ces théorèmes ne s'appliquent que lorsque la taille des paquets est constante cependant nous distinguons la situation où les données arrivent par paquets et la situation où chaque paquet arrive sur le réseau plus vite que la vitesse d'émission C

Théorème 11 (Courbe de service simple lorsque A est paquétisé)**.** *Soit S un serveur et $(A, D) \in S$ avec A continue à gauche et fait de paquets de taille fixe l, tel que lorsqu'un paquet débute son émission, il est émis jusqu'à la fin à la vitesse C. Si le serveur S offre à A une courbe de service simple β avec β continue à gauche et si A est paquétisé c'est-à-dire $A = l \left\lceil \frac{A}{l} \right\rceil$ alors le serveur S offre aussi à A une courbe de service simple :*

$$\beta' = l \left\lceil \frac{\beta}{l} \right\rceil * \lambda_C \tag{5.16}$$

Démonstration. Soit $t \in \mathbb{R}^+$, et considérons la quantité de données $D(t)$.

Si un paquet est en cours d'émission à la date t alors s est définie comme la date de début d'émission de ce paquet, nous savons que chaque paquet est émis jusqu'à la fin à vitesse constante C donc cet instant existe. S'il n'y a pas d'émission de paquet à la date t alors $s = t$.

Durant l'intervalle $[s, t]$ une partie de paquet est émise, à la vitesse constante C, donc :

$$D(t) = D(s) + C(t - s) \tag{5.17}$$

S offre à A une courbe de service simple β, donc $D(s) \geq (A * \beta)(s)$.
Mais par construction, il y a $n \in \mathbb{N}$ paquets de taille l émis durant $[0, s]$ c'est-à-dire $D(s) = nl$, donc $nl \geq (A * \beta)(s) \Rightarrow \lceil n \rceil = n \geq \left\lceil \frac{(A*\beta)(s)}{l} \right\rceil$ et

$$D(t) \geq l \left\lceil \frac{(A * \beta)(s)}{l} \right\rceil + C(t - s) \tag{5.18}$$

De plus puisque A et β sont continues à gauche

$$\exists u \leq s \text{ tel que } (A * \beta)(s) = A(u) + \beta(s - u) \tag{5.19}$$

Nous pouvons donc en déduire

$$D(t) \geq l \left\lceil \frac{A(u) + \beta(s-u)}{l} \right\rceil + C(t-s) \tag{5.20}$$

$$\geq l \left\lceil \frac{A(u)}{l} + \frac{\beta(s-u)}{l} \right\rceil + C(t-s) \tag{5.21}$$

Mais A paquétisé est donc $A = l \left\lceil \frac{A}{l} \right\rceil$, donc :

$$D(t) \geq l \left\lceil \left\lceil \frac{A(u)}{l} \right\rceil + \frac{\beta(s-u)}{l} \right\rceil + C(t-s) \tag{5.22}$$

$$\geq l \left\lceil \frac{A(u)}{l} \right\rceil + l \left\lceil \frac{\beta(s-u)}{l} \right\rceil + C(t-s) \tag{5.23}$$

$$\geq A(u) + l \left\lceil \frac{\beta(s-u)}{l} \right\rceil + C(t-s) \tag{5.24}$$

$$\geq \left(A * l \left\lceil \frac{\beta}{l} \right\rceil \right)(s) + \lambda_C(t-s) \tag{5.25}$$

$$\geq \left(A * \left(l \left\lceil \frac{\beta}{l} \right\rceil * \lambda_C \right) \right)(t) = (A * \beta')(t) \tag{5.26}$$

□

Théorème 12 (Courbe de service simple lorsque A arrive plus vite que la vitesse C)**.** *Soit S un serveur et $(A, D) \in S$ avec A continue à gauche et fait de paquets de taille fixe l, tel que lorsqu'un paquet débute son émission, il est émis jusqu'à la fin à la vitesse C. Si le serveur S offre à A une courbe de service simple β avec β continue et si A arrive plus vite que la vitesse constante C c'est-à-dire $A \geq l \left\lceil \frac{A}{l} \right\rceil * \lambda_C$ alors le serveur S offre aussi à A une courbe de service simple :*

$$\beta' = l \left\lceil \frac{\beta * \lambda_C}{l} \right\rceil * \lambda_C \tag{5.27}$$

Démonstration. Soit $t \in \mathbb{R}^+$, et considérons la quantité de données $D(t)$.

Si un paquet est en cours d'émission à la date t alors s est définie comme la date de début d'émission de ce paquet, nous savons que chaque paquet est émis jusqu'à la fin à vitesse constante C donc cet instant existe. S'il n'y a pas d'émission de paquet à la date t alors $s = t$.

Durant l'intervalle $[s, t]$ une partie de paquet est émise, à la vitesse constante C, donc :

$$D(t) = D(s) + C(t-s) \tag{5.28}$$

S offre à A une courbe de service simple β, donc $D(s) \geq (A * \beta)(s)$.
Mais par construction, il y a $n \in \mathbb{N}$ paquets de taille l émis durant $[0, s]$ c'est-à-dire $D(s) = nl$, donc $nl \geq (A * \beta)(s) \Rightarrow \lceil n \rceil = n \geq \left\lceil \frac{(A*\beta)(s)}{l} \right\rceil$ et

$$D(t) \geq l \left\lceil \frac{(A * \beta)(s)}{l} \right\rceil + C(t-s) \tag{5.29}$$

Puisque A arrive que la vitesse constante C c'est-à-dire $A \geq l \left\lceil \frac{A}{l} \right\rceil * \lambda_C$

$$D(t) \geq l \left\lceil \frac{\left(l \left\lceil \frac{A}{l} \right\rceil * \lambda_C * \beta\right)(s)}{l} \right\rceil + \lambda_C(t-s) \tag{5.30}$$

De plus A est continue à gauche et puisque β est continue alors $\beta * \lambda_C$ est continue

$$\exists u \leq s \text{ tel que } \left(l \left\lceil \frac{A}{l} \right\rceil * \beta * \lambda_C\right)(s) = \left(l \left\lceil \frac{A}{l} \right\rceil\right)(u) + (\beta * \lambda_C)(s-u) \tag{5.31}$$

On peut donc en déduire que :

$$D(t) \geq l \left\lceil \frac{\left(l \left\lceil \frac{A}{l} \right\rceil\right)(u) + (\beta * \lambda_C)(s-u)}{l} \right\rceil + \lambda_C(t-s) \tag{5.32}$$

$$\geq l \left\lceil \left\lceil \frac{A(u)}{l} \right\rceil + \frac{(\beta * \lambda_C)(s-u)}{l} \right\rceil + \lambda_C(t-s) \tag{5.33}$$

$$\geq l \left\lceil \frac{A(u)}{l} \right\rceil + l \left\lceil \frac{(\beta * \lambda_C)(s-u)}{l} \right\rceil + \lambda_C(t-s) \tag{5.34}$$

$$\geq \left(l \left\lceil \frac{A}{l} \right\rceil * l \left\lceil \frac{\beta * \lambda_C}{l} \right\rceil\right)(s) + \lambda_C(t-s) \tag{5.35}$$

$$\geq \left(l \left\lceil \frac{A}{l} \right\rceil * l \left\lceil \frac{\beta * \lambda_C}{l} \right\rceil * \lambda_C\right)(t) \tag{5.36}$$

$$\geq \left(A * l \left\lceil \frac{\beta * \lambda_C}{l} \right\rceil * \lambda_C\right)(t) = (A * \beta')(t) \tag{5.37}$$

□

5.4 Conclusion

De nombreuses preuves de service résiduel ne prennent pas en compte le fait que l'émission d'un paquet, à partir du moment où elle a débuté, se fait à une vitesse constante C. En effet calculer un service résiduel en considérant ce fait peut compliquer énormément la preuve (voir les travaux de William Mangoua Sofack [Sofack, 2014]).

Dans ce chapitre nous avons développé un résultat novateur en Calcul Réseau afin d'améliorer les courbes de services d'un flux. Nous avons montré comment sous différentes hypothèses il est possible d'améliorer les courbes de services obtenues. Ce résultat est générique, il s'applique à toute politique de service.

Sur la Figure 5.2 nous avons représenté le gain obtenu grâce à notre amélioration. Si nous reprenons la situation décrite au début de ce chapitre d'un flux A qui se voit réserver une part de la bande passante ϕ_A alors le gain sera d'autant plus important que ϕ_A sera petit ce qui correspond à une situation où de nombreuses files se partagent la bande passante.

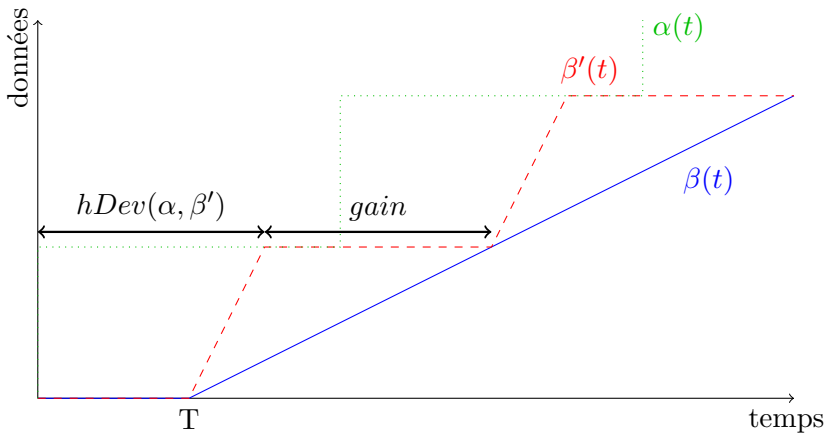


FIGURE 5.2 – Réduction du délai grâce à l'amélioration de la courbe de service.

Chapitre 6

Modélisation d'une synchronisation faible dans le cas du bus CAN

6.1 Définition d'un système faiblement synchronisé

Dans cette section nous allons présenter comment l'utilisation d'offset (c'est-à-dire émettre les messages sur le réseau avec un décalage dans leurs dates d'émission) permet de diminuer le temps de traversée.

Nous présenterons tout d'abord le cas des messages "*Time-Triggered*", aussi désigné comme le cas d'une horloge commune. Dans cette situation, les offsets (ou dates d'émission) se réfèrent à une même horloge partagée par tous les éléments du réseau.

Nous présenterons ensuite le cas d'offsets locaux, dans cette situation chaque élément du réseau a sa propre horloge, on parle d'horloges locales, et les offsets sont donc locaux et non plus globaux.

Enfin nous proposerons un compromis entre ces deux solutions, l'utilisation d'une synchronisation faible, c'est-à-dire l'utilisation d'horloges locales synchronisées entre elles mais avec une précision faible.

Afin d'évaluer le gain apporté par la synchronisation faible on se basera sur un réseau de type bus CAN. Ce dernier a été présenté en détail au Chapitre 2. En effet il présente plusieurs avantages. Tous d'abord il s'agit d'un réseau en bus au fonctionnement simple à modéliser. Ensuite c'est un bus très répandu, qui possède la possibilité d'utiliser des horloges locales et qui peut donc être adapté à une synchronisation faible. Enfin le bus CAN a déjà été largement étudié. Le temps de traversée dans le cas de l'utilisation d'offset dans le cas d'horloges locales a été l'objet de nombreuses études qui nous permettrons de comparer nos résultats. On peut par exemple citer [Tindell et al., 1994] premiers résultats sur le temps de traversée dans le cas du réseau CAN ; ces résultats ont par la suite été revus dans ce qui est maintenant devenu une référence lorsqu'il s'agit de borner le temps de traversée dans le cas du bus CAN [Davis et al., 2007]. De plus le calcul réseau a déjà été utilisé avec succès dans le cas du bus CAN avec par exemple les articles [Klehmet et al., 2008] et [Sofack and Boyer, 2012].

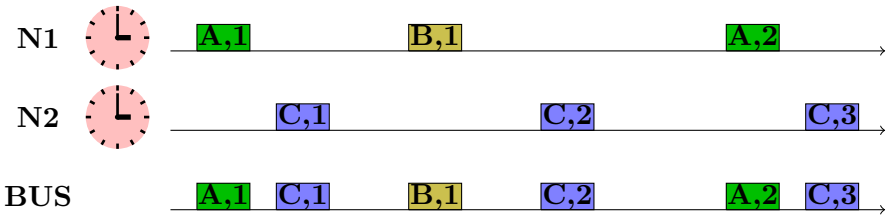


FIGURE 6.1 – Exemple d'ordonnancement sur un bus dans le cas d'une horloge commune.

6.1.1 Analyse des avantages et désavantages de la synchronisation au sein du réseau

Comme présenté dans le Chapitre 2, il existe énormément de sortes de réseaux. En particulier un certain nombre d'entre eux reposent sur des messages "*Time-Triggered*", c'est à dire dont la date d'émission sur le réseau a été prévue *a priori* (Section 2.2.2).

Cela nécessite de synchroniser tous les éléments du réseau, on parle alors d'*horloge commune*. En recourant uniquement à des messages *Time-Triggered*, l'émission de messages sur le réseau est déterministe, il est possible de minimiser le temps de traversée des message sur le réseau en évitant les contentions, c'est-à-dire une situation où deux messages devraient être envoyés sur le medium en même temps. Un exemple d'une telle situation est visible Figure 6.1. Dans cet exemple on considère le cas d'un réseau de type bus auquel sont connectés deux nœuds **N1** et **N2**. **N1** émet deux flux sur le bus, le flux **A** en vert et le flux **B** en jaune. **N2** quant à lui émet le flux **C** en bleu. En haut on peut voir l'ordonnancement de chacun des deux nœuds en fonction du temps. Une fenêtre est dédiée à chaque message. L'ordonnancement ayant été fait de façon efficace et les deux nœuds étant synchronisés, c'est-à-dire que leurs horloges sont égales, l'accès au bus se fait sans contention. Lorsque l'un des deux nœuds veut émettre il a la garantie que le bus sera disponible et qu'il ne va pas devoir attendre.

Cependant cette synchronisation est coûteuse à obtenir en pratique. En effet elle nécessite très souvent du matériel dédié qui rajoute un coût qui peut être trop important en pratique.

Cependant il est possible d'utiliser des messages avec offsets sans synchroniser les nœuds du réseau. Dans ce cas chaque nœud se référera à son horloge locale et aura son propre ordonnancement. Si l'on reprend l'exemple précédent avec nos deux nœuds cela revient à supposer maintenant que leurs horloges ne sont pas synchronisées. Deux exemples d'une telle situation sont visibles Figure 6.2. Il n'y a plus un unique exemple comme précédemment car la différence qui existe entre les deux horloges, la phase, est inconnue. Sur cette figure deux cas particuliers ont été représentés. Pour chacun d'eux l'ordonnancement local reste le même. Cet ordonnancement permet d'éviter les contentions intra-nœuds, c'est-à-dire entre le flux **A** et le flux **B**. Cependant des contentions inter-nœuds, entre flux de différents nœuds, peuvent avoir lieu. Dans le cas de la Figure 6.2a il y a une contention entre le flux **A** et le flux **C**, ce qui entraîne un délai supplémentaire pour le flux **C** alors que pour le flux **A** et le flux **B** le temps de traversée du réseau reste le même que dans le cas d'une horloge commune. Dans le cas de la Figure 6.2b la contention a lieu entre le flux **B** et le flux **C** et c'est cette fois le flux **B** qui subit un délai supplémentaire. Seulement deux cas ont été représentés mais il est facile de comprendre que chaque flux

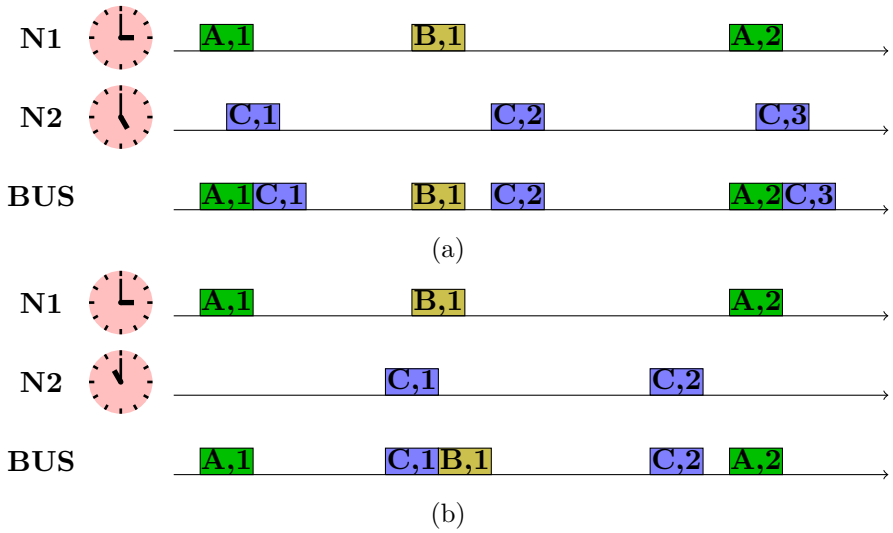


FIGURE 6.2 – Exemples d'ordonnancements sur un bus dans le cas d'horloges locales.

pourrait subir ce délai supplémentaire. De plus comme la valeur de la phase entre les horloges est inconnue a priori, lorsqu'on cherche à borner le temps de traversée de chaque message en envisageant le pire cas, il est nécessaire de considérer que chaque message sera retardé. Cependant, ce retard est limité, en effet l'utilisation d'un ordonnancement des messages permet de lisser le trafic. Dans notre exemple le flux **C** peut être en conflit pour accéder au bus avec le flux **A** ou avec le flux **B** mais jamais les deux à la fois.

Pour résumer l'utilisation d'offset, c'est-à-dire décaler les dates d'émissions des messages permet de diminuer le temps de traversée maximum des messages. En effet il limite les contentions. Si toutes les horloges sont synchronisées (on a donc une unique horloge commune) il est même possible d'éviter toute contention, mais cela a un coût important. Il faut du matériel dédié ainsi que des protocoles de synchronisation complexes. Si les horloges ne sont pas synchronisées et que chaque nœud se réfère à sa propre horloge, cela permet d'éviter les contentions entre messages d'un même émetteur et de réduire les contentions entre deux messages de nœuds différents. En effet l'utilisation d'un ordonnancement permet de lisser le trafic et donc de réduire les pics d'émission. Cependant dans le pire cas chaque message peut être retardé au moins une fois par un message d'une autre station. Chacune de ces solutions a donc ses avantages et ses inconvénients, nous allons proposer dans la suite une solution intermédiaire permettant d'obtenir des délais moins importants que lors de l'utilisation d'horloges locales mais sans avoir le coût de la solution d'une horloge commune.

6.1.2 Intérêt d'un système faiblement synchronisé

Dans cette section nous allons envisager une situation reposant sur des horloges locales synchronisées entre elles mais avec une précision faible. Il existe donc une phase entre ces horloges. Si cette phase entre horloges locales n'est pas négligeable devant le temps d'émission d'un message, on ne peut pas parler d'horloge commune. Dans ce cas on parle de *synchronisation faible*. Cette synchronisation est moins coûteuse qu'une synchronisation

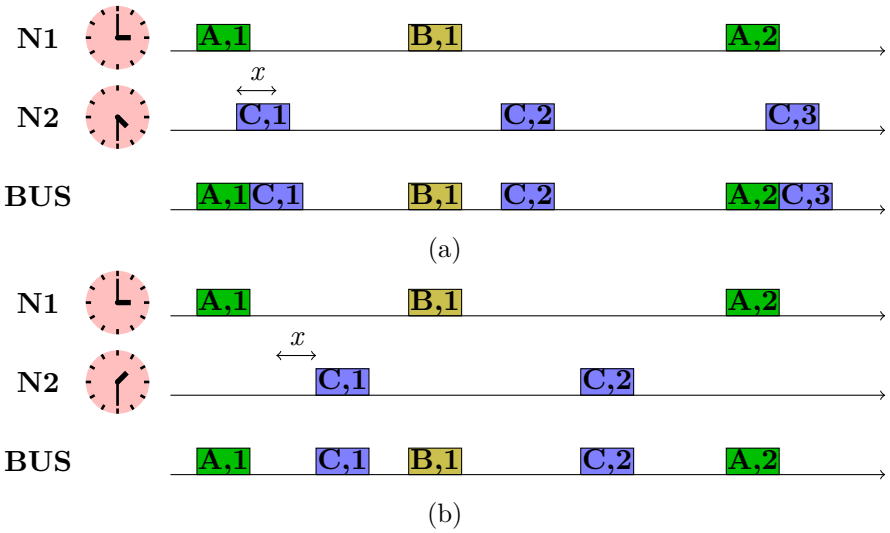


FIGURE 6.3 – Exemples d'ordonnancements sur un bus faiblement synchronisé.

parfaite avec une horloge commune (phases entre les horloges négligeables) mais il reste à montrer qu'elle présente un intérêt en terme de délai par rapport à l'utilisation d'horloges locales non synchronisées entre elles. Pour montrer que c'est bien le cas reprenons notre exemple précédent et supposons qu'il existe une phase entre les horloges inconnues mais que l'on est capable de borner par x. On va considérer que x est de l'ordre de grandeur du temps d'émission d'un message. Il n'est alors plus possible de négliger cette dernière et de considérer qu'on a une horloge commune. Sur la Figure 6.3 on a représenté les deux situations extrêmes pour les valeurs de phase ($\pm x$). On peut voir qu'une contention peut avoir lieu entre le flux **A** et le flux **C**. Dans le pire cas le flux **C** va donc subir un délai supplémentaire. Mais contrairement au cas des horloges locales, il n'existe pas de valeur de la phase qui entraînerait une contention entre le flux **B** et le flux **C**. La conséquence de ce dernier point est que l'on sait que quelle que soit la situation, lorsque **N1** voudra envoyer un message du flux **B** il aura la garantie que le bus sera disponible. Le délai maximum des messages du flux **B** sera donc minimum. Cette solution est donc un compromis entre une synchronisation parfaite avec tous les nœuds utilisant une horloge commune et un réseau non synchronisé où chaque nœud a son horloge locale. De plus une synchronisation faible permet d'obtenir des délais moins importants que lors de l'utilisation d'horloges locales mais sans avoir le coût de la solution d'une horloge commune.

Mécanisme de mise en place d'une synchronisation faible

Avant d'aller plus loin, il est nécessaire de comprendre ce qui rend la synchronisation entre différentes machines si difficile et coûteuse. Pour y parvenir nous avons représenté l'échange d'un message entre deux nœuds à travers le réseau (voir Figure 6.4). Il est possible de modéliser cet échange en utilisant le modèle OSI. Ici on s'est contenté de représenter trois couches : la couche physique, la couche liaison et la couche application (qui représente ici l'ensemble des couches hautes). Le message doit traverser toutes ces couches lors de la communication. Son temps de traversée total peut être décomposé

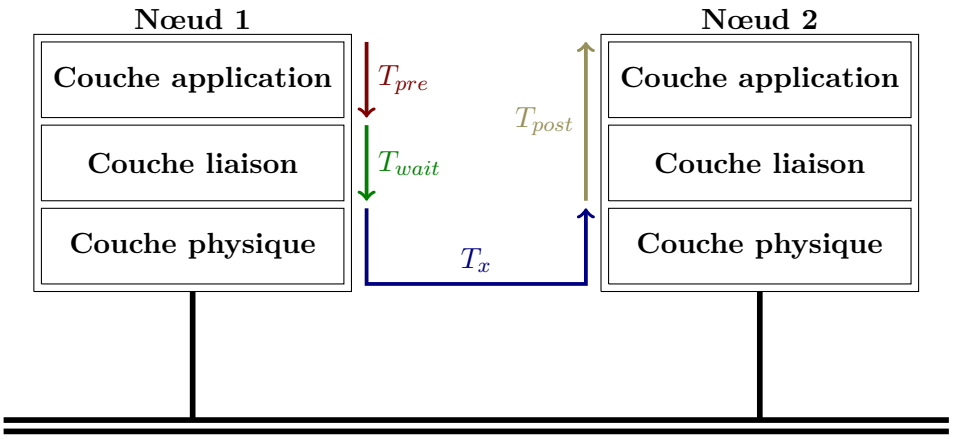


FIGURE 6.4 – Traversée des différentes couches lors de l'émission d'un message.

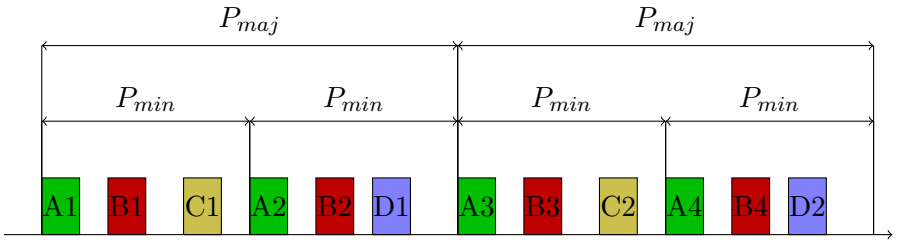


FIGURE 6.5 – Exemple d'ordonnancement utilisant le concept périodes mineures et majeures.

suivant [Lian et al., 2002] en quatre composants : un temps de pré-traitement T_{pre}, un temps d'attente T_{wait}, un temps de transmission T_x et un temps de post-traitement T_{post}. Le temps d'attente et le temps de transmission sont les temps calculés à l'aide du calcul réseau. Il s'agit du temps d'attente avant que le medium soit disponible et du temps que mettent les données physiques à être envoyées. Le temps de pré-traitement est le temps nécessaire pour acquérir des données de l'environnement et les coder dans des données de réseau, tandis que le temps de post-traitement est le temps requis pour décoder les données de réseau et les transmettre à l'environnement. Ces temps sont inconnus à la conception et dépendent des caractéristiques du logiciel et du matériel de l'appareil utilisé, en général ce temps est considéré constant et sa variabilité négligeable. Or il a été montré dans [Lian et al., 2002] que cette variabilité n'est pas négligeable en pratique et en particulier lorsque l'on cherche à synchroniser plusieurs machines. Lorsque l'on veut obtenir une horloge commune, c'est-à-dire un système synchronisé avec une phase entre les horloges négligeable devant le temps d'émission d'un message il faut donc passer par du matériel et des logiciels dédiés conçus spécifiquement pour résoudre ce problème c'est-à-dire avec une variabilité réellement négligeable ou réaliser le traitement dans la couche physique. Mais comme cela a déjà été dit cette solution n'est pas toujours satisfaisante puisqu'elle rajoute un coût supplémentaire.

Maintenant que nous avons expliqué ce qui rend la synchronisation entre les horloges coûteuse, nous allons montrer qu'il est possible à moindre coût d'obtenir une synchroni-

sation faible. Nous allons donc proposer un exemple de protocole de synchronisation qui permet d'obtenir une synchronisation faible. Il ne s'agit que d'un exemple, tous les résultats présentés dans ce chapitre peuvent être utilisés avec n'importe quel protocole de synchronisation, celui ci n'en est qu'un parmi beaucoup d'autres. Plusieurs protocoles de synchronisation existent, on peut citer par exemple le *Precision Time Protocol* [Lee et al., 2005], celui que nous allons proposer ici considère que le système étudié se base sur un ordonnancement utilisant un système de périodes majeures et mineures (voir Figure 6.5) et se base sur un fonctionnement maître/esclave. Une horloge, celle du maître, est définie comme la référence. C'est sur elle que se basent toutes les autres (esclaves). Toutes les périodes majeures, le maître envoie un message qui est par la suite utilisé par tous les nœuds esclaves afin de mettre à jour leur horloge. Les horloges ne sont pas idéales et leur fréquence n'est jamais exactement celle prévue à la conception c'est ce qu'on nomme la dérive d'horloge. Il est donc nécessaire d'avoir une émission périodique de ce message de synchronisation afin de prendre en compte ce phénomène de dérive d'horloge et remettre à jour la valeur des horloges des nœuds esclaves. Nous avons supposé que ce message de synchronisation était le message le plus prioritaire afin de réduire le temps de traversée. Cependant comme expliqué précédemment ce temps de traversée ne sera pas le même pour tous les nœuds puisque le temps de pré-traitement T_{pre} et le temps de post-traitement T_{post} (Figure 6.4) dépendront directement du matériel utilisé, T_{wait} de l'utilisation du bus au moment de l'émission du message et T_x de la distance de câblage entre l'émetteur et le récepteur et de la taille du message. Dans l'article [Lian et al., 2002] on trouve une borne pour les temps de traitement pour du matériel classique du bus CAN : $0.5ms < Tpre + Tpost < 1ms$. Puisque l'on a décidé que le message de synchronisation serait le plus prioritaire et que l'émission sur le bus CAN est non-préemptive on sait que $0 \leq T_{wait} < 0.5ms$. Enfin on peut considérer que T_x est constant et connu. En effet la taille du message de synchronisation sera toujours la même et la différence de distance entre le maître et les esclaves est négligeable par rapport à la vitesse de transmission du message. En conclusion on a :

$$0.5ms < T_{pre} + T_{post} + T_{wait} < 1.5ms \tag{6.1}$$

Il est donc possible d'avoir une synchronisation de $\pm$ 1 ms dans le cas du bus CAN de façon très simple et peu coûteuse. Bien sûr le protocole que nous venons de décrire peut être amélioré. Il s'agissait juste d'estimer un ordre de grandeur de la précision que l'on peut espérer obtenir avec un coût minimal. Dans la suite, lors de l'évaluation des délais maximaux, nous supposerons que c'est ce protocole de synchronisation qui est utilisé, cependant nos résultats peuvent être utilisés avec n'importe quel protocole de synchronisation.

Maintenant que nous avons vu comment obtenir une synchronisation faible et que nous avons avons montré en quoi une synchronisation faible pourrait permettre de réduire le pire temps de traversée des messages il nous reste à développer un modèle capable d'analyser un tel système.

6.2 Calcul des bornes du temps de traversée du réseau dans le cas d'un bus CAN faiblement synchronisé

6.2.1 Modèle considéré

Pour que notre modèle puisse être appliqué sur des configurations utilisées en pratique il est important qu'il soit le plus complet possible. La première chose à définir est le type de trafic étudié. Dans notre cas nous considérerons que deux types de trafic coexistent dans notre système. Le trafic localement synchrone *Time-Triggered* et le trafic asynchrone *Event-Triggered*.

Trafic synchrone

Le trafic synchrone comprend tous les messages envoyés de façon périodique. Le plus souvent la taille des messages est fixe, dans le cas contraire une taille maximale des paquets doit être respectée par l'émetteur. Ce type de trafic peut être utilisé pour transmettre de façon périodique des données depuis un capteur par exemple. L'émission étant périodique il est possible de fixer la date d'émission de chaque paquet en fixant la date d'émission du premier paquet. Lorsque l'on parle de date ici c'est par rapport à l'horloge locale de l'émetteur, que celle ci soit ou pas synchronisée au reste du réseau. On désigne alors par "offset" la date d'émission de ce premier paquet. Si l'on note O, l'offset d'un flux A et P sa période alors le k-ième message sera envoyé à la date $t = O + (k-1)P$.

Si l'on considère le cas où tous les messages font la même taille l l'expression du flux d'arrivée A ainsi que sa meilleure courbe d'arrivée sont connues :

$$A(t) = \left[l \left\lceil \frac{t-O}{P} \right\rceil \right]^+ \tag{6.2}$$

$$\alpha(t) = \left[l \left\lceil \frac{t}{P} \right\rceil \right]^+ \tag{6.3}$$

Si l'on considère le cas où la taille des messages n'est plus connue mais est bornée par l_{max} l'expression du flux d'arrivée A n'est plus connue mais il est possible de calculer la courbe d'arrivée :

$$\alpha(t) = \left[l_{max} \left\lceil \frac{t}{P} \right\rceil \right]^+ \tag{6.4}$$

Trafic asynchrone

Le trafic asynchrone comprend tous les messages qui ne sont pas envoyés de façon périodique et dont le comportement est dirigé par les événements. Ce type de trafic est par exemple utilisé pour la transmission des messages d'alarmes qui par définition ne peuvent pas être prévus *a priori*. Afin de garantir qu'un flux ne monopolisera pas l'ensemble de la bande passante l'émetteur doit respecter des "contrats" sur son émission. Chaque flux est caractérisé par son contrat, dans le cas du bus CAN on parle de flux sporadique, il s'agit du contrat le plus courant en pratique. Chaque flux est caractérisé par une taille

maximale de paquet l_{max}. Tous les paquets d'un flux doivent respecter cette contrainte. De plus l'émetteur se doit de respecter une distance minimale entre deux paquets consécutifs, ce temps est noté MUT pour *Minimal Update Time*. Contrairement au trafic périodique où l'on sait que le message suivant sera transmis exactement après la durée d'une période pour le trafic asynchrone on sait seulement qu'il sera transmis au plus tôt après la durée du MUT.

Comme les dates d'arrivée ne sont pas connues il n'est pas possible d'avoir une expression du flux d'arrivée, cependant l'existence des contraintes sur la taille des paquet l_{max} et sur la distance minimale entre deux paquets consécutifs MUT permet de définir la meilleur courbe d'arrivée :

$$\alpha(t) = \left[l_{max} \left\lceil \frac{t}{MUT} \right\rceil \right]^{+} \tag{6.5}$$

Phase d'arbitrage

Dans le Chapitre 2 la stratégie d'accès au medium dans le cas du bus CAN a déjà été présentée. Pour rappel il s'agit de la technique CSMA/CD il s'agit d'une politique d'intégration **NP-SP** *non-preemtive static-priority*. Chaque flux a une priorité qui lui être propre et qui sert aussi d'identifiant. Lorsque plusieurs messages veulent accéder au bus en même temps c'est celui avec la plus grande priorité qui est transmis. Dans notre modèle on notera le i-ième flux le plus prioritaire A_i. Ainsi si l'on considère deux flux A_i et A_j, si $i < j$ alors A_i est plus prioritaire que A_j.

Bit-stuffing

Maintenant que les différents trafics considérés ont été définis avec une expression de leurs courbes d'arrivée (et des flux d'arrivée lorsque c'est possible) il est possible de passer au reste de la modélisation. Dans le Chapitre 2 la méthode du "bit-stuffing" a été présentée. Pour rappel elle consiste, dans le cas où l'on a émis 5 bits de même polarité d'affilée, d'ajouter à la suite un bit de polarité contraire. Il est important de noter que seule la partie donnée est concernée par le "bit-stuffing", c'est-à-dire que l'entête n'est pas concerné par exemple. Cet ajout de bit peut alors augmenter la taille des données d'au plus 25%. Donc si la taille des données d'un paquet avant le "bit-stuffing" est de n bits, la taille maximale du paquet après le "bit-stuffing" est $l_{max} = 55 + n * 1.25$ bits.

Modèle de synchronisation

Il existe de nombreux protocoles de synchronisation, comme par exemple le *Precision Time Protocol* [Lee et al., 2005]. Quel que soit le protocole utilisé il nécessite l'émission de messages dédiés afin de partager l'horloge entre tous les noeuds. Nous avons précédemment (Section 6.1.2) proposé un mécanisme de synchronisation. Ces messages peuvent être synchrones ou asynchrones et doivent être pris en compte.

Chaque nœud possède sa propre horloge. Dans le cas d'une synchronisation faible ces différentes horloges ne sont pas égales. La différence entre elles, nommée *phase*, est inconnue mais bornée. On parle de système à phase bornée. Au contraire dans le cas d'une horloge commune la phase est nulle et dans le cas d'horloges locales non synchronisées la

phase est inconnue. On note $\phi(i,j)$ une borne supérieure sur la phase entre l'horloge du nœud i et l'horloge du nœud j.

Soit A un flux émis par le nœud i, et j un nœud différent de i. Notons t_j et t_i les dates indiquées par les horloges des nœuds i et j à un instant donné. On a donc :

$$t_i - \phi(i,j) \leq t_j \leq t_i + \phi(i,j) \tag{6.6}$$

Le flux A étant émis par le nœud i la fonction $A(t_i)$ est connue. Mais si l'on utilise l'horloge du flux j comme référence, alors $A(t_j)$ est inconnu mais peut être borné :

$$A \oslash \delta_{\phi(i,j)}(t_i) \leq A(t_j) \leq A * \delta_{\phi(i,j)}(t_i) \tag{6.7}$$

Modèle d'erreur

Enfin il est important d'être capable de modéliser les erreurs dans notre modèle. Le bus CAN possède un mécanisme de détection des erreurs très efficace. Un message d'erreur (*error flag*) peut être transmis par chaque nœud qui détecte une erreur. Ce message est constitué de six bits dominants consécutifs et il viole la règle du "bit-suffing". Après avoir reçu ce message d'erreur le nœud qui était en train d'émettre le message incorrect arrête sa transmission et le message re-rentrera dans la phase d'arbitrage avant d'être ré-émis. Les erreurs sont par nature un phénomène aléatoire qui ne peut pas être prévu. Cependant Tindell et Burns dans [Tindell and Burns, 1994] ont introduit l'idée que le nombre d'erreurs sur une période de temps peut raisonnablement être borné. Cette borne supérieure est caractérisée par :

— N_{error} le nombre maximum d'erreurs qui peut arriver de façon consécutive,

— T_{error} la période des erreurs résiduelles.

Le nombre d'erreurs de transmission durant un durée d est alors borné par : $N_{error} + \left\lceil \frac{d}{T_{error}} \right\rceil - 1$.

Maintenant que nous avons défini le modèle que nous considérons dans cette étude il est possible de passer au calcul des bornes du temps de traversée.

6.2.2 Calcul d'une borne supérieure sur le temps de traversée dans le cas des messages synchrones

Dans la section précédente nous avons défini l'ensemble des hypothèses du système étudié. Nous allons maintenant établir un modèle en calcul réseau ainsi que sa preuve afin de borner le temps de traversée des messages. Dans un premier temps nous considérerons un système uniquement composé de flux synchrones, sans erreur et avec une synchronisation faible. On suppose que cette synchronisation existe indépendamment du réseau, *i.e.* la synchronisation n'implique pas de messages supplémentaires sur le réseau. On parle de cas idéal. Afin de borner le délai nous allons proposer trois méthodes indépendantes.

La première est directement basée sur l'expression des flux d'arrivée.

La seconde utilise les courbes d'arrivée et consiste à borner la distance horizontale entre la courbe d'arrivée et la courbe de service.

La dernière repose elle aussi sur le courbe de service mais consiste à borner la période chargée (*backlogged period*).

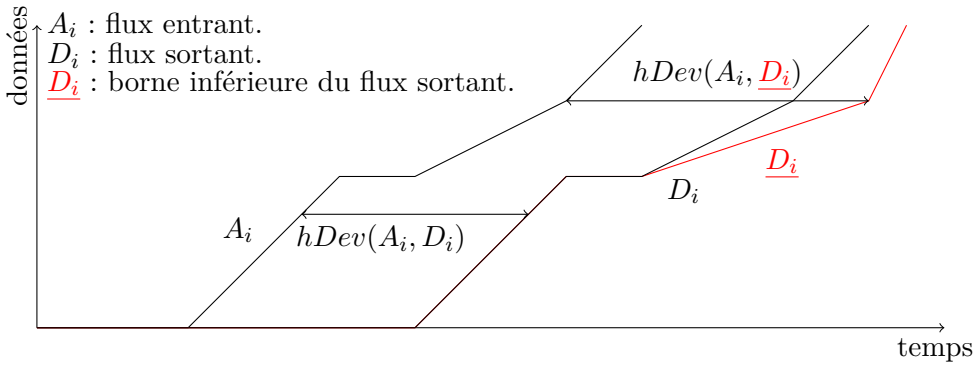


FIGURE 6.6 – Calcul du délai grâce au flux d'arrivée et de départ.

Borne basée sur le flux d'arrivée

Les bornes classiques calculées en utilisant le calcul réseau sont habituellement basées sur les courbes d'arrivée et les courbes de service. Ce n'est pas le cas de la méthode que nous présentons ici en effet nous utilisons une toute nouvelle approche du problème en utilisant non pas les courbes d'arrivée mais directement les flux d'arrivée. Dans le cas d'un système déterministe, ou du moins en partie déterministe les flux de données sont connus. Se baser sur le flux réel plutôt que sur les courbes d'arrivée peut donc permettre de meilleurs résultats : c'est ce que nous allons faire dans cette partie en nous basant sur l'équation 4.6, où le délai est exprimé comme la distance horizontale entre le flux d'arrivée et le flux de départ. Le flux d'arrivée étant connu, si l'on arrive à estimer une courbe inférieure au flux de départ il est possible de borner le délai, c'est ce qui est représenté à la Figure 6.6. Le flux entrant A_i est connu, si le flux sortant D_i est connu alors le délai maximal est lui aussi connu. Mais ce n'est pas le cas ici, l'idée est donc de calculer une borne inférieure du flux sortant $\underline{D_i}$ et de l'utiliser pour borner le temps de traversée.

Nous nous plaçons dans un premier temps dans le cas idéal. Nous faisons donc l'hypothèse que le système est uniquement composé de flux synchrones. La taille des paquets de ces flux n'est pas connue mais peut être bornée. De plus les offset de chaque flux sont relatifs à l'horloge du nœud émetteur et il existe une phase bornée entre ces différentes horloges.

Dans le Théorème 13 nous montrons comment borner le délai dans le cas du bus CAN si les flux sont connus.

Théorème 13. *Soit S un serveur NP-SP prenant l'ordre naturel comme ordre de priorité, offrant β comme courbe de service strict. Soit $\{A_1, ..., A_n\}$ n flux dont les courbes cumulatives d'arrivée sont connues. On note $l^{\max}_{i,NP}$ la taille maximale d'un paquet moins prioritaire que le flux i. Alors une borne supérieure du délai du flux i est :*

$$hDev\left(A_i, (\sum_{j\leq i} A_j) * [\beta - l^{\max}_{i,NP}]^+ - \sum_{j<i} A_j\right) \tag{6.8}$$

Démonstration. On note $\{D_1, ..., D_n\}$ les flux de départ associés aux flux $\{A_1, ..., A_n\}$. Soit

$i \leq n$ et soit t un instant :

$$D_i(t) = \sum_{j \leq i} D_j(t) - \sum_{j<i} D_j(t) \geq \sum_{j \leq i} D_j(t) - \sum_{j<i} A_j(t)$$

En utilisant le Théorème 8 on sait que $[\beta - l_{i,NP}^{\max}]^+$ est une courbe de service simple de $\sum_{j \leq i} A_j$, donc $\sum_{j \leq i} D_j \geq \left(\sum_{j \leq i} A_j \right) * [\beta - l_{i,NP}^{\max}]^+$.

On peut donc en déduire que :

$$D_i(t) \geq \left(\sum_{j \leq i} A_j \right) * [\beta - l_{i,NP}^{\max}]^+(t) - \sum_{j<i} A_j(t)$$

De plus, soit f, g, g', trois flux. Si $g \geq g'$ alors

$$hDev(f, g) \leq hDev(f, g')$$

Il est donc possible de borner $hDev(A_i, D_i)$. □

Ce résultat permet donc de borner le délai si les flux d'arrivée sont parfaitement connus. Or ce n'est pas le cas dans le modèle que nous considérons. Premièrement parce que l'offset de chaque flux n'est pas connu, la phase entre les différentes horloges étant supposée inconnue mais bornée. Secondement parce que la taille des paquets n'est pas connue.

Nous allons tout d'abord montrer comment prendre en compte la phase entre les horloges. Le Théorème 13 peut être adapté, nous obtenons alors la Propriété 1.

Propriété 1. *Une borne supérieure du délai du flux i est :*

$$hDev(A_i, D_i) \leq hDev \left(A_i, (\sum_{j \leq i} A_j \oslash \delta_{\phi(i,j)}) * [\beta - l_{i,NP}^{\max}]^+ - \sum_{j<i} A_j * \delta_{\phi(i,j)} \right) \tag{6.9}$$

où $\phi(i, j)$ est une borne supérieure sur la phase entre l'horloge du nœud émetteur du flux A_i et l'horloge du nœud émetteur du flux A_j.

Démonstration. Soit $i \leq n$. Notons t_j et t_i les dates indiquées par les horloges des nœuds émetteurs de A_i et A_j à un instant donné. C'est l'horloge du nœud émetteur du flux A_i qui sera utilisée comme référence. Soit $j \neq i$, le flux $A_j(t_j)$ est connu mais $A_j(t_i)$ est inconnu. $A_j(t_i)$ peut cependant être borné par : $A_j \oslash \delta_{\phi(i,j)}(t_j) \leq A_j(t_i) \leq A_j * \delta_{\phi(i,j)}(t_j)$

Il est donc possible de déduire du Théorème 8 :

$$D_i(t_i) \geq (\sum_{j \leq i} A_j) * [\beta - l_{NP}^{\max}]^+(t_i) - \sum_{j<i} A_j(t_i) \geq (\sum_{j \leq i} A_j \oslash \delta_{\phi(i,j)}) * [\beta - l_{i,NP}^{\max}]^+(t_j) - \sum_{j<i} A_j * \delta_{\phi(i,j)}(t_j)$$

Et donc de borner $hDev(A_i, D_i)$ dans le cas de l'existence de phase entre les nœuds. □

Maintenant il nous reste à prendre en compte la taille variable des paquets. Celle ci est due au fait qu'un flux est caractérisé par une taille maximale des paquets et pas une taille réelle mais elle est aussi due au protocole du *bit-stuffing*. Le Théorème 13 peut être adapté, nous obtenons alors la Propriété 2.

Propriété 2. *Soit $\overline{A_i} \geq A_i$ pour tout i, tel que $\overline{A_i} - A_i$ soit une courbe cumulative. Ce qui est bien le cas lorsque la taille des paquet est inconnue mais bornée. Le Théorème 13 peut être adapté :*

$$hDev(A_i, D_i) \leq hDev(\overline{A_i}, (\sum_{j \leq i} \overline{A_j}) * [\beta - l_{i,NP}^{\max}]^+ - \sum_{j<i} \overline{A_j}) \tag{6.10}$$

Démonstration. Soit $i \leq n$ et soit t un instant donné. Dans un premier temps nous allons considérer que A_i est connu mais que si $j < i$ alors A_j est inconnu mais peut être borné par $\overline{A_j} \geq A_j$

$$\begin{aligned}
(A_i + \sum_{j<i} \overline{A_j}) * [\beta - l_{i,NP}^{\max}]^+(t) - \sum_{j<i} \overline{A_j}(t) &= (A_i + \sum_{j<i} (\overline{A_j} - A_j + A_j)) * [\beta - l_{i,NP}^{\max}]^+(t) \\
&\quad - \sum_{j<i} \overline{A_j}(t) \\
&\leq \inf_{0 \leq s \leq t} \Bigg(\Big(A_i + \sum_{j<i} (\overline{A_j} - A_j + A_j) \Big)(t-s) \\
&\quad + [\beta - l_{i,NP}^{\max}]^+(s) \Bigg) - \sum_{j<i} \overline{A_j}(t) \\
&\leq \inf_{0 \leq s \leq t} \Bigg(\Big(\sum_{j \leq i} A_j \Big)(t-s) + \sum_{j<i} (\overline{A_j} - A_j)(t) \\
&\quad + [\beta - l_{i,NP}^{\max}]^+(s) \Bigg) - \sum_{j<i} \overline{A_j}(t) \\
&\leq (\sum_{j \leq i} A_j) * [\beta - l_{i,NP}^{\max}]^+(t) + \sum_{j<i} (\overline{A_j} - A_j)(t) \\
&\quad - \sum_{j<i} \overline{A_j}(t) \\
&\leq (\sum_{j \leq i} A_j) * [\beta - l_{i,NP}^{\max}]^+(t) - \sum_{j<i} A_j(t) \\
&\leq D_i(t)
\end{aligned}$$

Ce résultat permet de dire en appliquant le Théorème 13 que :

$$hDev(\overline{A_i}, \overline{D_i}) \leq hDev(\overline{A_i}, (\overline{A_i} + \sum_{j<i} \overline{A_j}) * [\beta - l_{NP}^{\max}]^+ - \sum_{j<i} \overline{A_j})$$

Il reste encore à montrer que $hDev(A_i, D_i) \leq hDev(\overline{A_i}, \overline{D_i})$. Pour cela nous allons décomposer $\overline{A_i}$ et $\overline{D_i}$: $\overline{A_i} = A_i + \widetilde{A_i}$ et $\overline{D_i} = D_i + \widetilde{D_i}$. A_i et $\widetilde{A_i}$ ont la même priorité, pour les flux de même priorité c'est la politique FIFO qui s'applique alors. Dans le cas FIFO un résultat présent dans [Le Boudec and Thiran, 2001] nous indique que $hDev(A_i, D_i) \leq hDev(A_i + \widetilde{A_i}, D_i + \widetilde{D_i})$. D'où le résultat. □

Il est bien sûr possible d'utiliser la Propriété 1 et la Propriété 2 simultanément. Nous avons montré comment, dans la cas d'un réseau uniquement parcouru par des flux synchrones, dont la taille des paquets est inconnue et où la phase entre les horloges des

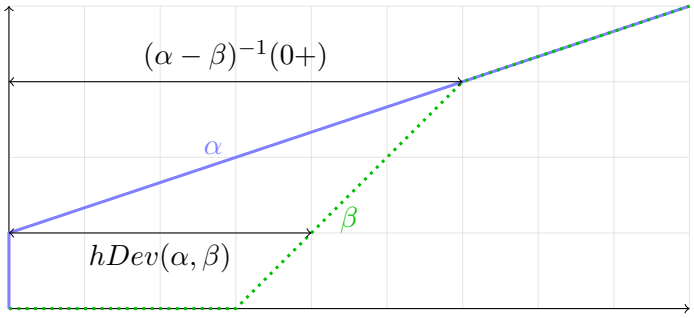


FIGURE 6.7 – Comparaison entre la taille de la période chargée et la distance horizontale.

différents nœuds est inconnue mais bornée, il est possible de borner le temps de traversée du réseau.

Nous allons maintenant montrer deux façons différentes de calculer des bornes garanties sur le temps de traversée du réseau en utilisant les courbes d'arrivée.

Bornes basées sur les courbes d'arrivée

Se baser sur le flux d'arrivée et sur une estimation du flux de départ n'est pas toujours la meilleure solution. En effet le fait que nous considérions un déphasage inconnu, mais borné, entre les nœuds même s'il peut être en partie pris en compte (Propriété 1) implique que le flux d'arrivée n'est pas connu et doit lui aussi être estimé. Dans certains cas, utiliser les courbes d'arrivée plutôt que le flux d'arrivée peut s'avérer une meilleure solution. En utilisant les courbes d'arrivée nous allons utiliser deux résultats du Calcul réseau. Tout d'abord le Théorème 3, qui permet de borner le délai en calculant la distance horizontale entre une courbe d'arrivée et une courbe de service. Ensuite nous calculerons la taille maximale d'une période chargée à l'aide du Théorème 4.

Le tout en utilisant le service résiduel donné par le Théorème 8.

Théorème 14. *Soit S un serveur offrant une courbe de service strict β, traversé par trois flux, A, A_P, A_{NP}, A_P étant plus prioritaire que A, et A_{NP} l'étant moins. Si α_P est une courbe d'arrivée de A_P, α_A une courbe d'arrivée de A, α_{A+P} une courbe d'arrivée de $A_P + A$, $L_{NP}^{\max}$ est une borne supérieure de la taille des paquets de A_{NP} et si pour finir $L_{A+NP}^{\max}$ est une borne supérieurs de la taille des paquets de $A + A_{NP}$.*

Alors il est possible de borner le délai d du flux A :

$$d \leq hDev(\alpha_A, [\beta - \alpha_P - L_{NP}^{\max}]_{\uparrow}^{+}) \tag{6.11}$$

$$d \leq (\alpha_{A+P} - [\beta - L_{A+NP}^{\max}]_{\uparrow}^{+})^{-1}(0+) \tag{6.12}$$

Dans le cas du bus CAN, si l'on considère n flux $A_1..A_n$, et que le flux qui nous intéresse est le flux de priorité i : A_i, alors $A_P = \sum_{k=1}^{i-1} A_k$ et $A_{NP} = \sum_{k=i+1}^{n} A_k$.

Il est important de comprendre pourquoi nous avons décidé d'utiliser à la fois la taille de la période chargée et la distance horizontale pour borner le temps de traversée. Pour cela nous les avons représentées sur un exemple dans la Figure 6.7. Ces deux valeurs

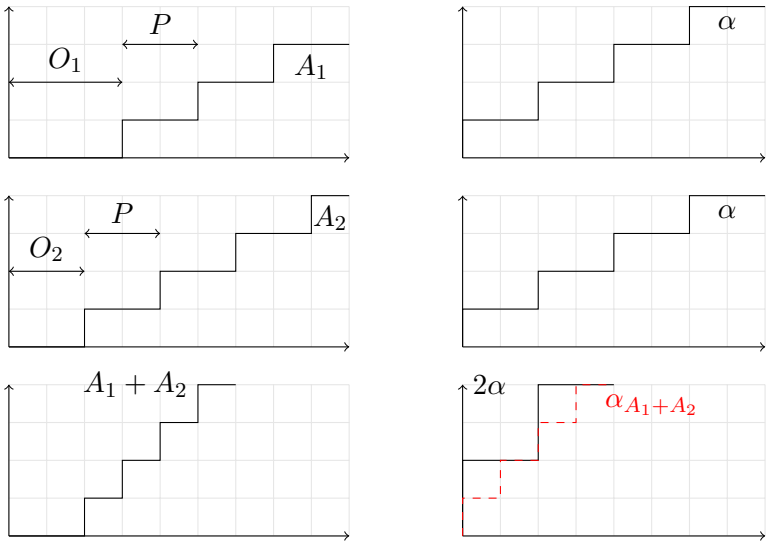


FIGURE 6.8 – Modélisation de la synchronisation par les courbes d'arrivée.

sont bien des bornes sur le temps de traversée d'un message et comme illustré sur la Figure 6.7 dans le cas général la distance horizontale est toujours inférieure à la taille de la période chargée. C'est donc la distance horizontale qui est utilisée de façon classique dans le calcul réseau. Mais nous sommes ici dans un cas particulier, en effet, contrairement à la distance horizontale qui considère α_A et α_P, la taille de la période chargée utilise α_{A+P}. Ce qui signifie que la taille de la période chargée est capable de prendre en compte la synchronisation qui existe entre le flux A et le flux A_P ce que la distance horizontale ne peut pas faire. Ainsi en calculant la taille de la période chargée dans cette situation elle sera parfois meilleure que la distance horizontale d'où l'importance d'utiliser les deux méthodes.

La première chose à faire et d'être capable de *capturer* la synchronisation dans l'expression de nos courbes d'arrivée, c'est-à-dire calculer finement α_P et α_{A+P}. Ce problème est illustré par la Figure 6.8. Si l'on considère deux flux périodiques A_1 et A_2 parfaitement connus, avec la même période P et des offsets O_1 et O_2 tels que représentés sur la figure, leurs courbes d'arrivée sont égales : α. Maintenant si l'on considère la somme de ces deux flux, on obtient un nouveau flux $A_1 + A_2$. La question est d'évaluer la courbe d'arrivée de ce nouveau flux. Un solution évidente consiste à considérer la somme des courbes d'arrivée de A_1 et A_2, ici 2α. Cette courbe est bien une courbe d'arrivée mais comme illustré sur le figure elle ne prend pas en compte la synchronisation, on a donc perdu de l'information. La courbe qui nous intéresse est la courbe rouge que nous avons notée $\alpha_{A_1+A_2}$. C'est cette courbe qu'il nous faut être capable de calculer si l'on veut pouvoir considérer la synchronisation qui existe entre les nœuds.

Pour la suite nous allons considérer $A_1, .., A_n$ un ensemble de flux synchrones et pour tout $i \leq n$, on note $\alpha_{1..i}$ la courbe d'arrivée de la somme des i premier flux : $\sum_{j=1}^{i} A_j$. Nous avons vu précédemment que $\sum_{j=1}^{i} \alpha_j$ est une courbe d'arrivée de ce flux, mais lorsque nous nous référons à $\alpha_{1..i}$ nous cherchons à prendre en compte les offsets et la phase entre les nœuds.

Pour cela nous allons d'abord faire une simplification en considérant que tous les flux

ont la même période. Il est possible de se ramener à cette hypothèse en utilisant l'hyper-période du système et des offsets. Cette période globale commune permet de décomposer chaque flux en une somme de sous flux ayant cette même période et des offset adaptés.

Lemme 1. *(Période commune) Soit A un flux périodique tel que $A = \nu_{P_A,l} * \delta_O$ et P l'hyper-période du système. Il existe donc $n_A \in \mathbb{N}$ tel que $P = n_A P_A$. Maintenant A peut être exprimé comme : $A = \sum_{i=1}^{n_A} \nu_{P,l} * \delta_{O+(i-1)P_A} = \sum_{i=1}^{n_A} A_i$. Ce qui définit bien notre fonctions comme une somme de fonctions P-périodiques. En appliquant ce procédé à tous les flux il est possible de décomposer chaque chaque flux en une somme de sous flux ayant la même période.*

Nous allons tout d'abord considérer que la synchronisation est parfaite, toutes les horloges locales sont supposées égales. La modélisation des différences de phases entre elle sera faite plus tard.

Nous avons vu que nous pouvons considérer une unique période P commune à tous les flux (Lemme 1). La synchronisation étant supposée parfaite, pour chaque flux M son offset O_M est connu. Nous allons considérer que $O_M \leq P$, c'est à dire que tous les flux émettent dès la première période. L'expression du flux d'arrivée est donc :

$$M(t) = \left[l_M \left\lceil \frac{t - O_M}{P} \right\rceil \right]^+ \tag{6.13}$$

Dans notre cas nous considérons la somme des i-premiers flux de notre ensemble : $\sum_{j=1}^{i} A_j$. Cette fonction est une *fonction en escalier* c'est-à-dire qu'elle est constante par morceau, sur une période elle ne change de valeur qu'un nombre fini de fois, au maximum i fois, à l'émission de chaque nouveau message. Pour prendre en compte ce phénomène nous avons besoin de définir la fonction $next(t)$:

Définition 25. *Soit $O_1, \ldots, O_n \in [0, P[$ tels que $0 \leq O_1 \leq \ldots \leq O_n < P$ et $O_{n+1} = O_1 + P$. On définit alors pour tout $t \in [0, P[$: $next(t) = \min_{1 \leq i \leq n+1} \{i | O_i \geq t\}$. $next(t)$ est le premier flux qui émet un message après la date t.*

La propriété suivante permet d'exprimer le fait que pour tout instant t, chaque flux restera constant au minimum jusqu'à la date $next(t)$:

Propriété 3. *Soient $A_1, \ldots, A_n$ un ensemble de flux, tels que $A_i = \left[l_i \left\lceil \frac{t-O_i}{P} \right\rceil \right]^+$ avec $\forall (i,j) \in \{1,n\}$, $i \leq j \Rightarrow 0 \leq O_i \leq O_j < P$. On définit $O_{n+1} = O_1 + P$. On a alors :*

$$\forall j \leq n, \forall t \in [0, P[, A_j(t) = A_j(O_{next(t)}) \tag{6.14}$$

Démonstration. Soit $t \in [0, P[$,

$$\begin{cases} j < next(t) \Rightarrow O_j + P \geq O_{next} \geq t > O_j \\ j \geq next(t) \Rightarrow t \leq O_{next(t)} \leq O_j \end{cases}$$
$$\Rightarrow \begin{cases} A_j(t) = A_j(O_j + P) = A(O_{next(t)}) \\ A_j(t) = A_j(O_j) = A(O_{next(t)}) \end{cases}$$

□

Maintenant que nous avons cette propriété il est possible d'en déduire la courbe d'arrivée qui nous intéresse $\alpha_{1..n}$:

Théorème 15. *(Courbe d'arrivée d'une somme de flux périodiques) Soient $A_1, \ldots, A_n$ un ensemble de flux périodiques, tels que $A_i = \left[l_i \left\lceil \frac{t-O_i}{P} \right\rceil\right]^+$ avec $\forall (i,j) \in \{1,n\}$, $i \leq j \Rightarrow 0 \leq O_i \leq O_j < P$. On définit $O_{n+1} = O_1 + P$. Alors $\alpha_{1..n}$ est une courbe d'arrivée de la somme des flux $A = \sum_{1 \leq i \leq n} A_j$:*

$$\alpha_{1..n}(d) = \max_{1 \leq i \leq n} (A(O_i + d) - A(O_i)). \tag{6.15}$$

De plus $\alpha_{1..n}$ est la meilleure courbe d'arrivée de $A = \sum_{1 \leq i \leq n} A_j$.

Démonstration. Soit $t, d \in \mathbb{R}^+$ tel que $t = kP + t_0$ avec $0 \leq t_0 < P$ et $k \in \mathbb{N}$.

$$\begin{aligned} A(t+d) - A(t) &= A(t_0 + d) - A(t_0) \\ &= A(t_0 + d) - A(O_{next(t_0)}) \\ &\leq A(O_{next(t_0)} + d) - A(O_{next(t_0)}) \\ &\leq \max_{1 \leq i \leq n} (A(O_i + d) - A(O_i)) = \alpha_{1..n}(d) \end{aligned}$$

Donc $\max_i (A(O_i + d) - A(O_i))$ est bien une courbe d'arrivée de $A = \sum_{1 \leq i \leq n} A_j$. De plus c'est bien sa meilleure courbe d'arrivée possible. On peut le prouver rapidement par l'absurde : soit α_{min} la meilleure courbe d'arrivée avec $\alpha_{min} \neq \alpha_{1..n}$. Alors il existe d et i tels que $\alpha_{min}(d) < \alpha_{1..n}(d) = A(O_i + d) - A(O_i)$. Donc α_{min} n'est pas une courbe d'arrivée ce qui n'est pas possible. Donc $\alpha_{1..n}$ est bien la meilleure courbe d'arrivée. □

Il est possible de donner une expression de $\alpha_{1..n}$ équivalente :

$$\alpha_{1..n}(d) = \max_{1 \leq i \leq n} \left(\sum_{j=1}^{n} l_j \left\lceil \frac{t - dist(i,j)}{P} \right\rceil \right) \tag{6.16}$$

Où $dist(i,j)$ représente la distance orientée entre le flux A_i et le flux A_j :

$$dist(i,j) = O_j - O_i \mod P \tag{6.17}$$

L'équation 6.16 permet de calculer α_P et α_{A+P} dans le Théorème 14. Cependant notre expression ne prend pour l'instant pas en compte la différence de phase qui peut exister entre les nœuds. Pour la prendre en compte il nous faut modifier notre définition de $dist(i,j)$:

Théorème 16. *Soit A_i et A_j deux flux synchrones d'offset respectif O_i et O_j. On note $\phi(i,j)$ la borne supérieure sur la phase entre l'horloge du nœud émetteur de A_i et l'horloge du nœud émetteur de A_j. La distance orientée $dist(i,j)$ entre le flux A_i et le flux A_j peut être bornée inférieurement par :*

$$dist(i,j) \geq \min \left(\max \Big(0, dist_0(i,j) - \phi(i,j)\Big), \min \Big(P, dist_0(i,j) + \phi(i,j)\Big) \mod P \right) \tag{6.18}$$

Où $dist_0(i,j)$ est la distance orientée entre le flux A_i et le flux A_j lorsque la phase est nulle : $dist_0(i,j) \equiv O_j - O_i \mod P$.

Démonstration. Lorsque la phase est non nulle mais peut être bornée par $\pm\phi(i,j)$ nous savons que :

$$dist_0(i,j) - \phi(i,j) \leq dist(i,j) \leq dist_0(i,j) + \phi(i,j)$$

De plus A_i et A_j étant périodiques de période P, nous savons que :

$$0 \leq dist(i,j) < P$$

Nous pouvons donc en déduire :

$$\max\Big(0, dist_0(i,j) - \phi(i,j)\Big) \leq dist(i,j) \leq \min\Big(P, dist_0(i,j) + \phi(i,j)\Big)$$

Et donc nous pouvons en conclure que :

$$dist(i,j) \geq \min\left(\max\Big(0, dist_0(i,j) - \phi(i,j)\Big), \min\Big(P, dist_0(i,j) + \phi(i,j)\Big) \mod P\right)$$

□

En utilisant cette borne inférieure nous sommes donc maintenant capables de calculer α_P et α_{A+P} dans le Théorème 14 lorsque tous les messages sont synchrones dans le cas d'une synchronisation faible.

6.2.3 Calcul d'une borne supérieure sur le temps de traversée dans le cas général

Dans la partie précédente nous nous sommes placés dans le cas idéal où un certain nombre d'hypothèses avaient été faites. Tout d'abord nous avons considéré que tout le trafic était synchrone, or en pratique il est parfois nécessaire de recourir à un trafic asynchrone, pour les alarmes notamment. De plus nous avons présenté un modèle d'erreurs dans la Section 6.2.1 que nous n'avons pas pris en compte dans le calcul des bornes. Enfin toute synchronisation, même faible, nécessite un mécanisme de synchronisation, cette dernière impliquera l'émission de messages sur le bus qui perturberont le reste du trafic et doit donc être prise en compte.

Prise en compte du trafic asynchrone

Dans la section précédente nous avons proposé trois façons de borner le délai dans le cas où le réseau n'est parcouru que par des flux synchrones, les résultats sont détaillés dans les Théorèmes 13 et 14.

Le premier (Théorème 13) se base sur les flux d'arrivée, ces derniers n'étant pas connus dans le cas du trafic asynchrone il ne peut pas être utilisé. Il aurait pu être adapté en utilisant des bornes sur les flux d'arrivée à l'aide des courbes d'arrivée maximale et minimale mais la perte d'information que cela aurait impliqué aurait rendu le résultat inutilisable en pratique.

Le Théorème 14 peut directement être utilisé, en effet il ne suppose pas que le trafic soit entièrement synchrone. Cependant le calcul de $\alpha_{1..n}$ dans le Théorème 15 n'est lui plus applicable mais il est possible de l'adapter :

Corolaire 2. *Soit $A_1, .., A_n$ un ensemble de flux et α_n une courbe d'arrivée de A_n. Si $\alpha_{1..n-1}$ est une courbe d'arrivée de $A_{1..n-1} = \sum_{1 \leq i \leq n-1} A_i$, alors une courbe d'arrivée de $A = \sum_{1 \leq i \leq n} A_i$ est :*

$$\alpha_{1..n} = \alpha_{1..n-1} + \alpha_n \tag{6.19}$$

Il est donc possible de calculer une courbe d'arrivée en considérant d'un coté les flux synchrones (Théorème 15, courbe $\alpha_{1..n-1}$), d'un autre coté les flux asynchrones (Propriété 2, courbe α_n) et en faisant la somme.

Prise en compte du mécanisme de synchronisation

Comme cela a été dit plus tôt, tout mécanisme de synchronisation nécessite l'émission de messages non seulement pour établir la synchronisation mais aussi pour la maintenir. Quel que soit le mécanisme utilisé, il faut être capable d'évaluer l'impact de ces messages.

Dans le cadre de l'évaluation nous avons considéré que, quel que soit le mécanisme utilisé, il nécessitait au plus l'émission d'un message par hyper-période. De plus nous avons décidé de modéliser ces messages comme un flux asynchrone de priorité maximale.

Pour résumer le mécanisme de synchronisation est pris en compte en supposant l'existence d'un flux asynchrone de priorité maximale qui émet au plus un message par hyper-période. Il est possible de le prendre en compte grâce à la Propriété 2 comme tous les flux asynchrones.

Modélisation du flux d'erreurs

Nous avons déjà proposé à la Section 6.2.1 un modèle permettant de prendre en compte l'existence d'erreurs sur le bus. Pour rappel le nombre d'erreurs durant une durée d peut être borné par $N_{error} + \left\lceil \frac{d}{T_{error}} \right\rceil - 1$. Chaque erreur représente non seulement un message incorrect de taille maximale L_{max} mais aussi un message d'erreurs de taille L_{erreur}. Nous connaissons donc le nombre d'erreurs et la taille maximale des messages transmis à chaque erreur, il nous est alors possible de modéliser ces erreurs comme un flux asynchrone prioritaire supplémentaire de courbe d'arrivée :

$$\alpha_{erreur} = (L_{max} + L_{erreur}) \left(N_{error} + \left\lceil \frac{d}{T_{error}} \right\rceil - 1 \right) \tag{6.20}$$

Encore une fois ce flux peut être pris en compte grâce à la Propriété 2.

Remarque : Il pourrait sembler intéressant de se demander quelle est la priorité du flux d'erreurs par rapport à celle du flux de synchronisation à ce point. Du point de vue du modèle mathématiques cela n'a pas d'importance, en effet le délai de ces deux flux ne sera pas calculé, car cela n'a dans notre cas pas d'intérêt voire pas de sens dans le cas du flux d'erreurs. Et pour tous les autres flux, la seule information qui importe est que ces flux soient plus prioritaires, l'ordre de priorité entre ces deux flux n'a pas d'importance. D'un

point de vue plus général, il faudrait considérer que le flux d'erreurs est le plus prioritaire. En effet les messages de synchronisation peuvent eux même être incorrects et subiront donc un délai supplémentaire dû aux erreurs de transmission. Si le nombre d'erreurs sur le bus était trop important il serait même possible que cela affecte la synchronisation, cependant cette problématique ne seront pas considérée ici. Nos hypothèses sont qu'il existe une synchronisation faible entre les nœuds et qu'elle est maintenue grâce à un mécanisme de synchronisation. Ce mécanisme nécessite l'émission d'au plus un message par hyper-période. La synchronisation entre les nœuds n'est pas affectée par la présence d'erreurs sur le bus.

6.3 Évaluation du gain apporté par une synchronisation faible

Maintenant que nous avons développé notre modèle pour borner le temps de traversée dans le cas d'un bus faiblement synchronisé, il nous reste à évaluer le gain que permet d'apporter une synchronisation faible. Pour cela, nous étudierons tout d'abord deux configurations particulières et comparerons les bornes sur les délais en fonction de la situation retenue, par exemple nous comparerons le cas où les horloges locales ne sont pas du tout synchronisées avec le cas où l'on a une synchronisation faible. Dans cette première partie nous regarderons aussi l'impact de la précision de la synchronisation sur le temps de traversée. Dans un second temps nous évaluerons le gain d'un point de vue de l'utilisation du bus qu'une synchronisation faible permet d'obtenir. En effet il est en général nécessaire de charger peu le bus afin de respecter les contraintes temporelles cependant si l'on est capable de garantir que des temps de traversée sont faibles alors on peut charger plus le bus et ainsi effectuer des économies.

6.3.1 Comparaison et analyse des bornes sur le temps de traversée de deux configurations

Configuration 1

Cette première configuration a pour but de comparer et d'évaluer les différentes bornes que l'on a calculées précédemment (Théorème 13 et Théorème 14). Pour cela un certain nombre d'hypothèses vont être faites pour cette première configuration. Tout d'abord nous considérerons que tous les messages sont synchrones. Ensuite nous négligerons les erreurs et nous considérerons que le bus est parfait. Enfin nous ne modéliserons pas de messages de synchronisation pour cette configuration et ferons l'hypothèse que la synchronisation entre les nœuds, lorsqu'elle existe est établie indépendamment du bus CAN.

Maintenant que les hypothèses sont faites, il nous faut définir les caractéristiques de cette configuration. Celle-ci est un réseau CAN avec un débit de 250kbit/s et constitué de 10 nœuds. Les périodes des messages sont choisies uniformément dans l'ensemble $\{20, 50, 100, 200, 500, 1000\}$ms et la taille des messages est comprise entre 1 et 8 octets. La Figure 6.9 montre la distribution des tailles et périodes des messages de cette configuration. Chaque cercle représente le nombre de flux partageant la même valeur de période et taille des messages Par exemple il y a 4 flux ayant une période de 1000ms et une taille des messages de 6 octets. La charge totale sur le bus est de 35%, et il n'y a pas de différence entre

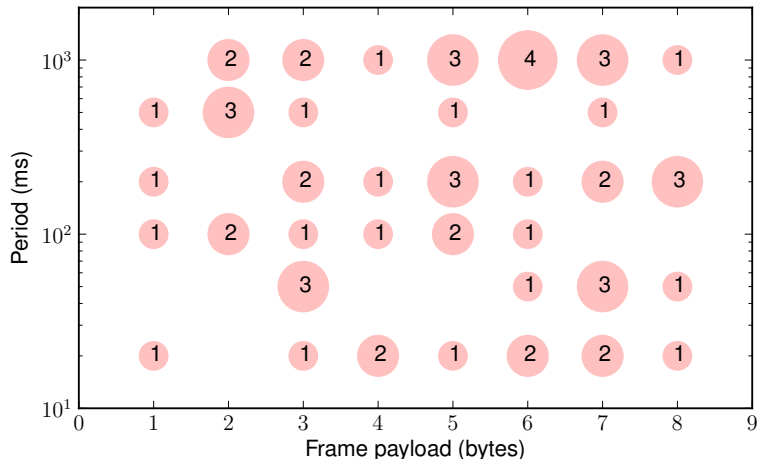


FIGURE 6.9 – Distribution de la taille et de la période des messages de la configuration 1.

les nœuds, chacun d'entre eux génère une quantité similaire de trafic. Le choix des offsets a été fait en utilisant l'algorithme SOPA disponible dans l'outil RTaW-Pegase de l'entreprise RTaW, dont les résultats ont servi de référence pour valider l'analyse développée précédemment et le gain obtenu avec une synchronisation faible. L'algorithme SOPA a été choisi car d'expérience il surpasse les quelques autres algorithmes de choix d'offsets disponibles (voir [Grenier et al., 2008, Grenier et al., 2006]) et donc nous permet d'avoir l'estimation du meilleur gain envisageable grâce à l'utilisation d'offset.

Dans le but d'évaluer le gain apporté par l'utilisation d'un bus faiblement synchronisé nous comparons quatre situations. Le premier cas est celui où il existe une synchronisation parfaite entre les horloges des différents nœuds. Chaque nœud possède la même horloge commune.

Le deuxième cas est celui où il existe une synchronisation entre les horloges des différents nœuds, mais cette synchronisation n'est pas parfaite. Il existe une phase bornée entre les horloges des différents nœuds.

Le troisième cas est celui où il n'existe pas de synchronisation entre les nœuds, chaque horloge est locale et indépendante des autres.

Le dernier cas est celui où les offset ne sont pas utilisés. Les flux sont périodiques (ou sporadiques) mais les offsets sont tous inconnus.

Pour évaluer une borne sur le délai dans les différentes situations nous utilisons plusieurs méthodes :

- Bus faiblement synchronisé ou phase bornée :
  - Méthode 1 (**M.1**) : Utilisation du Théorème 14 équation (6.11), et calcul des courbes d'arrivée (Théorème 15).
  - Méthode 2 (**M.2**) : Utilisation du Théorème 14 équation (6.12), et calcul des courbes d'arrivée (Théorème 15).
  - Méthode 3 (**M.3**) : Utilisation des flux d'arrivée, Théorème 13.
- Horloges locales :
  - Method 4 (**M.4**) : Utilisation de l'algorithme publié dans [Yomsi et al., 2012] disponible dans RTaW-Pegase. Cet algorithme est exact à une trame prêt (sur-

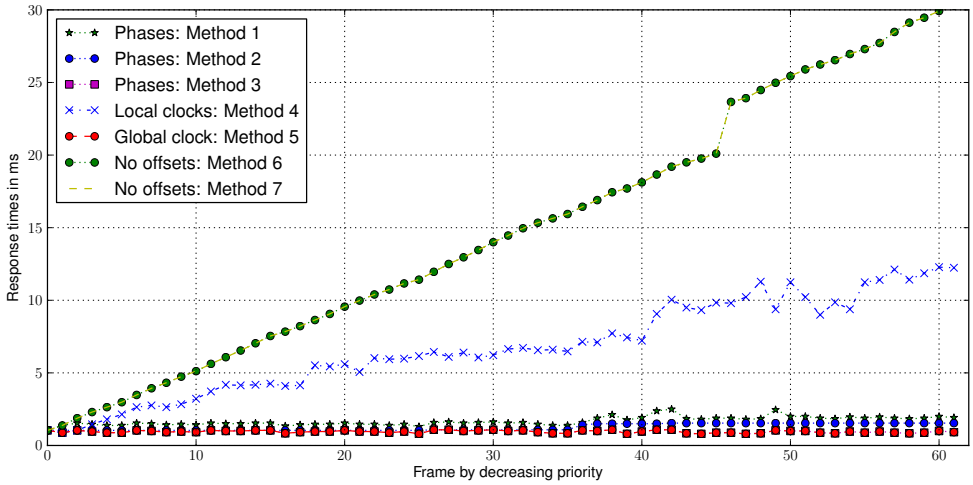


FIGURE 6.10 – Bornes sur le délai lorsque la phase est nulle.

estimation du facteur de blocage de la trame CAN) et il prend en compte un offset local (sans considérer la phase entre les horloges).

— Horloge commune ou globale :

  — Méthode 5 (**M.5**) : Du point de vue du réseau un unique nœud émetteur de tous les flux où plusieurs nœuds parfaitement synchronisés produisent le même flux sur le bus. Le même algorithme que précédemment est donc utilisé mais en supposant un unique émetteur.

— Sans offset :

  — Méthode 6 (**M.6**) : Calcul du temps exact de traversée présenté dans [Davis et al., 2007], sans considérer d'offset.

  — Méthode 7 (**M.7**) : Le calcul réseau sans offset est utilisé ($\alpha_{1..i-1} = \sum_{k=1}^{i-1} \alpha_k$) avec l'équation (6.11).

Maintenant que l'on a les différentes méthodes nous allons comparer les résultats que l'on obtient pour différentes valeurs de phases entre les horloges.

Horloge commune (phases nulles) Commençons par la configuration d'un bus parfaitement synchronisé (phases entre les horloges nulles) où tous les nœuds utilisent la même horloge commune. La Figure 6.10 détaille le délai obtenu pour chaque message et pour chaque méthode. Cette configuration permet d'évaluer dans quelle mesure les méthodes développées pour les phases introduisent un pessimisme dans le calcul de la borne sur les délais d'un message.

Tout d'abord on observe que lorsque la phase entre les horloges est nulle la méthode (**M.3**) est aussi bonne que (**M.5**). Les résultats avec (**M.1**)(**M.2**) sont légèrement plus grands qu'avec (**M.5**) mais restent proches : la différence est en moyenne inférieure à 0.5ms, ce qui représente le temps de transmission d'un message d'une taille de 8 octets à 250kbit/s.

TABLE 6.1 – Bornes sur le délai lorsque les phases sont nulles.

Prioritées	**2-16**	**17-31**	**32-46**	**47-61**	**2-61**
Délai moyen (en ms)					
Phases : **M.1**	1.47	1.49	1.82	1.94	1.68
Phases : **M.2**	1.04	1.05	1.40	1.54	1.25
Phases : **M.3**	0.97	0.98	0.94	0.91	0.95
Horloges locales : **M.4**	3.00	5.84	8.04	10.88	6.94
Horloges communes : **M.5**	0.97	0.98	0.94	0.91	0.95
Pas d'offsets : **M.6.7**	4.80	11.23	17.96	27.11	15.27
Délai maximum (en ms)					
Phases : **M.1**	1.54	1.62	2.50	2.46	2.50
Phases : **M.2**	1.04	1.08	1.54	1.54	1.54
Phases : **M.3**	1.04	1.08	1.08	1.04	1.08
Horloges locales : **M.4**	4.26	6.64	10.04	12.28	12.28
Horloges communes : **M.5**	1.04	1.08	1.08	1.04	1.08
Pas d'offsets : **M.6.7**	7.84	14.46	23.66	30.30	30.30

Cette expérience met en évidence le gain apporté par l'utilisation d'une horloge commune (méthodes **M.1,2,3,5**) en comparaison avec des horloges locales (**M.4**). Comme attendu de résultats connus dans la littérature [Grenier et al., 2008] l'utilisation d'offsets permet des délais bien moins importants comme on peut le voir avec les méthodes (**M.6**) et (**M.7**). Sans offset, le délai maximum augmente de façon presque linéaire. La discontinuité à 20ms peut simplement s'expliquer, il s'agit de la plus petite période d'émission, ainsi à partir de ce point les messages peuvent être retardés par plus d'un message par flux de priorité supérieure.

Enfin on remarque que la méthode du calcul réseau sans offset (**M.7**) donne les mêmes résultats que l'analyse exacte de [Davis et al., 2007] (**M.6**). Dans la suite ces deux méthodes ne seront plus distinguées et seront désignées par (**M.6,7**).

Phases bornées par au plus 1ms Dans l'expérience suivante, la configuration du système reste la même, la seule différence est que nous supposons que maintenant nos phases sont inconnues mais bornées par plus ou moins 1ms. Les résultats de cette configuration sont présentés dans la Figure 6.11. Ce choix de phases a été fait car l'on a montré précédemment (Section 6.1.2) que c'est un précision très simplement accessible. Dans cette configuration, l'analyse de l'horloge commune (**M.5**) ne peut plus être utilisée.

Considérons d'abord (**M.3**) : pour les messages de haute priorité (0-35), c'est toujours la méthode qui fournit les bornes minimales, cependant pour les flux de priorité inférieure, cette méthode renvoie un résultat très variable. Parfois, c'est le meilleur que nous ayons, mais parfois c'est même pire que lorsque les offsets ne sont pas utilisés (**M.6,7**).

Ensuite considérons (**M.1**) et (**M.2**) : cette imprécision de la valeur de la phase de plus ou moins 1ms a augmenté le délai d'en moyenne de 0.5ms par rapport au cas précédent avec une horloge globale. Il est important de noter que ces bornes sont toujours bien inférieures à celles de (**M.4**), qui est la valeur exacte à considérer lorsque il n'y a pas de

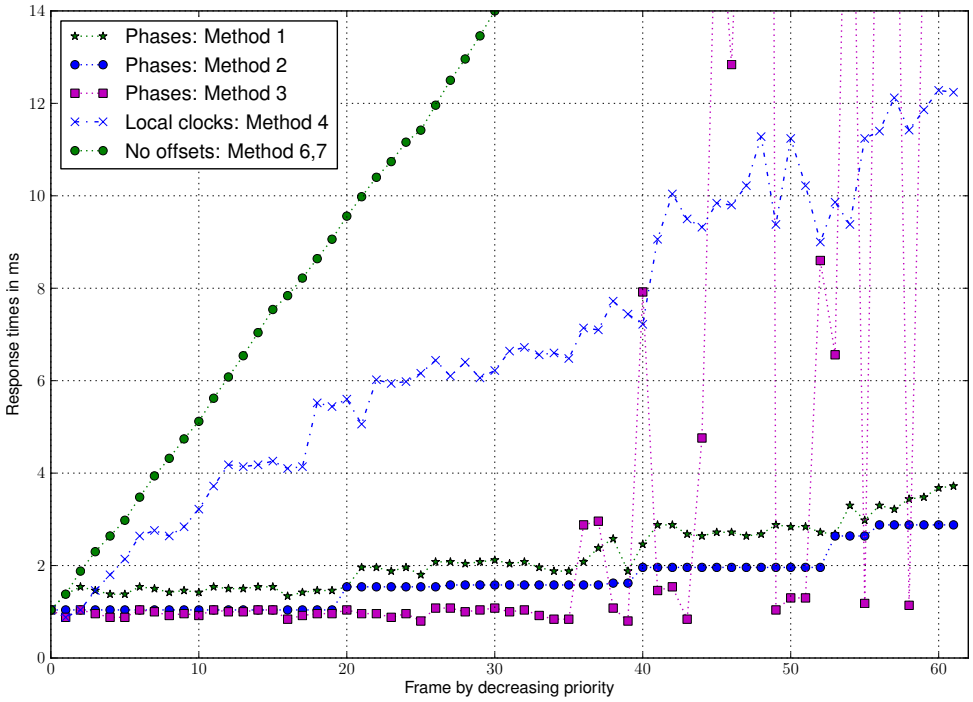


FIGURE 6.11 – Bornes sur le délai lorsque la phase est bornée par 1ms.

synchronisation entre nœuds.

Phases bornées par au plus 5ms Dans l'expérience suivante nous augmentons significativement l'imprécision sur les phases jusqu'à 5ms (voir la Figure 6.12). Considérons d'abord (**M.3**) : l'observation précédente est confirmée, pour les messages de priorité très haute (0-12), c'est toujours la méthode qui fournit la borne la plus précise, mais pour les trames de priorité inférieure, les limites calculées sont inutilisables.

Considérons ensuite (**M.1**) et (**M.2**) : par rapport aux bornes données par (**M.4**), le calcul en considérant des phases bornées permet une réduction des bornes de près de 40 %, ce qui est très important pour de nombreuses applications temps réel distribuées.

Phases bornées par au plus 10ms Dans la dernière expérience utilisant cette configuration, les phases sont maintenant bornées par plus ou moins 10ms (voir Figure 6.13). Pour rappel, des phases limitées par plus ou moins 10ms impliquent une fenêtre de transmission de longueur 20ms. La plus petite période d'émission de notre réseau est de 20 ms, ce qui signifie que l'utilisation d'offsets n'a pas d'influence sur les flux avec la plus petite période. Cette expérimentation n'est pas du tout représentative, mais permet de tester les limites de notre modèle.

Il est confirmé que (**M.3**) ne devrait pas être utilisé pour des phases trop grandes, même s'il donne toujours le meilleur résultat pour les 8 messages les plus prioritaires.

Cette expérience montre que même si (**M.2**) renvoie très souvent de meilleurs résultats que (**M.1**), ce n'est pas toujours le cas. De plus, si ces méthodes donnent la plupart du temps de meilleurs résultats que lorsque les horloges des différents nœuds ne sont

TABLE 6.2 – Bornes sur le délai lorsque les phases $\leq$ 1ms.

Prioritées	**2-16**	**17-31**	**32-46**	**47-61**	**2-61**
Délai moyen (en ms)					
Phases : **M.1**	1.47	1.85	2.38	3.09	2.20
Phases : **M.2**	1.04	1.45	1.76	2.46	1.68
Phases : min(**M.3**.**M.7**)	0.97	0.98	3.86	13.88	4.93
Local clocks : **M.4**	3.00	5.84	8.04	10.88	6.94
No offsets : **M.6.7**	4.80	11.23	17.96	27.11	15.28
Délai maximum (en ms)					
Phases : **M.1**	1.54	2.12	2.88	3.72	3.72
Phases : **M.2**	1.04	1.58	1.96	2.88	2.88
Phases : min(**M.3**.**M.7**)	1.04	1.08	17.3	30.3	30.3
Local clocks : **M.4**	4.26	6.64	10.04	12.28	12.28
No offsets : **M.6.7**	7.84	14.46	23.66	30.30	30.30

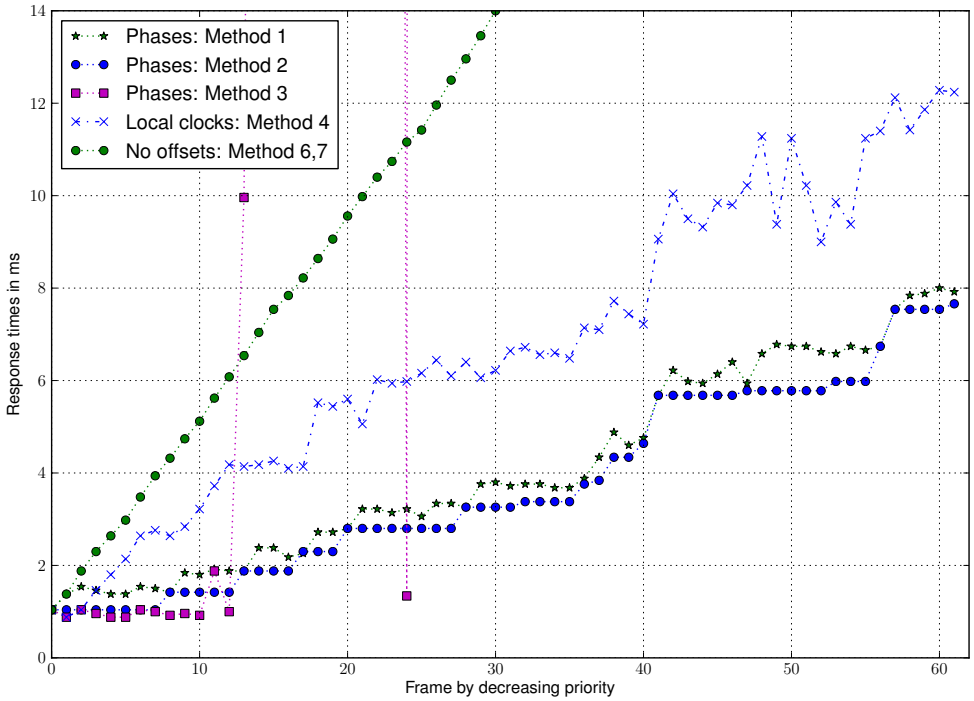


FIGURE 6.12 – Bornes sur le délai lorsque la phase est bornée par 5ms.

TABLE 6.3 – Bornes sur le délai lorsque les phases $\leq$ 5ms.

Priorités	**2-16**	**17-31**	**32-46**	**47-61**	**2-61**
Délai moyen (en ms)					
Phases : **M.1**	1.77	3.17	4.91	7.02	4.22
Phases : **M.2**	1.39	2.82	4.57	6.48	3.82
Phases : min(**M.3**.**M.7**)	2.70	10.58	17.96	27.12	14.59
Local clocks : **M.4**	3.01	5.85	8.04	10.88	6.94
No offsets : **M.6.7**	4.80	11.23	17.96	27.12	15.28
Délai maximum (en ms)					
Phases : **M.1**	2.38	3.80	6.40	8.00	8.00
Phases : **M.2**	1.88	3.26	5.68	7.66	7.66
Phases : min(**M.3**.**M.7**)	7.84	14.46	23.66	30.30	30.30
Local clocks : **M.4**	4.26	6.64	10.04	12.28	12.28
No offsets : **M.6.7**	7.84	14.46	23.66	30.30	30.30

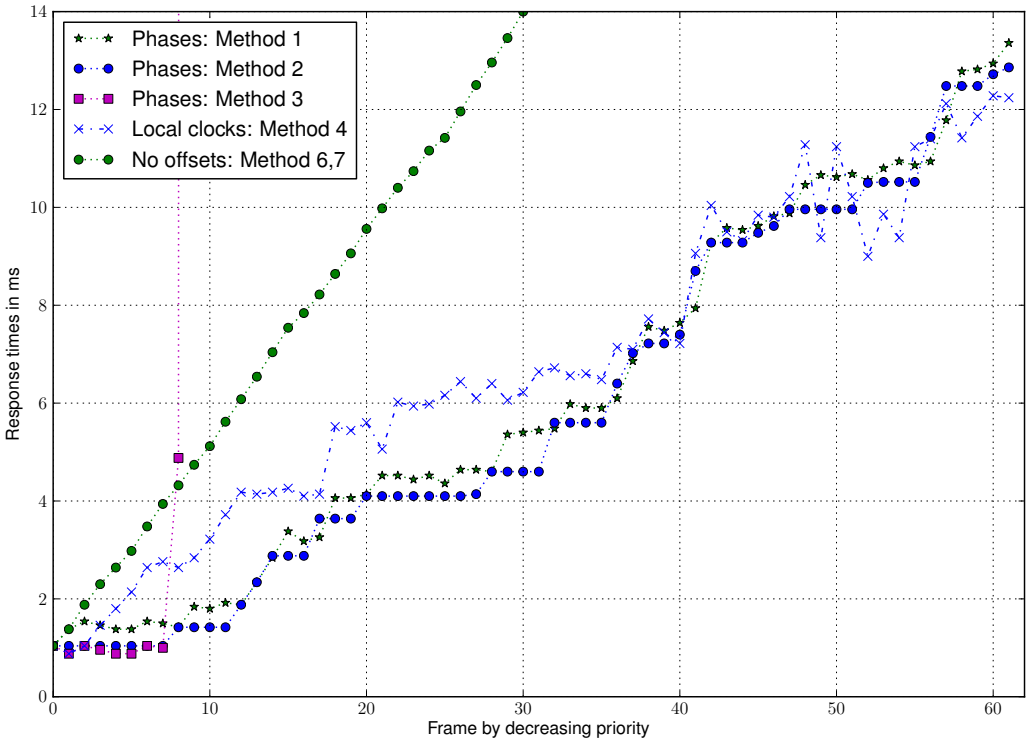


FIGURE 6.13 – Bornes sur le délai lorsque la phase est bornée par 10ms.

TABLE 6.4 – Bornes sur le délai lorsque les phases $\leq$ 10ms.

Prioritées	**2-16**	**17-31**	**32-46**	**47-61**	**2-61**
Délai moyen (en ms)					
Phases : **M.1**	1.96	4.53	7.65	11.34	6.37
Phases : **M.2**	1.65	4.14	7.55	11.09	6.11
Phases : min(**M.3**.**M.7**)	4.04	11.23	17.96	27.12	15.09
Local clocks : **M.4**	3.01	5.85	8.04	10.88	6.94
No offsets : **M.6.7**	4.80	11.23	17.96	27.12	15.28
Délai maximum (en ms)					
Phases : **M.1**	3.38	5.44	9.82	13.36	13.36
Phases : **M.2**	2.88	4.60	9.62	12.86	12.86
Phases : min(**M.3**.**M.7**)	7.84	14.46	23.66	30.30	30.30
Local clocks : **M.4**	4.26	6.64	10.04	12.28	12.28
No offsets : **M.6.7**	7.84	14.46	23.66	30.30	30.30

pas synchronisées, il arrive (en particulier pour les flux de faible priorité) que les bornes calculées en utilisant ces fonctions (**M.2** et **M.3**) soient parfois plus grandes que celles de (**M.4**).

Configuration 1 : Bilan Il a été largement démontré dans la littérature que les offsets, en réduisant les contentions sur le bus, sont un moyen efficace de diminuer le pire temps de traversée des messages dans les systèmes périodiques en temps réel. Dans les systèmes distribués, on distingue deux approches principales d'utilisation d'offset : soit utiliser une horloge commune, nécessitant un mécanisme de synchronisation très précis et coûteux soit recourir à des horloges locales et donc sans l'exigence de synchronisation entre nœuds.

Dans ce chapitre un compromis a été proposé : des horloges locales faiblement synchronisées. Trois méthodes pour calculer une borne garantie sur le temps de traversée ont été proposées. La configuration 1 a permis de montrer qu'elles sont incomparables, *i.e.* il n'y a pas une des méthodes qui surpasse toujours les deux autres, même s'il apparaît que la méthode (**M.3**) donne des résultats plus précis lorsque les phases sont faibles ($\pm$1ms dans notre exemple), et que la méthode (**M.2**) donne elle les meilleurs résultats, en moyenne. Pour la suite ces trois méthodes ne seront plus étudiées séparément, au contraire le résultat qui sera retenu sera le minimum des trois bornes obtenues, *i.e.* nous savons que le temps de traversée est inférieur à ces trois valeurs donc il est inférieur à leur minimum.

Cette première expérimentation a permis de mettre en évidence que l'utilisation d'un bus faiblement synchronisé est très avantageux. En comparant l'utilisation d'un bus faiblement synchronisé à la solution d'horloge commune, il apparaît qu'une phase d'au plus 1 ms conduit à une augmentation des délais maximaux limitée, inférieure à 3 ms pour le réseau à 250 kbit/s utilisé dans cette configuration. Si l'on compare maintenant le bus faiblement synchronisé à la solution des horloges locales, cette expérience a montré qu'une phase d'au plus 5ms fournit déjà un gain moyen d'environ 40%. Ce sont des résultats très prometteurs car cela montre qu'un synchronisation peu coûteuse ne dégrade que légèrement le temps de traversée par rapport à une synchronisation parfaite et qu'au contraire elle permet un

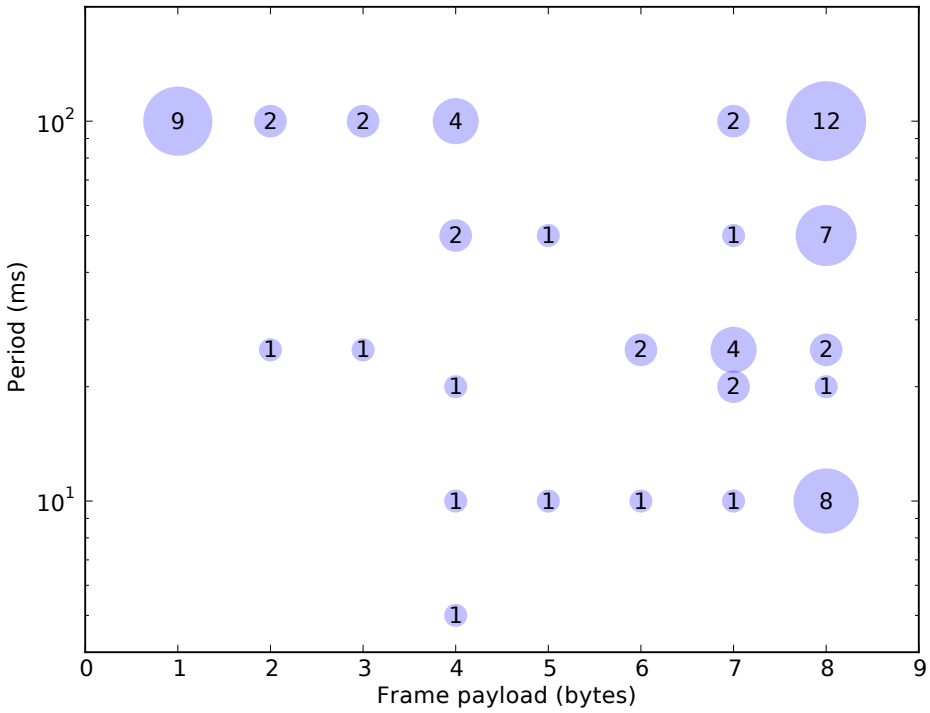


FIGURE 6.14 – Distribution de la taille et de la période des messages de la configuration 2.

gain très important par rapport à une absence de synchronisation entre les nœuds.

Cette première configuration comportait cependant un certain nombre d'hypothèses simplificatrices. Si ces premiers résultats sont prometteurs il reste encore beaucoup de paramètres à évaluer, comme par exemple l'impact des flux asynchrones. C'est entre autre ce qui sera fait dans la configuration 2.

Configuration 2

La première configuration a permis une comparaison des différentes méthodes qui ont été développées pour modéliser le cas d'un bus faiblement synchronisé. Un certain nombre d'hypothèses simplificatrices avaient été faites afin de simplifier la première interprétation en limitant le nombre de paramètres à prendre en compte, comme par exemple considérer que le bus n'était parcouru que par des flux synchrones, qu'il n'y avait pas d'erreurs sur le bus ou enfin ne pas prendre en compte les messages de synchronisation. Pour cette deuxième expérimentation l'étude porte sur une configuration réelle d'un bus CAN présentée dans [Zeng et al., 2010] et le modèle tentera d'être le plus complet possible.

Cette configuration est composée de 6 nœuds identiques qui partagent le même bus CAN. Il y a 69 flux dans le système. La Figure 6.14 présente la distribution des différents flux. On remarque que cette configuration est beaucoup moins homogène que celle présentée dans le cas précédent, il n'y a par exemple qu'un flux avec une période de 5ms alors qu'il y en a 31 avec une période de 100ms. La vitesse du lien est de 500kbit/s ce qui donne une utilisation du bus de 60.25%, beaucoup plus que précédemment. Comme

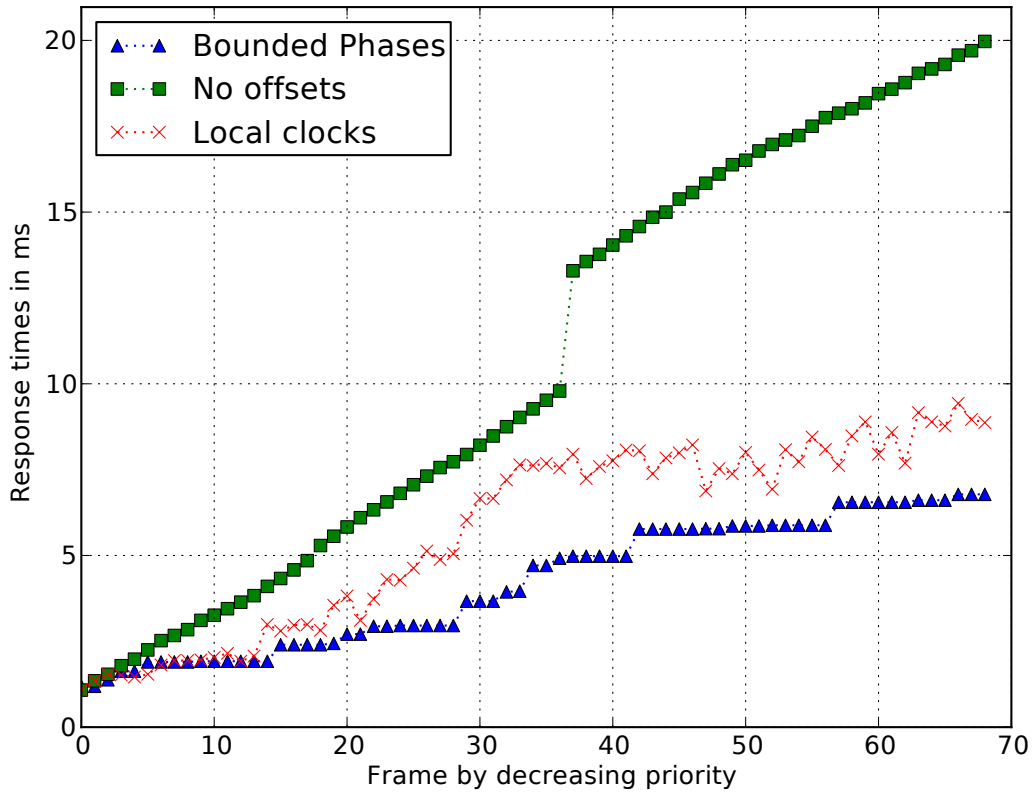


FIGURE 6.15 – Bornes sur le délai lorsque tous les flux sont synchrones (phase=1ms).

cela a été expliqué dans la Section 6.1.2 on considère que le mécanisme de synchronisation nécessaire dans le cas d'un bus faiblement synchronisé nécessite un flux prioritaire supplémentaire. Dans cette expérimentation ce dernier sera pris en compte, lorsque l'on donnera une borne sur le délai dans le cas d'un bus faiblement synchronisé un flux additionnel aura donc été pris en compte. De plus nous ne considérerons plus que le cas où la phase est bornée par plus ou moins une milliseconde. Enfin pour être le plus proche possible de la réalité le lien ne sera pas considéré comme parfait et les erreurs seront modélisées comme décrit dans la Section 6.2.1 avec $N_{erreur} = 2$ et $T_{erreur} = 100ms$ (ces valeurs sont pessimistes pour un bus CAN classique mais permettent de garantir nos bornes pour un cas réel).

100 % de flux synchrones Dans un premier temps tous les flux seront supposés synchrones. Les résultats sont présentés sur la Figure 6.15 ils représentent les délais dans trois situations : dans le cas d'un synchronisation faible où la phase est bornée par plus ou moins 1ms, le cas où les horloges locales ne sont pas synchronisées et une dernière situation où les offsets ne sont pas utilisés. Pour rappel les résultats de la synchronisation faible prennent en compte un flux supplémentaire, utilisé pour maintenir cette synchronisation, considéré comme ayant la plus haute priorité.

Les résultats révèlent un gain moyen d'environ 53% par rapport à un système n'utilisant pas d'offset et un gain moyen d'environ 22% par rapport à un système avec des horloges locales.

		Synchrone	Asynchrone
Priorités	**1-17**	50%	50%
	18-34	75%	25%
	35-69	100%	0%
Part de la charge		60%	40%

TABLE 6.5 – Distribution des messages entre synchrone et asynchrone.

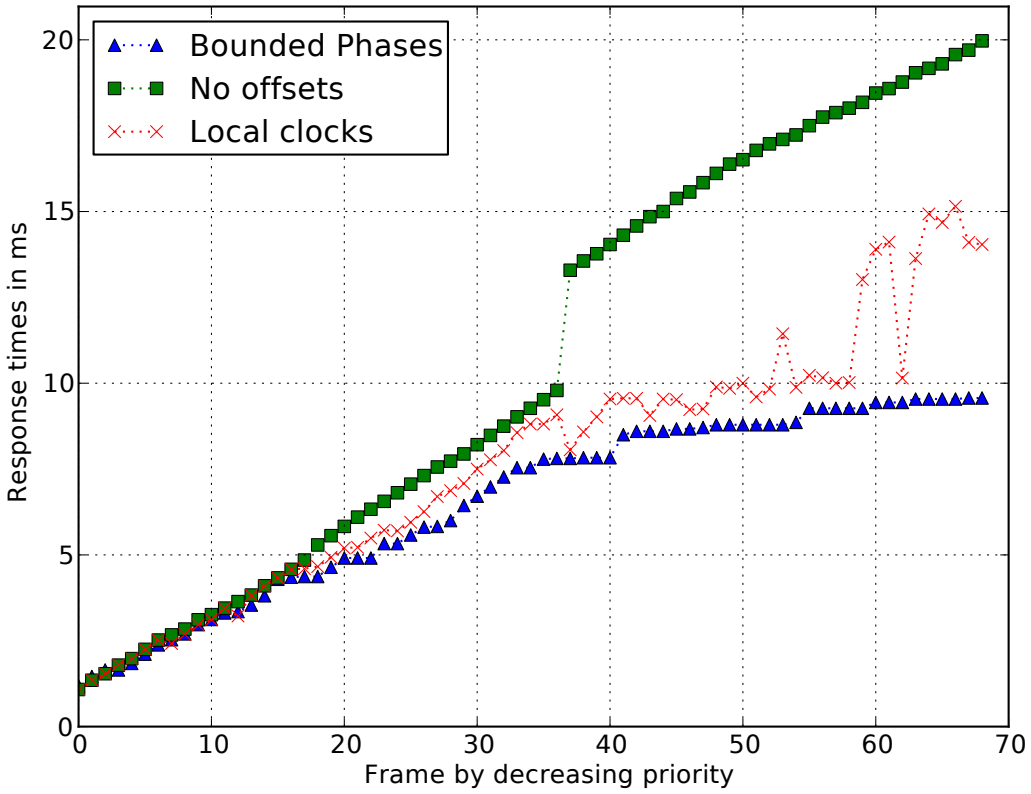


FIGURE 6.16 – Bornes sur le délai lorsque 40% des flux sont asynchrones (phase=1ms).

Pour les trames de très haute priorité (1-15), ce gain est beaucoup plus faible en effet maintenir la synchronisation nécessite d'ajouter un flux alors que ce flux n'est pas présent dans les deux autres situations lorsqu'il n'y a pas de synchronisation. Cependant le gain apporté par la synchronisation compense très rapidement le délai induit par l'ajout de ce flux supplémentaire.

40 % de flux asynchrones Dans un second temps, une partie des flux sera supposée asynchrone. Les flux asynchrones seront choisis de façon préférentielle parmi les flux ayant la plus grande priorité afin de modéliser leur utilisation pour les alarmes. Ainsi il a été décidé de considérer comme sporadiques 50% des flux ayant une priorité entre 1 à 17 et 25% de ceux ayant une priorité entre 18 à 34. Tout le reste du trafic reste périodique. La distribution est résumée dans la Table 6.5.

Il est important de noter que dans cette configuration (décrite en détail

dans [Zeng et al., 2010]) les flux les plus prioritaires sont aussi des flux avec la plus petite période, c'est pourquoi dans ce système, plus de 40% de la charge totale ont été changés en trafic asynchrones.

Les résultats sont présentés Figure 6.16. Pour les trames à priorité élevée (1-30), le gain lié à l'utilisation d'offset est limité car une partie importante du trafic est asynchrone, de plus, comme mentionné précédemment, il existe un flux supplémentaire pour maintenir la synchronisation qui augmente le délai. Cependant pour les trames de faible priorité (40-75) le gain dû au décalage reste très important, 45% par rapport à un système n'utilisant pas d'offsets et 17% par rapport à un système sans synchronisation avec uniquement des horloges locales.

Configuration 2 : Bilan Cette configuration a permis de montrer plusieurs résultats intéressants. Tout d'abord les méthodes de modélisations proposées pour le cas d'un bus faiblement synchronisé peuvent être utilisées dans un cas réel. Il est possible de prendre en compte les flux asynchrones et les erreurs. De plus nous avons montré que si l'ajout d'un flux prioritaire afin de maintenir la synchronisation entre les différentes horloges a un effet négatif sur le délai, cependant ce dernier est largement compensé par le gain apporté grâce à la synchronisation même faible qui en découle.

6.3.2 Évaluation du gain de l'utilisation du bus obtenue grâce à l'utilisation d'un bus faiblement synchronisé

Nous avons donc pu évaluer à l'aide de ces deux exemples que l'utilisation d'un bus faiblement synchronisé permet de réduire très significativement le pire temps de traversée du réseau. Dans un système temps réel chaque flux doit respecter ses contraintes temporelles, il faut donc être capable de garantir que le délai maximum du flux est inférieur à sa *deadline*. Cette contrainte empêche de charger trop le lien, cependant ne pas charger le lien c'est risquer d'avoir un réseau sur-dimensionné,c'est-à-dire risquer un coût supplémentaire non nécessaire. Nous avons vu précédemment que l'utilisation d'un bus faiblement synchronisé permet de réduire le temps de traversée, il doit donc être possible d'augmenter la charge maximale du lien. C'est ce que nous allons évaluer dans cette partie. Pour cela, dans un premier temps nous définirons un paramètre pour évaluer la part du bus qui est utilisable puis nous l'utiliserons pour estimer le gain sur l'utilisation du bus qu'il est possible d'espérer en utilisant un bus faiblement synchronisé.

Définition du *breakdown utilisation*

Afin d'évaluer la part du bus qui peut être utilisée nous allons utiliser le critère du *breakdown utilisation*. Le *breakdown utilisation* a déjà été utilisé dans la littérature dans plusieurs articles comme par exemple [Davis and Navet, 2012, Davis et al., 2011, Davis et al., 2013].

Le *breakdown utilisation*, pour une configuration donnée, correspond à la part maximale d'utilisation du bus qu'il est possible d'atteindre tout en garantissant que toutes les *deadlines* sont respectées. Expliquons cette notion à travers un exemple : soit un ensemble de nœuds émettant 150 kilobits de données par seconde sur le bus ; si pour garantir que toutes les *deadlines* sont respectées il faut un lien d'une vitesse minimale de 300kbit/s

alors le *breakdown utilisation* pour cette configuration sera de 50%. Cette valeur de 50% permet de dimensionner la vitesse du lien, il faut que la charge soit au plus de 50% donc que la vitesse soit d'au moins 300kbit/s.

Nous avons donc un critère qui pour une configuration donnée, donne la part maximale du bus qu'il est possible d'utiliser. Cependant d'une configuration à l'autre cette valeur sera différente. En effet beaucoup de paramètres ont un impact sur les délais (la taille des paquets, la période des flux, le nombre de flux, le nombre d'émetteurs, les offsets, etc...). Or ce qui nous intéresse ici c'est de pouvoir comparer différentes approches (sans offset, horloges locales et faible synchronisation), pour avoir une idée de la charge moyenne que l'on peut espérer d'une configuration a-priori.

Pour cela, l'idée consiste à générer un nombre très important de configurations, dans notre cas 5000. Pour chacune des configurations il faut ensuite calculer le *breakdown utilisation* pour chacune des situations que l'on considère, c'est-à-dire le cas où l'on ne considère pas d'offset, le cas où les horloges sont locales et non synchronisées et le dernier cas ou le bus est faiblement synchronisé. Pour chaque situation l'on obtient donc 5000 *breakdown utilisation*. Il est possible de les représenter sous forme d'histogramme comme c'est le cas sur la Figure 6.17a. On peut voir par exemple qu'il y a un peu plus de 500 flux qui ont un *breakdown utilisation* compris entre 50 et 51%. Sur la Figure 6.17a il a été représenté en vert tous les flux qui ont un *breakdown utilisation* supérieur à 50%. Comment interpréter ce résultat ? Il faut comprendre que pour cet ensemble de flux, si la vitesse du lien est suffisante pour que l'utilisation du bus soit d'au plus 50% alors on a la garantie que toutes les contraintes temporelles seront respectées. A l'inverse pour les autres flux on ne peut plus garantir que ces contraintes seront respectées. On peut voir sur ces histogrammes qu'avec une charge de 38% les contraintes seront respectées pour tous nos flux et qu'en moyenne on pourra espérer monter à une charge de 50%. Pour que le résultat soit plus lisible nous avons décidé de représenter les histogrammes par leurs approximations continues (appelée "kernel density estimation" dans python (*scipy.stats.gaussian_kde*) comme c'est le cas sur la Figure 6.17c, le résultat s'interprète ensuite de la même façon. Pour une question de visibilité ces fonctions ne seront souvent tracées qu'entre 35% et 100%.

Pour résumer voici le protocole suivi dans la suite de cette section :

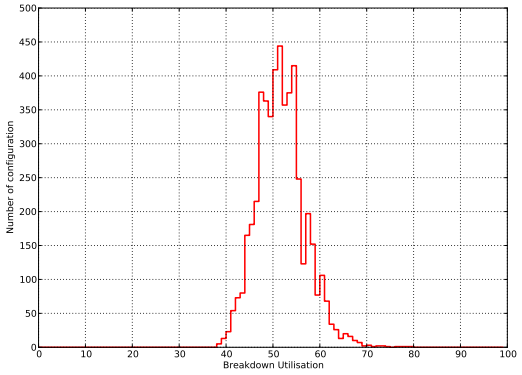
— 5000 configurations vont être générées de façon aléatoire.

— Pour chaque configuration le *breakdown utilisation* est calculé dans trois situations : sans offsets, avec des horloges locales et enfin avec des horloges faiblement synchronisées.

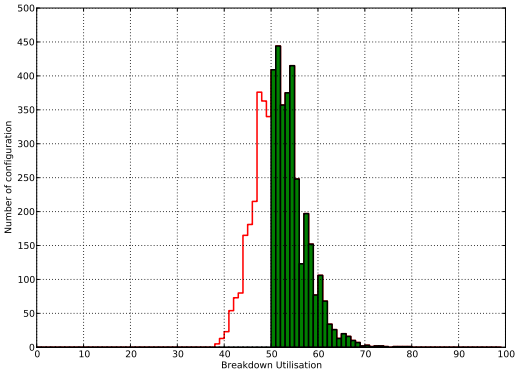
— Trois histogrammes sont ainsi obtenus, et leur comparaison permet d'évaluer le gain sur l'utilisation du bus apporté par l'utilisation d'un bus faiblement synchronisé.

Chaque configuration est identique pour les trois situations, excepté pour le cas où l'on considère les horloges faiblement synchronisées. Lorsque cette situation est considérée l'on rajoute un flux de priorité maximale pour modéliser les messages de synchronisation.

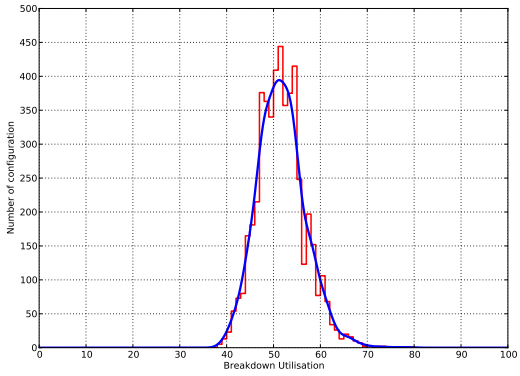
Le *breakdown utilisation* est calculé en faisant diminuer de façon itérative la vitesse du lien et en calculant des bornes sur le temps de traversée avec une méthode (comme par exemple celle du Théorème 13). La dernière vitesse du lien où toutes les contraintes temporelles sont respectées permet d'obtenir le *breakdown utilisation* pour la méthode utilisée. Pour chacune des situations une ou plusieurs méthodes seront utilisées :



(a) Histogramme de la distribution.



(b) *Breakdown utilisation* supérieur à 50%.



(c) Approximation de la distribution.

FIGURE 6.17 – Représentations d'une distribtion de *breakdown utilisation*.

— **Sans offset :** Calcul du temps exact de traversée présenté dans [Davis et al., 2007]. Le résultat sera représenté en noir.

— **Horloges locales :** Le calcul a été fait à l'aide de l'algorithme publié dans [Yomsi et al., 2012] et disponible dans RTaW-Pegase. Cet algorithme est exact à une trame prêt (surestimation du facteur de blocage de la trame CAN). Il sera représenté en vert.

— **Horloges faiblement synchronisées :** Nous avons utilisé le minimum des trois méthodes présentées précédemment dans les Théorèmes 13 et 14. Ce résultat est lui représenté en rouge. Sauf indication contraire nous considérons que la phase entre les horloges est bornée par ± 1ms.

Nous avons donc maintenant une méthode qui nous permet de comparer le gain apporté par l'utilisation d'horloges faiblement synchronisées sur l'utilisation du bus. Cependant il nous reste à définir comment générer les 5000 configurations qui nous serviront pour nos expérimentations.

Modèle de nos 5000 configurations

Il nous faut maintenant définir comment générer nos 5000 configurations représentatives d'un bus CAN "classique". Malheureusement il n'existe pas de bus CAN "classique", c'est un bus très largement utilisé et il existe un nombre très important de configurations différentes : seulement composé de flux asynchrones, présence d'un *gateway* générant une part importante du trafic... Puisqu'il n'existe pas de configurations type, notre choix, inspiré de [Davis and Navet, 2012, Zeng et al., 2010], a été le suivant :

— 16 nœuds connectés via un bus CAN, échangent des messages périodiques asynchrones (sauf lorsque l'inverse est précisé).

— Chaque priorité est assignée de façon aléatoire en suivant une distribution uniforme.

— La période des flux est choisie de façon uniforme dans l'ensemble $\{20, 25, 40, 50, 100, 200\}$ms.

— La taille maximale des données est de 8 octets.

— La charge à 500kbit/s est de 50%, ce qui représente entre 35 et 55 flux selon les configurations.

— Le choix des offsets a été fait en utilisant l'algorithme SOPA disponible dans l'outil RTaW-Pegase.

— La contrainte temporelle de chaque flux est définie comme sa période (sauf indication contraire), ce choix est désigné par *implicit deadlines*.

— Chaque flux a une source choisie aléatoirement de façon uniforme parmi les 16 nœuds (sauf indication contraire).

Comme cela est indiqué la génération est légèrement différente dans certaines expérimentations, si c'est le cas le nouveau modèle sera alors détaillé. Sans indication contraire les 5000 configurations ont été générées en suivant les règles énoncées précédemment.

Maintenant que nous avons expliqué comment sont générées les 5000 configurations il est possible de passer aux expérimentations. Chacune d'entre elle s'intéressera a un problème en particulier comme l'impact des erreurs par exemple.

Expérimentation 1 : *deadline-monotonic*

Pour cette première expérimentation il va falloir revenir sur un point que l'on a peu évoqué jusque là : le choix des priorités.

Dans le cas préemptif, il a d'abord été montré [Liu and Layland, 1973] que, dans le cas où les *deadlines* sont égales aux périodes, l'algorithme $rate - monotonic$, qui attribue la priorité la plus forte à la tâche qui possède la plus petite période, est optimal parmi toutes les choix possibles. Pour le cas plus général où les délais sont égaux ou inférieurs aux périodes, cet algorithme peut être adapté en *deadline-monotonic* et il reste optimal, voir [Leung and Whitehead, 1982]. Cependant, dans le cas général où il n'y a pas de relation entre les périodes et les délais, ces algorithmes ne sont plus optimaux, une procédure optimale d'affectation de priorité a été fournie dans [Buttle, 2012], connue sous le nom d'algorithme Audsley.

Contrairement au cas préemptif, dans le cas non-préemptif même lorsque les *deadlines* sont inférieures ou égales aux périodes, l'algorithme *deadline-monotonic* n'est plus optimal. Il a d'abord été supposé que si la taille de la trame de priorité la plus élevée est inférieure ou égale à la taille de la trame de priorité inférieure, alors l'algorithme *deadline-monotonic* était optimal [George et al., 1996]. Cependant, ce théorème s'est avéré incorrect comme le montre le contre exemple présenté dans [Davis and Burns, 2009]. L'attribution prioritaire d'Audsley présentée dans [Buttle, 2012] reste optimale dans le cas général.

Même si comme nous venons de l'expliquer l'algorithme *deadline-monotonic* n'est pas optimal il reste tout de même très performant et il est de loin le plus simple à mettre en place. Dans cette première expérience il a été fait le choix de l'utiliser en supposant tout d'abord que l'on est dans le cas d'*implicit deadlines*, c'est-à-dire que les *deadlines* sont choisies égales aux périodes. Dans cette expérimentation seule la situation sans offset a été envisagée. Le résultat est présenté sur la Figure 6.18 par la courbe noire continue. Pour 5000 configurations le *breakdown utilisation* est toujours supérieur à 85% et sa valeur moyenne est de 95%. Cela confirme bien que l'algorithme *deadline-monotonic* est très performant et permet de charger énormément le bus. On comprend bien que si l'on a la possibilité de l'utiliser, l'utilisation d'offset ne sera pas forcément utile.

Dans un second temps, nous avons donc décidé de nous intéresser à une situation où les contraintes sont plus importantes : au lieu de supposer que les *deadlines* des flux sont égales à leurs périodes, nous avons considéré qu'elles étaient égales à la *moitié* de leurs périodes. Nous avons alors étudié les trois situations. Tout d'abord lorsque l'on n'utilise pas d'offset (ligne noire pointillée de la Figure 6.18) l'utilisation de l'algorithme *deadline-monotonic* permet d'atteindre une charge moyenne de 73%. L'utilisation d'horloges locales augmente légèrement cette valeur jusqu'à 75%. Enfin dans le cas d'une faible synchronisation l'utilisation maximale du bus est proche de 100%, entre 85% et 97%.

Ces expérimentations confirment tout d'abord l'intérêt d'utiliser l'algorithme *deadline-monotonic*. De plus elles montrent que l'usage d'offsets peut être fait en combinaison avec le choix des priorités pour augmenter encore plus l'utilisation du bus. Cependant en pratique l'algorithme *deadline-monotonic* ne peut pas toujours être utilisé. En effet dans le bus CAN la priorité sert aussi d'identifiant pour un flux. Or certains identifiants sont fixés que ce soit pour une question de réutilisation du matériel (lorsque le matériel est utilisé sur un nouveau système on conserve les identifiants des flux) ou pour une question de fonction d'une tâche (tous les flux liés à une même fonction auront des identifiants similaires,

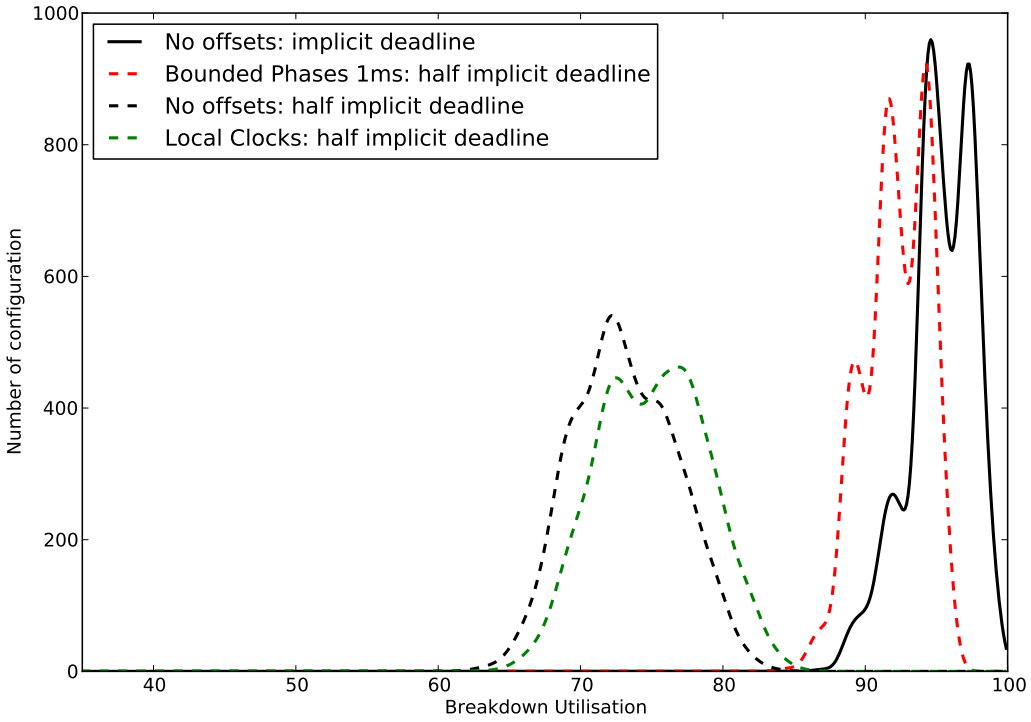


FIGURE 6.18 – *Breakdown utilisation* dans le cas *deadline-monotonic*.

voir par exemple les labels dans [ARINC 825, 2011]). Dans tous les cas en pratique il est souvent impossible de choisir librement la priorité de chaque flux, c'est pour cela que dans la suite nous supposerons que le choix des priorités est fait de façon aléatoire.

Expérimentation 2 : précision de la synchronisation.

Cette deuxième expérience ainsi que les suivantes considérera les priorités assignées aléatoirement (pour modéliser des choix de conception inconnus). Cette expérimentation a pour objectif d'évaluer l'impact de la précision de la synchronisation sur l'utilisation de bus.

Le gain dû à un bus faiblement synchronisé dépend directement de la borne sur la phase entre les horloges. Plus la synchronisation est de mauvaise qualité (borne importante), plus le gain est faible. La borne sur la phase entre les horloges dépend du mécanisme de synchronisation choisi. Notre méthode ne suppose pas un mécanisme de synchronisation spécifique, nous considérons seulement qu'il faut au plus envoyer une trame à chaque hyper-période (dans ce cas 200ms). Nous avons montré précédemment qu'avec un mécanisme de synchronisation simple, une précision d'environ 1 ms peut être obtenue. Mais l'on peut sans problème imaginer une synchronisation meilleure ou au contraire moins bonne. C'est pourquoi, nous allons comparer le *breakdown utilisation* de plusieurs solutions : pas d'utilisation d'offsets, utilisation d'horloges locales et l'utilisation d'horloges faiblement synchronisées avec des précisions de 0.5ms, 1ms et 2ms. Les résultats sont présentés à la Figure 6.19.

Tout d'abord, considérons la situation où les offsets ne sont pas utilisés. Nous allons

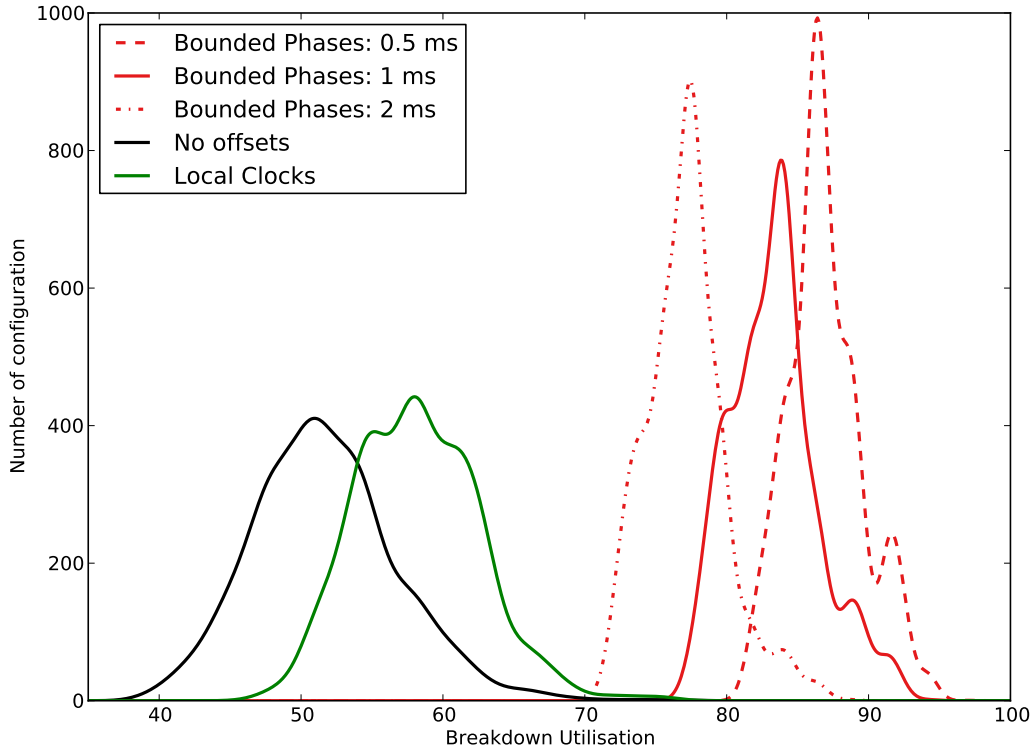


FIGURE 6.19 – *Breakdown utilisation* pour une phase bornée par 0.5ms, 1ms et 2ms.

commencer par comparer ces résultats avec ceux obtenus lorsque l'algorithme *deadline-monotonic* était utilisé. La charge maximale est beaucoup plus faible, entre 40% et 65%, avec une valeur moyenne d'environ 50%, ce qui est presque deux fois moins qu'auparavant. Une fois de plus, il confirme le résultat classique que la charge d'un bus CAN doit être inférieure à 35%, pour s'assurer que toutes les *deadlines* sont respectées [Buttle, 2012, Davis et al., 2013] (la charge de 35% prend en compte les erreurs ce n'est pas le cas ici).

Nous allons maintenant, considérer le cas des horloges locales. La charge maximale est plus grande que dans le cas sans offsets, entre 55% et 70%, avec une valeur moyenne d'environ 58%. Cela signifie que l'utilisation d'offsets avec des horloges locales permet d'augmenter le nombre de messages envoyés sur le réseau de plus de 20%.

Troisièmement, considérons la situation des phases bornées. Avec une borne sur les phases entre les nœuds de 1ms, la charge maximale du bus est beaucoup plus grande, entre 75% et 95%, avec une valeur moyenne de 83%. Ce gain est très important. Cela signifie par exemple qu'il est possible de garantir toutes les *deadlines* même avec 50% de messages supplémentaires comparé au cas avec les horloges locales.

Comme la précision de la synchronisation dans le cas des phases bornées est primordiale, d'autres calculs ont été faits avec une précision de 0.5ms et 2ms (voir Figure 6.19). Comme attendu plus la synchronisation est faible (c'est-à-dire plus grande), plus le gain est faible. Cependant, même avec une synchronisation de 2 ms, les gains restent importants (environ 15% de gain par rapport aux horloges locales). De plus quelle que soit la configuration, la charge atteignable en utilisant une synchronisation faible est toujours supérieure à la charge atteignable sans offset.

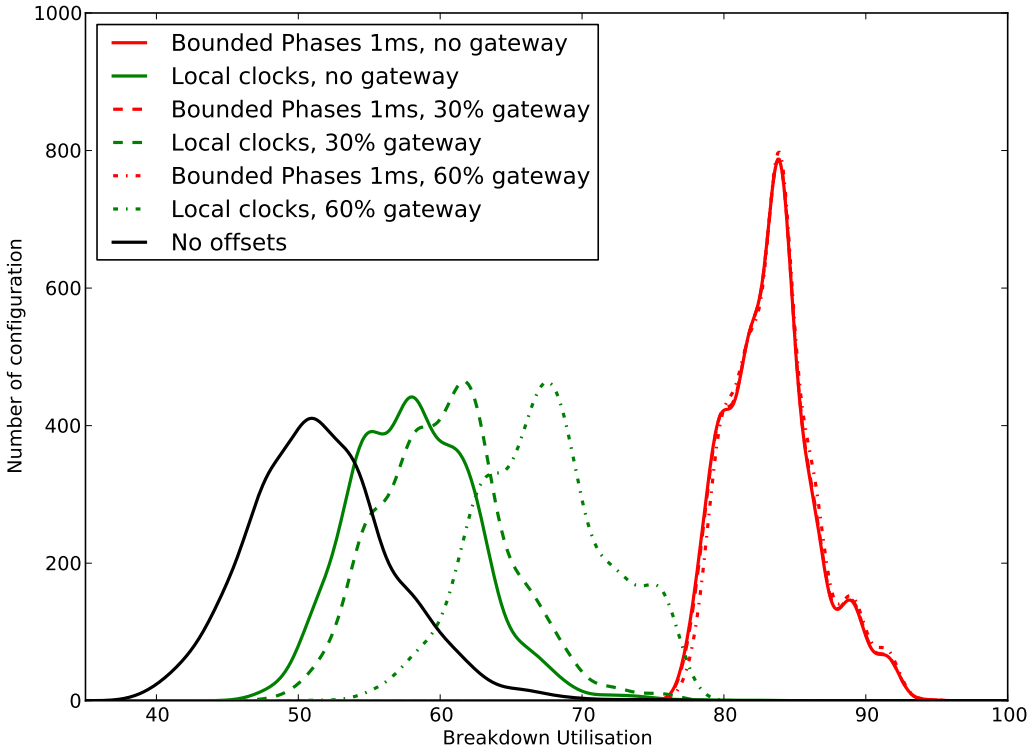


FIGURE 6.20 – *Breakdown utilisation* avec une passerelle émettant 30% ou 60% du trafic.

Expérimentation 3 : Passerelle.

Dans cette troisième expérimentation, nous allons essayer de déterminer l'impact que peut avoir une répartition non homogène des sources des messages. Cette situation est par exemple celle où l'un des nœuds est une passerelle qui envoie une part très importante du trafic sur le réseau. Par la suite nous désignerons le nœud qui émet la majorité du trafic par "passerelle" (*gateway*). Jusqu'à maintenant chacun des 16 nœuds émettait autant de trafic soit un peu plus de 6% pour chacun en moyenne. Nous allons comparer cette situation de référence à deux autres, dans le première la passerelle émet 30% du trafic et dans la seconde elle émet 60% de l'ensemble du trafic. Les résultats sont présentés Figure 6.20.

Tout d'abord il est normal que dans le cas où les offsets ne sont pas utilisés il n'y ait pas de différence. En effet il peut aussi bien y avoir des contentions intra-nœuds qu'inter-nœuds.

Dans le cas des horloges locales, plus la passerelle émet de trafic plus la part maximale augmente (de 58% à 61% dans le cas d'une passerelle à 30% et de 67% dans le cas d'une passerelle à 60%). Ce résultat est logique, les offsets dans le cas d'horloges locales ont deux effets, diminuer voire empêcher les contentions intra-nœuds et réduire les contentions inter-nœuds en lissant le trafic. En passant la majorité du trafic sur un même nœud on diminue donc le nombre de contentions.

Enfin lorsque l'on regarde le cas des phases bornées, on voit que la répartition du trafic a peu voire aucun impact sur l'utilisation maximale du bus. Encore une fois ce résultat est prévisible. Dans le cas d'une synchronisation faible les contentions inter-nœuds étaient déjà majoritairement évitées, c'est d'ailleurs ce qui explique le gain par rapport aux horloges

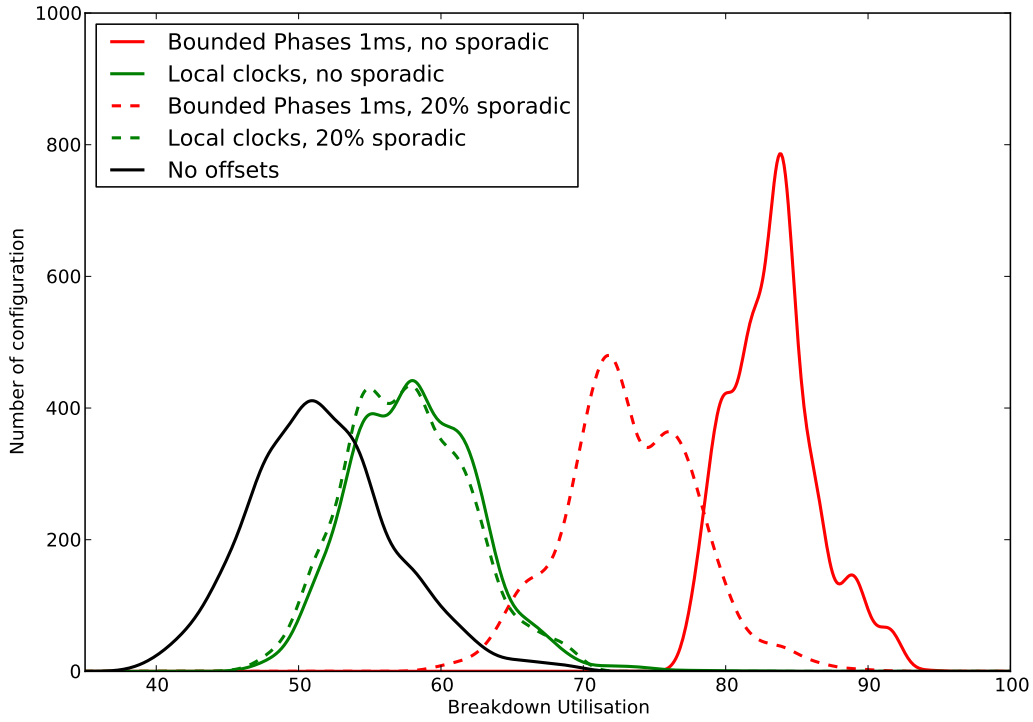


FIGURE 6.21 – *Breakdown utilisation* : pas d'erreur, 20% d'asynchrone.

locales. Passer tous le flux sur le même émetteur n'a donc que peu d'impact.

Expérimentation 4 : configuration réaliste, erreurs + asynchrones.

Dans cette dernière expérimentation, nous allons essayer de nous rapprocher autant que possible d'une configuration réelle. Pour cela, deux modifications sont nécessaires. Premièrement considérer qu'une partie des flux est asynchrone/sporadique. Deuxièmement supposer que le bus n'est pas parfait et que des erreurs se produisent.

Pour prendre en compte l'existence de flux asynchrones, 20% des flux des configurations précédentes sont modifiés pour devenir sporadiques au lieu de périodiques. Pour les erreurs nous utilisons le modèle de la Section 6.2.1 avec $N_{erreur} = 2$ et $T_{erreur} = 100ms$. Ce qui nous emmène à trois expériences, une avec 20% de flux asynchrones mais pas d'erreurs, une avec des erreurs mais pas de flux asynchrone et la dernière qui contient à la fois des messages asynchrones et des erreurs. Les résultats sont présentés Figures 6.21, 6.22 et 6.23.

Nous allons commencer par interpréter les résultats dans le cas où il n'y a pas d'erreurs mais 20% du trafic asynchrone, ce cas est présenté dans la Figure 6.21. Tout d'abord un message asynchrone est un message sans offset. On peut voir le cas sans offset comme 100% d'asynchrone. Cela a un impact faible (1.5%) sur le *breakdown utilisation* dans le cas des horloges locales et un impact plus important (10%) dans le cas d'un bus faiblement synchronisé. L'impact est plus important dans le cas du bus faiblement synchronisé car c'est dans cette situation que les offsets étaient le plus bénéfiques, c'est les offsets qui étaient à l'origine du gain. Il est donc normal qu'en les supprimant (passer les flux de synchrones avec offsets à asynchrones) on perde ce gain, et prévisible que plus le gain est

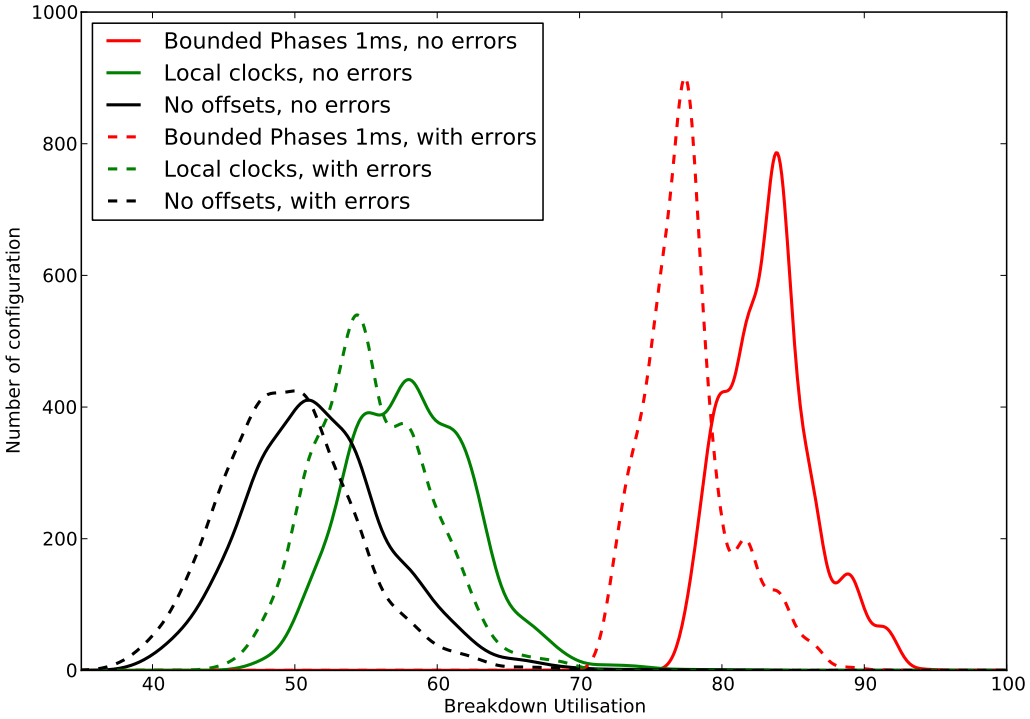


FIGURE 6.22 – *Breakdown utilisation* : des erreurs, pas d'asynchrone.

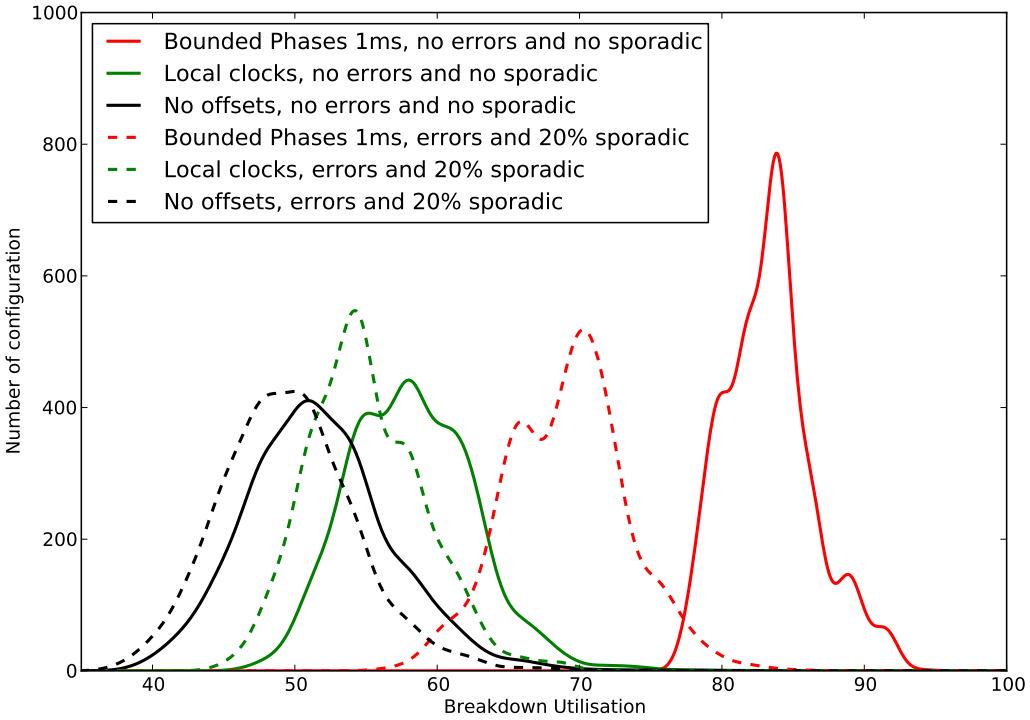


FIGURE 6.23 – *Breakdown utilisation* : des erreurs, 20% d'asynchrone.

important, plus la perte le sera. Cependant malgré 20% de trafic asynchrone, l'utilisation d'un bus faiblement synchronisé reste de loin la meilleure solution.

Nous allons maintenant interpréter les résultats dans le cas où il y a des erreurs mais que tout le trafic est synchrone : ce cas est présenté dans la Figure 6.22. Prendre en compte les erreurs implique de considérer le cas ou l'on devrait ré-émettre un message ce qui bien sûr augmente son délai maximum. Pour respecter les contraintes temporelles il est donc nécessaire de moins charger le bus. C'est bien ce résultat que l'on retrouve, et prendre en compte les erreurs a globalement le même effet quelle que soit la situation (sans offset, horloges locales, synchronisation faible).

Enfin le cas réel, avec erreur et flux sporadique est présenté Figure 6.23. On retrouve globalement les mêmes résultats que précédemment, le *breakdown utilisation* a diminué mais la conclusion reste la même : l'utilisation d'horloges faiblement synchronisées reste très avantageux d'un point de vue de l'utilisation de la bande passante.

6.4 Conclusion

Dans ce chapitre nous avons tout d'abord proposé et défini une nouvelle façon d'utiliser les offsets avec un bus faiblement synchronisé. Il avait déjà été montré que l'utilisation d'offset réduisait fortement les délais de traversée du bus en réduisant le nombre de contentions. Cependant jusqu'à présent seules deux solutions étaient proposées. La première consistait à synchroniser parfaitement (ou en tout cas avec une précision très importante) tous les nœuds du système. Tous les nœuds utilisant alors la même horloge il est possible d'établir un ordonnancement des dates d'émission des messages, les offsets, et ainsi d'éviter les contentions. Cette première solution permettait de minimiser le délai de traversée mais en contre partie était très coûteuse. Établir une telle synchronisation a un coût non seulement d'un point de vue logiciel mais aussi matériel. La deuxième solution consiste à ne pas synchroniser les nœuds, chaque nœud possède donc uniquement sa propre horloge et l'ordonnancement des dates d'émission reste local. Cela permet tout de même d'éviter les contentions intra-nœuds et de lisser le trafic ce qui diminue le temps de traversée des messages, mais de façon moins efficace qu'avec une horloge commune. L'idée d'un système faiblement synchronisé était d'obtenir un compromis entre ces deux solutions. Un délai réduit en évitant au maximum les contentions mais sans avoir un coût de synchronisation trop important.

Afin d'évaluer l'intérêt que peut avoir l'utilisation d'un système faiblement synchronisé sur le pire temps de traversée d'un message il est nécessaire d'être capable de borner le délai dans une telle situation. Or il n'existait pas jusqu'à présent de modèle capable de prendre en compte de façon satisfaisante cette synchronisation faible. Nous avons donc développé, à l'aide du calcul réseau, un modèle d'un bus faiblement synchronisé. Ce modèle permet non seulement de modéliser un bus parcouru uniquement par des flux synchrones mais aussi des flux asynchrones. De plus nous avons montré comment ce modèle permettait de prendre en compte les possibles erreurs sur le bus ainsi que les messages liés à la synchronisation entre les horloges.

Il restait en suite à évaluer le gain apporté par l'utilisation d'un bus faiblement synchronisé. C'est ce qui a été fait, tout d'abord à travers une comparaison sur les délais garantis en fonction de la solution retenue : soit sans utiliser d'offset soit avec des horloges locales

soit avec un système dont la phase entre les horloges est bornée. Ces expériences ont révélé qu'une synchronisation faible permettait une diminution très significative des délais (jusqu'à 40%) et que même lorsque l'on considérait une partie des flux comme asynchrones les gains sur les délais restaient importants. Par la suite, nous avons cherché à évaluer le gain sur l'utilisation maximale du bus qu'apportait cette diminution du délai maximum. Pour cela 5000 configurations ont été générées, et différentes situations ont été envisagées. Ces expérimentations ont permis de montrer beaucoup de résultats intéressants. Tout d'abord le fait que l'on puisse utiliser les offsets en combinaison avec un algorithme de choix de priorité, il est alors possible d'atteindre 95% de charge lorsque les *deadlines* valent la moitié des périodes pour chaque flux. Ensuite grâce à l'expérience nous avons pu voir que la répartition de la charge émise entre les nœuds n'impactait que très légèrement les délais lorsque le bus est faiblement synchronisé, la présence d'une passerelle qui émet la majorité du trafic ne modifie pas le pire temps de traversée du réseau. Enfin dans un cas réel avec des erreurs et des flux sporadiques, la synchronisation faible permet d'utiliser en moyenne près de 70% du lien alors que sans utiliser d'offsets on n'atteindra même pas les 50%.

Chapitre 7

Étude des interactions entre flux Rate-Constrained et Time-Triggered dans TTEthernet

7.1 La synchronisation dans TTEthernet

Les réseaux en bus, comme le bus CAN étudié dans le Chapitre 6 par exemple, sont limités à la fois en nombre maximum de nœuds, en débit maximum et en nombre de messages échangés. Afin de pallier ces limitations, le domaine avionique s'est tourné vers une solution de réseau commuté utilisant la technologie Ethernet : l'AFDX (*Avionics Full DupleX Switched Ethernet*). Cette nouvelle solution a très bien rempli ce rôle. L'AFDX est une technologie Ethernet sans mécanisme temps réel, mais où la garantie de temps de réponse vient des restrictions sur le trafic d'entrée et de la méthode d'analyse. Par la suite, une solution proposée a été d'ajouter la possibilité d'utiliser un trafic dirigé par le temps (*Time-Triggered*) ce qui a donné TTEthernet.

Dans ce chapitre nous allons étudier les interactions qui existent entre les flux synchrones et les flux asynchrones dans TTEthernet.

7.1.1 Différents niveaux de priorité et rôle de la synchronisation

Pour rappel dans TTEthernet les *End-Systems* échangent des messages au travers de *virtual links*. Un *virtual link* définit une connexion unidirectionnelle d'un *End-System* émetteur à un ou plusieurs *End-Systems* récepteurs : communication *Uni- ou Multicast*. Chaque *virtual link* a une bande passante réservée, exprimée à travers : une taille maximum des paquets, (*Maximum Frame Size*- **MFS**), et une durée minimale entres deux trames (*Bandwidth Allocation Gap* -**BAG**). Dans TTEthernet il y a trois niveaux de priorité, chacun correspondant à une classe de trafic particulière : *Time-Triggered*, *Rate-Constrained*, et *Best-Effort*.

— *Time-Triggered* **TT** : les messages sont envoyés en fonction du temps en suivant un ordonnancement prédéfini. C'est le trafic le plus prioritaire. L'idée derrière ce trafic est de lui allouer des dates où les messages seront envoyés et d'empêcher les trafics moins prioritaires de retarder son émission.

— *Rate-Constrained* **RC** : les messages sont envoyés en fonction d'un événement, mais respectent la durée minimale entre deux trames du même flux. Ce trafic est moins prioritaire que le trafic TT.

— *Best-Effort* **BE** : les messages correspondent au trafic standard d'une communication Ethernet. Il n'y a aucune garantie sur la transmission des messages. Le trafic *best-effort* utilise la bande passante restante du réseau et a une priorité inférieure aux deux autres types de trafic.

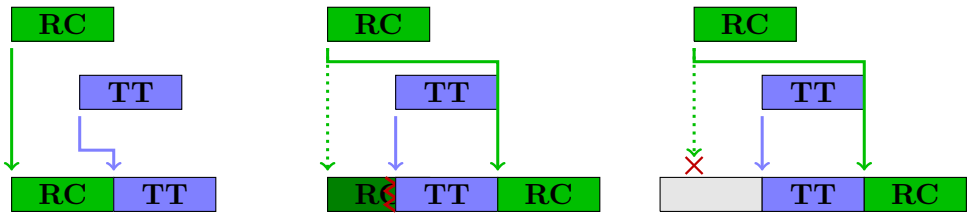
Si dans le cas du bus CAN les offsets jouaient surtout un rôle global, en limitant les contentions, dans le cas de TTEthernet un VL *Time-Triggered* est plus prioritaire et l'ordonnancement est à l'origine de ses garanties temporelles à condition de ne pas attendre à l'émission, ce qui suppose que la partie réseau et la partie applications sont synchronisées entre elles. Dans cette étude nous nous concentrons sur la partie réseau mais il est important de garder à l'esprit que l'utilisation d'offset peut ajouter un délai supplémentaire entre la partie applications et la partie réseau. En effet, en supposant que l'ordonnancement est fait correctement, il n'y a alors aucune contention entre les flux TT, de plus le niveau de priorité plus élevé des flux TT par rapport au flux RC permet de garantir les temps d'attente au niveau des commutateurs (ce point sera détaillé dans la section suivante) et ils ont donc de bien meilleures caractéristiques temps réels. Cependant cela a bien évidement un coût, le fait d'ajouter ce nouveau niveau de trafic implique de devoir "dégrader" le service offert au flux *Rate-Constrained*. Une question est alors d'être capable de borner le temps de traversée des flux *Rate-Constrained* et pour cela il faut évaluer l'impact de leurs interactions avec les flux *Time-Triggered*.

Avant cela nous allons détailler les politiques d'intégration qui existent dans TTEthernet.

7.1.2 Différentes politiques d'intégration

Nous avons déjà évoqué le fait dans le Chapitre 6 que lorsque des messages asynchrones sont émis il peut se produire un phénomène de contentions. C'est-à-dire que deux messages cherchent à être émis en même temps. Dans ce cas il est nécessaire d'avoir une politique de choix pour déterminer quel message sera émis. Il y a ensuite plusieurs possibilités, soit les deux messages ont la même priorité, la politique suivie dans TTEthernet est alors celle du FIFO, soit l'un des deux messages est plus prioritaire. En particulier le cas qui nous intéresse est celui où un message prioritaire tente d'être émis alors qu'un message moins prioritaire est déjà en cours d'émission. Il y a alors dans TTEthernet trois politiques d'intégration [Steiner et al., 2009] : le *"Shuffling"*, la *"Préemption"* et le "*Timely Block*". Ces politiques sont visibles sur la Figure 7.1 dans le cas d'un message RC et d'un message TT et vont être détaillées ci-dessous.

Le Shuffling L'envoi du message de faible priorité ayant commencé on le termine et c'est seulement après que le message le plus prioritaire sera transmis. C'est une méthode non-préemptive. Cette méthode est simple à mettre en place et du point de vue d'utilisation de la bande passante c'est la plus intéressante puisqu'il n'y a pas d'instant où le lien est inactif. Mais si le message prioritaire est de type TT alors elle implique de lui rajouter un délai supplémentaire, le réseau offrira donc de moins bonnes caractéristiques temps réels.



(a) Shuffling. (b) Preemption. (c) Timely block.

FIGURE 7.1 – Les politiques d'intégration proposées dans TTEthernet.

Dans TTEthernet, c'est toujours la politique de *shuffling* qui s'applique lorsque la trame prioritaire n'est pas de classe TT (par exemple en cas de contention RC/BE). Les deux politiques suivantes ne concernent donc que des contentions avec des trames TT.

La Préemption Lorsque le message prioritaire (TT) devient disponible, on interrompt l'émission du message de faible priorité qui sera renvoyé entièrement une fois que le message prioritaire aura été transmis. Cette méthode reste simple à mettre en place et contrairement à précédemment, la trame TT est envoyée comme cela avait été prévu initialement. Par contre, on perd de la bande passante et il faudra distinguer le début de la trame préemptée d'un réel message. Cette solution n'a été implémentée que dans les premières versions, versions académiques, de TTEthernet [Steiner et al., 2009].

Le Timely Block Le Timely Block est basé sur l'idée que l'on sait *a priori* quand sera envoyée une trame TT, il est donc possible d'empêcher l'émission d'une trame RC si celle ci gêne autrement l'envoi du message TT. On désigne par *guard band* cette durée durant laquelle l'émission est empêchée. Cette méthode est plus compliquée à mettre en place, mais la trame TT sera envoyée comme cela avait été prévu initialement. On perd toujours de la bande passante puisqu'on arrête d'émettre pendant un moment, il y a donc un moment où le lien est inactif.

De plus il existe un type de trafic spécifique utilisé pour établir et maintenir la synchronisation. Les messages de ce type de trafic sont désignés comme PCF (Protocol Control Frames). Ils ont une petite taille (64 octets) et appartiennent à une catégorie particulière de flux RC qui a la priorité la plus élevée. La politique utilisée pour ces flux en particulier, est toujours le *shuffling*. Ainsi lorsque l'ordonnancement des trames TT est construit il prend toujours en compte au minimum un possible décalage dû aux PCF et potentiellement un décalage supplémentaire en cas de *shuffling*.

Maintenant que nous avons présenté les différentes politiques d'intégration disponibles dans TTEthernet nous allons voir comment les prendre en compte pour borner le temps de traversée des messages RC.

FIGURE 7.2 – Exemple d'un flux RC.

7.2 Modélisation des flux *Rate-Constrained* à l'aide du calcul réseau

Tout comme dans le chapitre précédent nous allons maintenant chercher à borner le délai d'un message de classe RC. La politique d'ordonnancement entre flux RC étant FIFO c'est le Théorème 6 vu au Chapitre 4 qui s'applique.

Propriété 4. *Soit S un serveur FIFO. Soit $\{A_{RC,1}, ..., A_{RC,n}\}$ n flux RC dont les courbes d'arrivée sont connus $\alpha_{RC,i}$. Si S offre un service résiduel simple β_{RC} à l'ensemble de flux RC alors le délai du flux j ($d_{RC,j}$) est borné par :*

$$d_{RC,j} = hDev(\sum_{i=1}^{n} \alpha_{RC,i}, \beta_{RC}) \tag{7.1}$$

Il nous faut donc estimer les courbes d'arrivée des flux RC mais aussi déterminer le service résiduel simple que leur offre le serveur.

7.2.1 Calcul des courbes d'arrivée des flux RC

Un flux RC étant asynchrone et de taille de paquet variable son flux d'arrivée n'est pas connu. Chaque flux RC est cependant associé à une taille maximale de paquets (**MFS**) et à une distance minimale entre l'émission de deux trames successives (**BAG**). De plus la norme définit la gigue maximum en sortie de *End System* J_{max} pouvant affecter le BAG comme :

$$\begin{cases} J_{max} \leq 40\mu s + \frac{\sum_{j \in \{\text{set of VLs}\}} ((20+L_{max_j})*8)}{net\_bandwidth} \\ J_{max} \leq 500\mu s \end{cases}$$

Les différentes caractéristiques d'un flux RC ont été représentées sur la Figure 7.2.

Il est alors possible de définir la courbe d'arrivée d'un flux RC comme :

$$\alpha_{RC}(t) = MFS \left\lceil \frac{t + J_{max}}{BAG} \right\rceil \tag{7.2}$$

Démonstration. La courbe cumulative d'un flux RC A_{RC} à un instant t vaut :

$$A_{RC}(t) = \sum_{k} h_k \mathbb{1}_{\{>O_k+J_k\}}(t)$$

Où h_k représente la taille du k-ième paquet, O_k sa date d'émission si sa gigue était nulle et J_k sa gigue. Ces trois informations sont inconnues mais peuvent être bornées :

$$O_{k+n} - O_k \geq nBAG \qquad h_k \leq MFS \qquad 0 \leq J_k \leq J_{max}$$

De plus, il existe une date d'émission $O_j + J_j \geq t$ telle que $A_{RC}(t) = A_{RC}(O_j + J_j)$. Il est donc possible d'en déduire :

$$\begin{aligned} A_{RC}(t+d) - A_{RC}(t) &= A_{RC}(t+d) - A_{RC}(O_j + J_j) \\ &\leq A_{RC}(O_j + J_j + d) - A_{RC}(O_j + J_j) \\ &\leq \sum_{k \geq j} h_k \mathbb{1}_{\{>O_k + J_k - O_j - J_j\}}(d) \\ &\leq \sum_{k \geq j} MFS \mathbb{1}_{\{>O_k - O_j - J_j\}}(d) \\ &\leq \sum_{k \geq j} MFS \mathbb{1}_{\{>(k-j)BAG - J_{max}\}}(d) \\ &\leq MFS \left\lceil \frac{t + J_{max}}{BAG} \right\rceil \end{aligned}$$

□

Maintenant que l'on a calculé la courbe d'arrivée d'un flux RC il nous faut calculer le service résiduel offert à l'ensemble des flux RC.

7.2.2 Calcul du service résiduel offert à l'ensemble des flux RC

Pour pouvoir calculer le service résiduel offert à l'ensemble des flux RC il faut déjà savoir quels sont les autres flux qui partagent le serveur. Il y a tout d'abord les flux TT, plus prioritaires, comme on l'a vu précédemment plusieurs politiques d'intégration existent et leur impact sera donc variable. Il y a aussi les flux BE moins prioritaires, c'est alors le *shuffling* qui est la politique d'intégration appliquée. Enfin il y a les flux de synchronisation, avec l'émission de PCF, un flux prioritaire qui lui aussi utilise le *shuffling*.

En utilisant des résultats classiques du calcul réseau, nous allons décomposer β_{RC}. Tout d'abord nous allons considérer l'impact des trames BE et des PCF. Il s'agit de *static priority* non préeemptive, le Théorème 8 nous apprend que l'on peut définir β_{TT+RC} le service résiduel offert au flux TT et au flux RC comme :

$$\beta_{TT+RC} = [\beta - \alpha_{PCF} - L_{BE}^{\max}]_{\uparrow}^{+} \tag{7.3}$$

Où $L_{BE}^{\max}$ est la longueur maximale d'une trame BE et α_{PCF} est la courbe d'arrivée des PCF. Les PCF sont des trames dédiées à la synchronisation d'horloge dans TTEthernet. Tous les paquets utilisés pour la synchronisation sont des PCF qui font exactement 64 octets et sont envoyés de façon périodique. Il est donc parfaitement possible de calculer leur courbe d'arrivée comme cela a été fait précédemment pour tous les autres messages de classe RC.

Il nous faut maintenant prendre en compte l'impact des flux TT. Pour cela nous allons considérer le cas général décrit par la Figure 7.3.

Dans le cas général nous allons supposer un système avec une hyper-période P, composé de N dates d'émission TT par hyper-période. Chacune de ces dates est définie par un offset o_i et une taille constante L_i. De plus l'émission d'un paquet TT_i peut être précédée par H_i (*header*) et suivie par T_i (*trailer*) de tailles variables mais bornées par $L_{H_i}^{\max}$ and $L_{T_i}^{\max}$. Le *trailer* ne sera pas utile pour modéliser TTEthernet mais pourrait avoir une

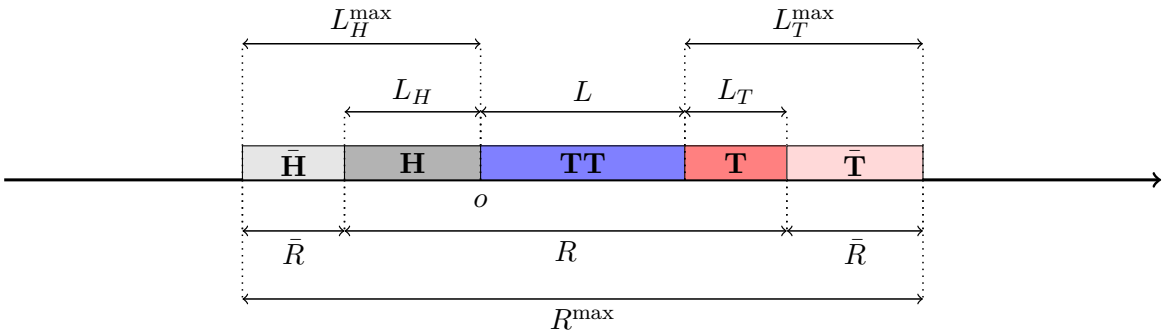


FIGURE 7.3 – Représentation de l'impact d'un flux TT.

utilité sur des technologies différentes comme TSN par exemple. On cherche ici un résultat général que l'on pourra potentiellement appliquer à plusieurs systèmes et pas uniquement à TTEthernet.

Le *header* et le *trailer* peuvent avoir plusieurs origines. Dans le cas de TTEthernet si l'on considère H cela désigne par exemple la première partie d'un paquet RC lors d'une préemption (Figure 7.1b). Ce paquet ne sera pas pris en compte puisque l'on réémet entièrement le paquet RC et il ne peut apparaître que devant un paquet TT puisque c'est ce dernier qui en est la cause directe. La taille de ce "paquet perdu" devant une trame TT ne peut pas être connue *a priori*. Cependant on en connaît une borne supérieure : la taille maximale d'un paquet RC. L'*header* H peut aussi désigner une *guard band*, c'est-à-dire une fenêtre où on empêche l'émission d'une trame RC si celle ci gêne autrement l'envoi du message TT. Il est possible de considérer non pas que l'on arrête d'émettre à cet instant mais qu'on émet un paquet prioritaire fictif qui empêche le message RC d'être émis. Ce paquet fictif a une taille inconnue mais bornée par la taille maximale d'un paquet RC. En ce qui concerne le *trailer* T il n'a pas d'équivalent pour TTEthernet. Il pourrait être utilisé en cas de préemption avec reprise de l'émission pour signaler que le prochain message que l'on va envoyer est la suite d'un message qui a été préempté. C'est par exemple le cas dans TSN. On cherche ici à avoir un modèle général qui ne s'applique pas uniquement à TTEthernet, d'où la prise en compte du *trailer*.

La première étape consiste à ne considérer que les bornes supérieures. Si l'on considère deux flux, R_1 et R_2, une courbe d'arrivée $R_1 + R_2$ est une courbe d'arrivée de R_1 de façon évidente :

Démonstration.

$$\begin{aligned}\alpha_{R1+R_2}(d) &\geq (R_1+R_2)(t+d) - (R_1+R_2)(t)\\ &\geq R_1(t+d) - R_1(t) + R_2(t+d) - R_2(t)\\ &\geq R_1(t+d) - R_1(t)\end{aligned}$$

En effet R_2 étant un flux on sait que : $R_2(t+d) - R_2(t) \geq 0$. □

Donc, soit R_i le flux lié au i-ième message TT (incluant H_i et T_i) et $R_i^{\max}$ son flux maximal (incluant $H_i^{\max}$ et $T_i^{\max}$). Il existe $\bar{R}_i$ tel que $R_i + \bar{R}_i = R_i^{\max}$, donc, une courbe d'arrivée de $R_i^{\max}$ est aussi une courbe d'arrivée de R_i.

Donc dans la suite nous considérerons la pire courbe cumulative de l'ensemble des flux TT R_{TT}, qui peut être définie comme $R_{TT} = \sum_i R_i^{\max} = \sum_i L_{TT_i} C \left\lceil \frac{t - o_i + L_{H_i}^{\max}}{P} \right\rceil$, où $L_{TT_i} = L_{H_i}^{\max} + L_i + L_{T_i}^{\max}$ est la taille des fenêtres, $o_i - L_{H_i}^{\max}$ leur date d'émission et C désigne la vitesse du lien supposée constante.

La suite est très similaire aux calculs effectués dans le Chapitre 6. Pour tout $t \geq 0$, il existe $i \in [0, N-1]$ tel que le i-ième message TT soit le prochain message TT rencontré après la date t. Une possible borne sur la variation de R_{TT} entre cet instant t et un instant $t + \triangle t$ peut être donnée par :

$$\begin{aligned} & R_{TT}(t + \triangle t) - R_{TT}(t) = R_{TT}(t + \triangle t) - R_{TT}(o_i - L_{H_i}^{max}) \\ & \leq R_{TT}(o_i - L_{H_i}^{max} + \triangle t) - R_{TT}(o_i - L_{H_i}^{max}) \\ & \leq \sum_{j=i}^{i+N-1} L_{TT_j} C \left\lceil \frac{\triangle t + o_{i,j} - (L_{H_i}^{max} - L_{H_j}^{max})}{P} \right\rceil = \alpha_{TT,i}(t) \end{aligned} \tag{7.4}$$

où $o_{i,j}$ est la distance entre o_i et l'offset de la prochaine émission d'un message du flux j (dans la même hyper-période où la suivante), définit par $o_i - o_j$ si $o_j \geq o_i$ et par $o_i - o_j + P$ sinon.

Une courbe d'arrivée de l'ensemble de flux TT (en considérant aussi H et T) est une enveloppe supérieure des courbes définit précédemment $\alpha_{TT,i}^h$ et est donné par le Théorème 17 :

Théorème 17. *Une courbe d'arrivée de l'ensemble de flux TT est*

$$\alpha_{TT}(t) = \max_{0 \leq i \leq N-1} \{\alpha_{TT,i}(t)\}, \tag{7.5}$$

avec $\alpha_{TT,i}^h(t)$ définit dans (7.4)

Démonstration. Soit t un instant et $\triangle t$ une durée.

Nous avons déjà prouvé qu'il existe $k \in [0, N-1]$ tel que :

$$R_{TT}(t + \triangle t) - R_{TT}(t) \leq \alpha_{TT,k}(\triangle t)$$

Or de façon évidente :

$$\alpha_{TT,k}(\triangle t) \leq \max_{0 \leq i \leq N-1} \{\alpha_{TT,i}(\triangle t)\}$$

D'où le résultat. □

Comment adapter ce résultat à TTEthernet ? Cela dépend de la politique de service utilisée.

— En cas de *shuffling* H et T sont connus et nuls.

— En cas de préemption H représente une partie d'un message RC émise qui a été préemptée. Sa taille est inconnue mais peut être bornée par la taille maximale d'un message RC donc $L_H^{\max} = \frac{L_{RC}^{\max}}{C}$, avec C la vitesse du lien et $L_{RC}^{\max}$ la taille maximale d'un message RC. T est lui nul.

— En cas de *timely block* H représente la *guard band*, c'est la durée durant laquelle si l'on commence à émettre un message RC il n'aura pas fini d'être émis. Cette durée est au plus égale à la taille maximale d'un message RC donc $L_H^{\max} = \frac{L_{RC}^{\max}}{C}$. T est lui nul.

Enfin le service service résiduel offert à l'ensemble des flux RC est donc :

$$\beta_{RC}(t) = [\beta_{TT+RC}(t) - \alpha_{TT}(t)]^+_\uparrow \tag{7.6}$$

Maintenant que nous avons une expression de β_{RC}, ce qui n'avait jamais été fait au moment de la publication de [Boyer et al., 2016], il nous est possible de calculer une borne sur le temps de traversée des flux RC en utilisant la Propriété 4.

7.3 Évaluation de l'impact sur les délais des interactions entre flux TT et RC

Dans la section précédente nous avons montré comment borner le temps de traversée des messages de classe RC. Puisque nous sommes capables de borner les délais, nous allons pouvoir évaluer l'impact qu'ont les interactions avec les flux TT sur ces derniers. Pour cela nous considérerons une configuration donnée où l'ensemble des flux est de classe RC. Puis nous transformerons progressivement une partie des flux en trafic TT. Nous bornerons les délais dans chacune des situations et comparerons les variations sur les bornes ainsi obtenues. Il est difficile de prévoir la conclusion de notre expérimentation car les effets du trafic TT sur le trafic RC sont multiples et contradictoires :

— **Changement de priorité** : En passant des flux de RC vers TT nous augmentons leur priorité ce qui diminue la bande passante restante pour le reste des flux RC et augmente donc leurs délais.

— **Perte de bande passante** : Dans le cas de la politique de service *timely block* une partie de la bande passante est perdue afin de garantir que les flux TT ne seront pas retardés. Cette part de bande passante augmente les délais pour les flux RC.

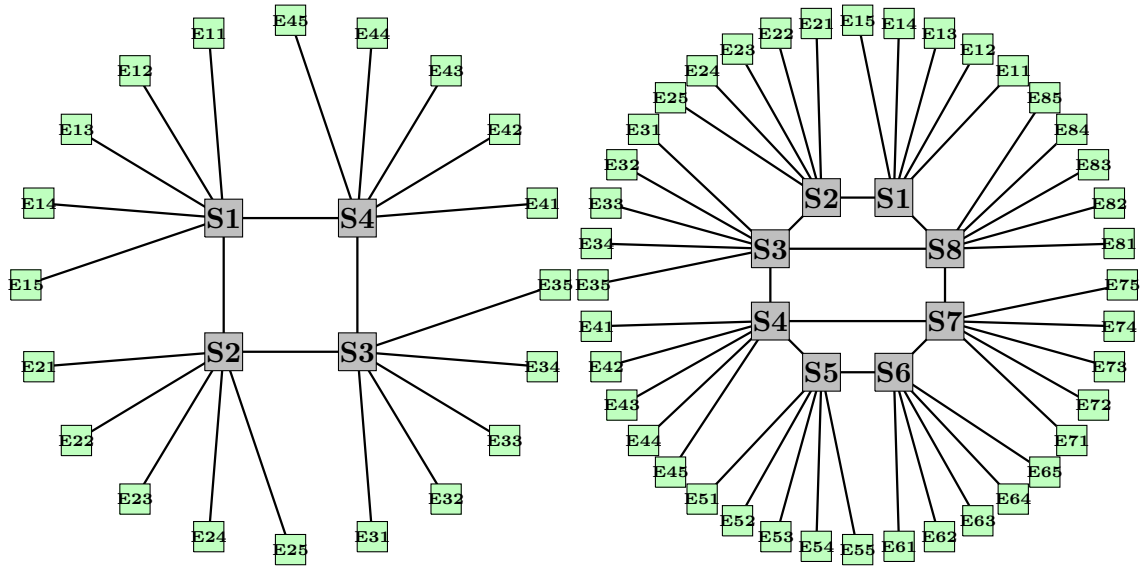
— **Réduction des contentions** : L'utilisation de flux TT réduit le nombre de contentions et diminue donc les délais des flux RC.

Nous prendrons en compte les deux politiques d'intégration qui ont été implémentées dans TTEthernet le *shuffling* et le *timely block*, la *preemption* n'étant présente que dans les premières versions.

7.3.1 Configurations étudiées

4S-200VL La première configuration étudiée est un réseau constitué de 20 stations reliées entre elles par quatre commutateurs. Chaque commutateur est relié à cinq stations. La vitesse des liens est de 100Mb/s et le délai des commutateurs est de 1.5μs. Le trafic est composé de 200VL.

8S-500VL La seconde configuration étudiée est un réseau constitué de 40 stations reliées entre elles par 8 commutateurs. Chaque commutateur est relié à cinq stations. La vitesse



(a) Configuration 4S-200VL. (b) Configuration 8S-500VL.

FIGURE 7.4 – Représentations des deux configurations étudiées.

des liens est de 100Mb/s et le délai des commutateurs est de 1.5μs. Le trafic est composé de 500VL.

Ces deux configurations sont illustrées Figure 7.4.

Pour chacune des configurations un nombre prédéterminé de VL (200 et 500) est généré en suivant cette procédure :

— La source est choisie uniformément parmi l'ensemble des stations.

— Le nombre de récepteurs est tiré aléatoirement selon une loi uniforme entre 1 et 5. Il y a donc en moyenne trois fois plus de flux que de VL. Chaque récepteur est ensuite choisi parmi l'ensemble des stations (privé de la station émettrice) de façon équiprobable.

— La taille maximale des trames de chaque VL est choisie uniformément sur un intervalle fixé au préalable : entre 64-340 octets pour la configuration 1 et entre 64-680 octets pour la configuration 2.

— Pour chaque VL le BAG est choisi parmi toutes les valeurs autorisées par la norme AFDX. Ce choix est pondéré pour que chaque valeur de BAG génère une quantité comparable de charge sur le réseau. Si un choix uniforme avait été fait l'ensemble des VL avec de grandes valeurs de BAG (128ms par exemple) aurait eu un impact négligeable en comparaison à l'ensemble des VL avec de petites valeurs de BAG (2ms).

Les distributions ainsi générées sont visibles sur la Figure 7.5. La charge sur les liens entre les commutateurs dans la configuration 4S-200VL est comprise entre 2.05% et 7.56%, la variation étant due à l'aléa du choix des émetteurs et destinataires. La charge sur les liens entre les commutateurs dans la configuration 8S-500VL est elle comprise entre 5.20% et 27.45%. Ces valeurs sont supérieures pour la seconde configuration car le nombre de VL

Trafic TT	**0%**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
A-Shuffling	1.27	1.22	1.17	1.06	0.97	0.86	0.80	0.64	0.52	0.43
A-Timely block	1.27	1.30	1.29	1.20	1.16	1.02	0.95	0.76	0.61	0.51
B-Shuffling	1.20	1.16	1.08	0.99	0.90	0.82	0.68	0.59	0.47	0.33
B-Timely block	1.20	1.25	1.21	1.13	1.05	0.97	0.82	0.70	0.54	0.37

TABLE 7.1 – **4S-500VL** : Moyennes des bornes en ms sur le délai des flux RC.

est plus important et la taille maximale des messages est presque le double de celle de la première configuration.

7.3.2 Augmentation de la part de trafic TT

L'objectif de ces expérimentations est d'évaluer l'impact du trafic TT sur le trafic RC. Pour cela nous allons progressivement faire varier la part de flux TT et de flux RC. L'ensemble des VL est divisé en 10 sous-ensembles de même taille (S1, S2, ..., S10). On désigne par "N0 TT" la situation où les VL appartenant à (S1, S2, ..., SN) sont TT et tous les autres VL sont RC. Ainsi "0 TT" désigne la situation où l'ensemble des flux sont RC et "100 TT" la situation où l'ensemble des flux sont TT. Il existe donc en tout 11 situations différentes.

Pour chaque configuration deux expériences sont étudiées :

— **Expérience A :** Pour chaque situation le routage de l'ensemble des flux et l'ordonnancement des flux TT est généré indépendamment de ceux des autres situations à l'aide de l'outil TTPlan de l'entreprise TTTech. Il est donc possible par exemple qu'un même flux n'ait pas la même route assignée dans la situation "0 TT" et "10 TT".

— **Expérience B :** L'ordonnancement et le routage de l'ensemble des flux sont générés dans la situation "100 TT" encore une fois en utilisant l'outil TTPlan de l'entreprise TTTech. Il est ensuite conservé pour toutes les autres situations. Si l'on considère un flux, il aura donc la même route assignée dans toutes les situations et dans toutes les situations où il sera TT il aura le même ordonnancement.

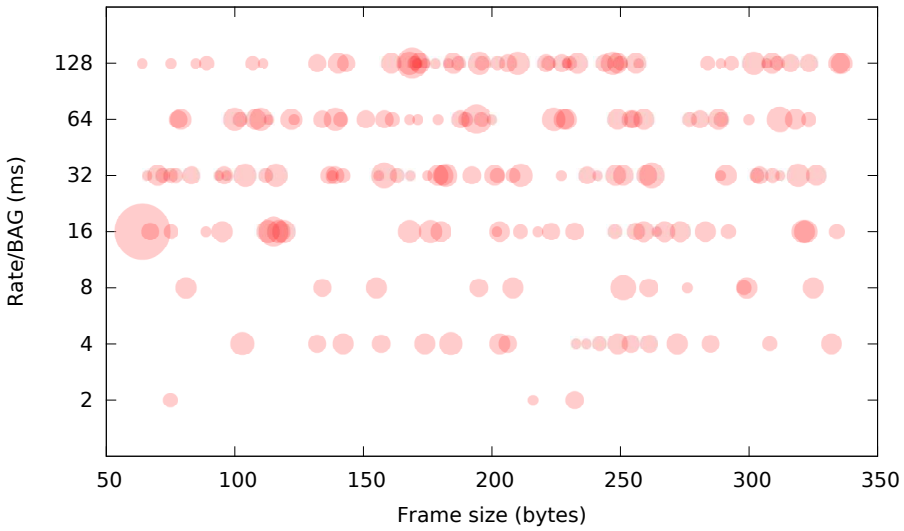
L'expérience B rend plus simple la comparaison entre les différentes situations puisqu'un minimum de paramètres est modifié d'une situation à l'autre mais elle n'est cependant pas optimale en termes de performance. En effet l'ordonnancement prévu lorsque l'ensemble des flux était TT n'est souvent pas le plus adapté lorsque seulement 10% des flux sont TT.

7.3.3 Résultats pour la configuration 1 : 4S-200VL

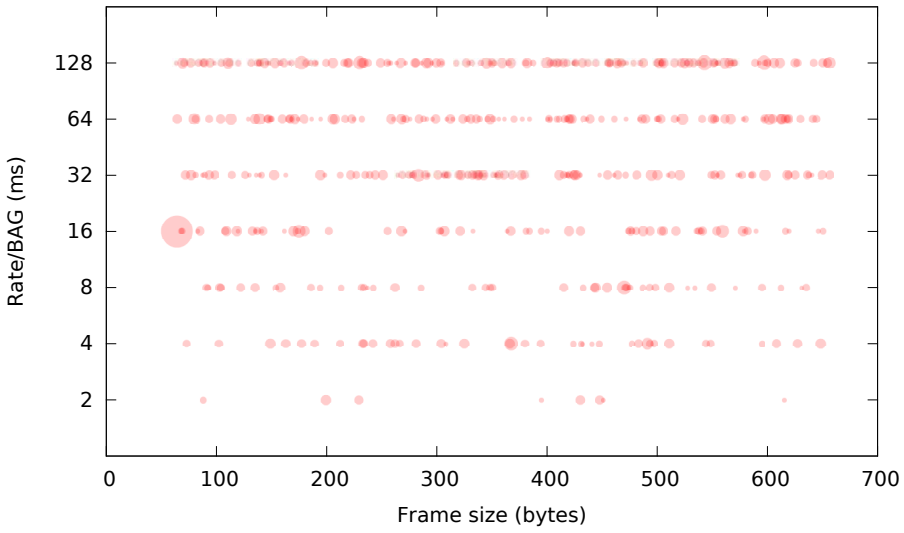
L'ensemble des résultats a été résumé dans la Table 7.1. On y retrouve la moyenne des bornes sur le temps de traversée des flux RC dans chacune des situations.

Expérience A

Dans un premier temps nous allons regarder plus en détail les résultats dans le cas de l'expérience A, c'est-à-dire en réaffectant une route et un nouvel ordonnancement à chaque



(a) Configuration 4S-200VL



(b) Configuration 8S-500VL

FIGURE 7.5 – Distribution de la taille et de la période des messages des deux configurations.

Trafic TT	**0%**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
A-Shuffling	6.07	5.84	5.35	5.16	4.50	4.21	3.72	3.27	2.63	1.83
A-Timely block	6.07	6.54	6.79	7.40	7.00	6.87	6.93	6.85	6.06	4.63
B-Shuffling	5.76	5.57	5.26	4.92	4.44	3.95	3.60	3.01	2.33	1.52
B-Timely block	5.76	6.33	6.57	6.94	6.69	6.82	6.60	5.99	5.35	4.40

TABLE 7.2 – **8S-500VL** : Moyennes des bornes en ms sur le délai des flux RC.

situation. Il y a 200VL, avec chacun entre 1 et 5 destinataires ce qui représente environ 600 flux. Pour chaque flux RC une borne sur le temps de traversée a été calculée, une partie des résultats est représentée Figure 7.6. Les flux ont été triés par bornes croissantes dans la situation “0 TT”.

Tout d’abord si l’on considère les bornes moyennes calculées dans chaque situation, on observe que les bornes sur le délais sont toujours inférieures en utilisant la politique *shuffling* par rapport à la politique *timely block*. Ce résultat était prévisible, en effet en cas de *timely block* une part de la bande passante n’est pas utilisée pour améliorer les garanties temporelles du trafic TT ce qui augmente le temps de traversée des flux RC.

De plus passer des flux de RC à TT permet de lisser le trafic et donc de diminuer le contentions ce qui finit par réduire le temps de traversée des flux RC. Dans le cas de *timely block* cet effet positif ne prend le dessus sur les effets négatifs qu’à partir de plus de 20% de trafic TT. Avant cela la borne sur le délai augmente.

Expérience B

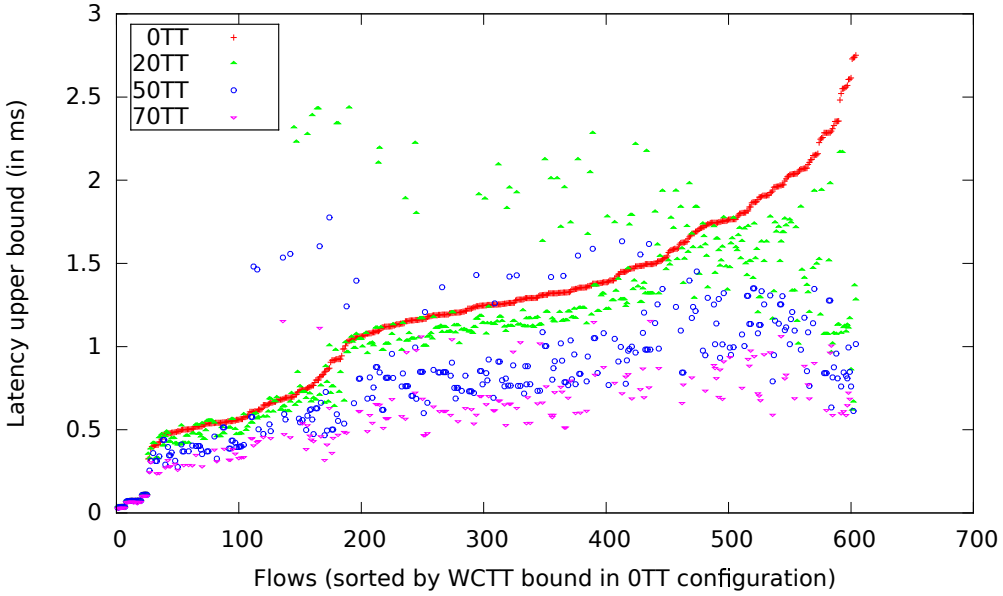
Nous allons maintenant regarder les résultats dans le cas de l’expérience B, c’est-à-dire en conservant le routage et l’ordonnancement de la situation “100 TT” pour toutes les situations. Une partie des résultats est représentée Figure 7.7.

Tout d’abord il est possible d’évaluer l’impact du routage et de l’ordonnancement sur les bornes en comparant à l’expérience A. Les bornes sur les délais sont inférieures dans la situation B (alors que c’est l’ordonnancement “100 TT” qui est utilisé), cela s’explique car l’ordonnancement et le routage sont faits pour favoriser les flux TT et se font donc au détriment des flux RC.

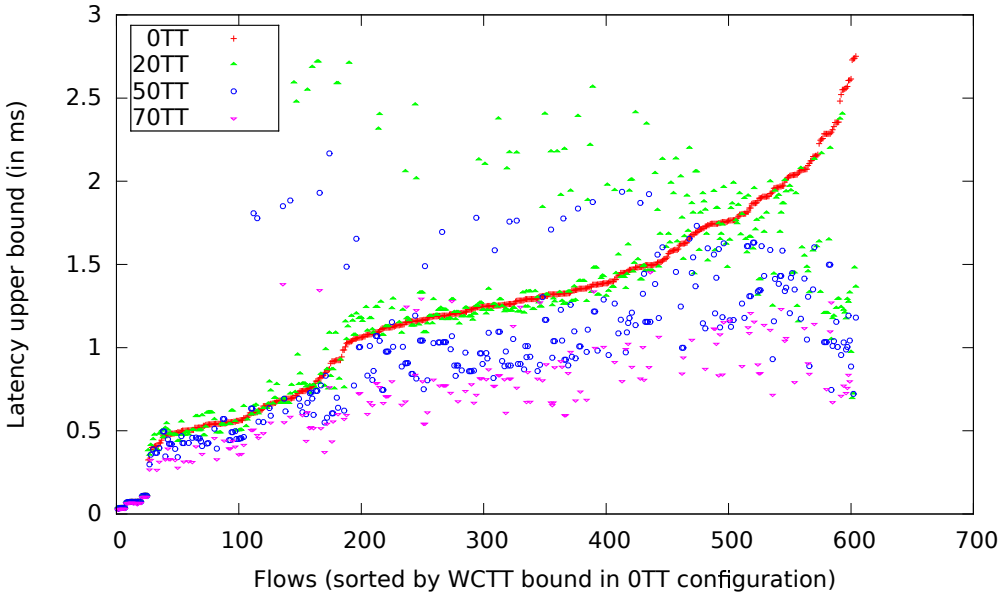
Encore une fois le *shuffling* est la meilleure politique de service en matière de délais maximaux en ce qui concerne les flux RC. Et comme précédemment en cas de *timely block* l’utilisation de flux TT ne s’avère avantageux en matière de délais maximaux pour les flux RC qu’à partir du moment où la part de flux TT dépasse les 20%. Avant cela passer une partie des flux de RC à TT a un impact négatif sur les délais maximaux des flux RC.

7.3.4 Résultats pour la configuration 2 : 8S-500VL

Nous allons maintenant étudier en détail la seconde configuration. L’ensemble des résultats a été résumé dans la Table 7.2. On y retrouve la moyenne des bornes sur le temps de traversée des flux RC dans chacune des situations. Cette configuration se différencie de la précédente non seulement par sa taille, il y a en effet deux fois plus de stations, mais aussi par la quantité de données transmises sur le réseau puisque le nombre de VL a été plus que doublé et la taille maximale des messages a été augmentée. Une des conséquence

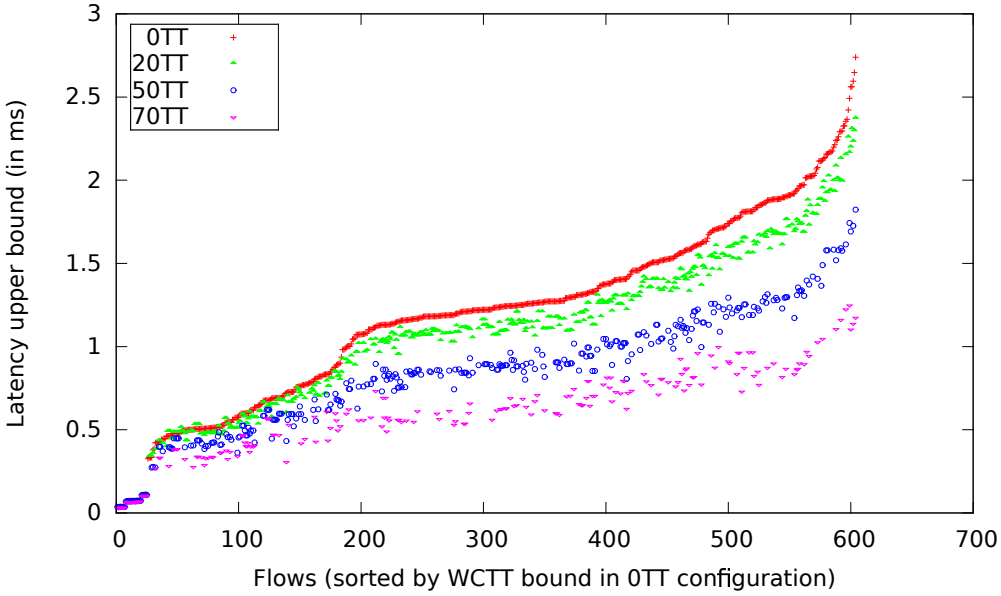


(a) Shuffling

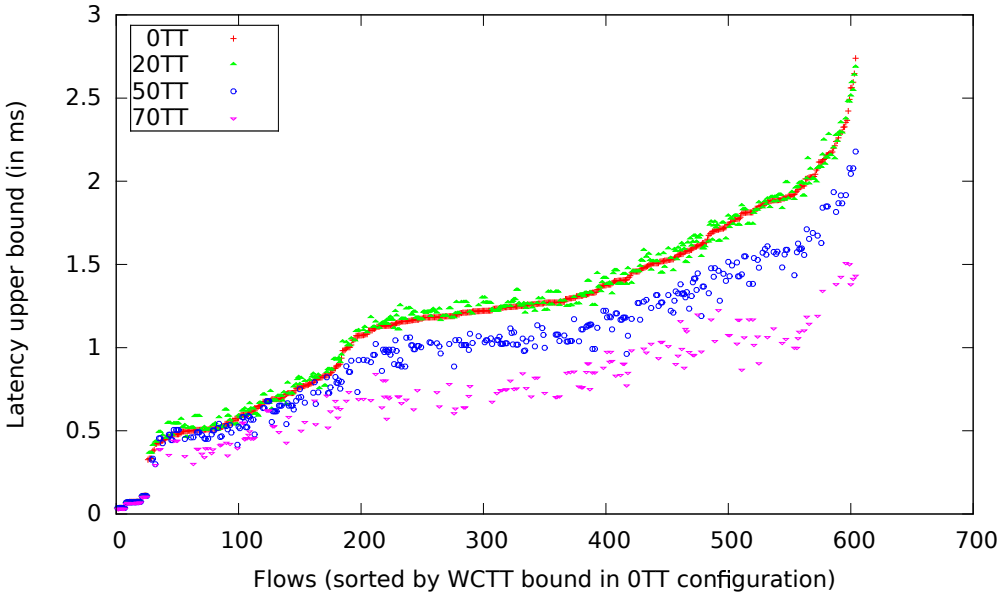


(b) Timely block

FIGURE 7.6 – **4S-200VL** Expérience A : bornes sur le délai pour chaque flux RC



(a) Shuffling



(b) Timely block

FIGURE 7.7 – **4S-200VL** Expérience B : bornes sur le délai pour chaque flux RC

est un réseau beaucoup plus chargé que dans la configuration 1 avec des charges sur les liens pouvant aller jusqu'à 27.5%. Comme nous venons de le préciser, il y a beaucoup plus de VL que précédemment et chaque VL ayant entre 1 et 5 destinataires nous obtenons environ 1500 VL.

Expérience A

Nous débuterons par l'expérience A, avec un nouveau routage et un nouvel ordonnancement dans chacune des situations donc. Les résultats de cette expérience sont présentés Figure 7.8.

Une première remarque est que le routage et l'ordonnancement des flux sont très différents d'une situation à l'autre. Les délais maximaux d'un flux donné sont donc très variables en fonction des situations. Ce qui explique cet aspect de "nuage de points".

Nous allons interpréter les résultats de la Table 7.2 dans le cas de l'expérience A. Si en cas de *shuffling* le fait de passer une partie du trafic de RC à TT est toujours bénéfique en moyenne en termes de bornes sur le délai des autres flux RC ce n'est pas vrai en cas de *timely block*. En effet comme nous l'avons expliqué au début de cette section passer un flux de RC à TT a plusieurs impacts sur le reste des flux RC. Il y a un impact positif en terme de délais, puisque cela permet de lisser le trafic mais il y a aussi un impact négatif dû au changement de priorité et à la politique de service. On peut voir que pour un réseau chargé comme celui de la configuration 2 c'est cette partie négative qui a le plus d'impact sur le délai moyen. Ainsi il faut atteindre les 80% en flux TT pour que le délai moyen pour les flux RC soit inférieur à celui observé lorsqu'il n'y avait pas de flux TT.

Expérience B

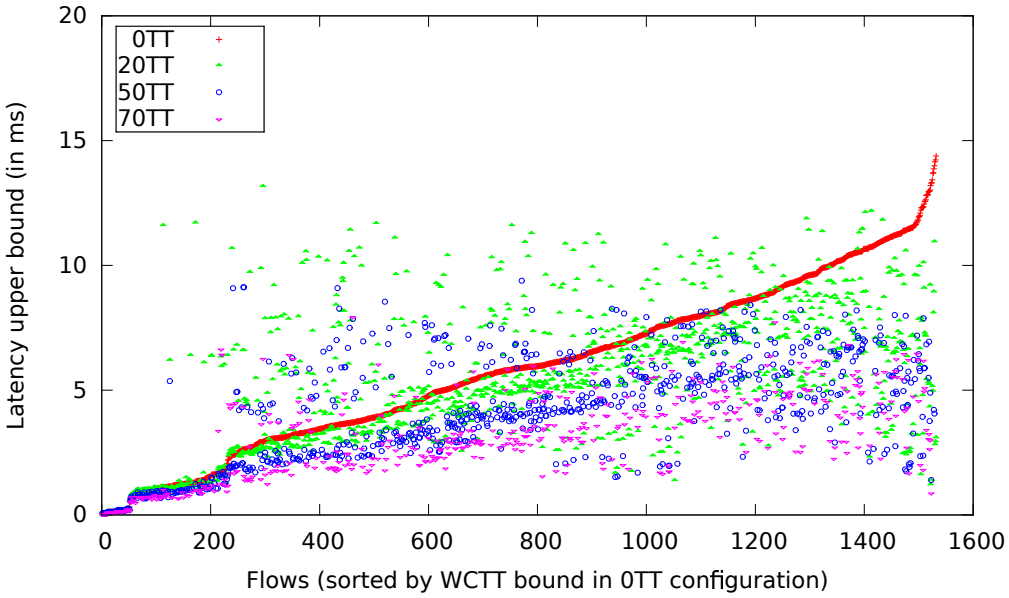
Dans le cas de l'expérience B, le routage et l'ordonnancement restant les mêmes, les variations sur les bornes de délai pour un flux donné d'un situation à l'autre sont beaucoup moins importantes. Sur la Figure 7.9 il est donc plus simple que précédemment d'observer les conséquences qu'ont la variation de la proportion de flux TT.

Les résultats restent les mêmes que précédemment en cas de *shuffling* le changement de priorité, qui tend à augmenter le délai pour les autres flux RC, est largement compensé par le gain apporté en utilisant des offsets et en lissant le trafic ce qui a pour conséquence qu'une augmentation de la proportion des flux TT diminue les bornes sur le délai pour les flux RC.

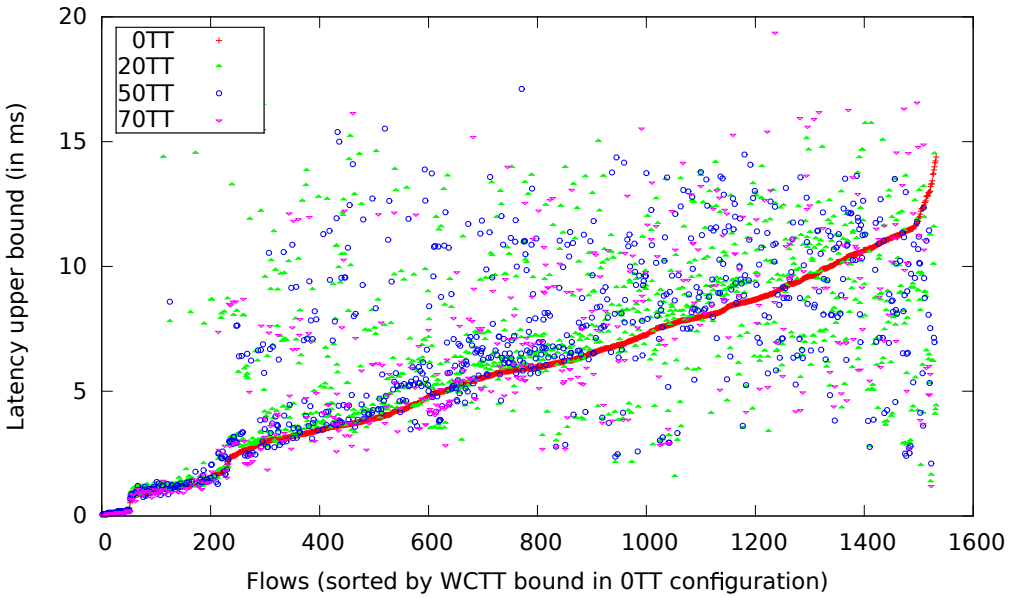
Au contraire en cas de *timely block* il faut qu'une part très importante du trafic soit TT avant de compenser l'augmentation du délai pour les flux RC.

7.4 Conclusion

Dans ce chapitre nous avons étudié une façon d'utiliser la synchronisation dans les réseaux embarqués. Le réseau TTEthernet avec ses trois classes de trafic offre énormément de flexibilité en terme d'organisation de la communication. Cependant les interactions qui existent entre ces différents trafics sont multiples et complexes et il est donc difficile de prévoir l'impact qu'elles peuvent avoir sur le temps de traversée du réseau. En particulier TTEthernet propose l'utilisation d'une classe *Time-Triggered* de priorité maximale. Cette

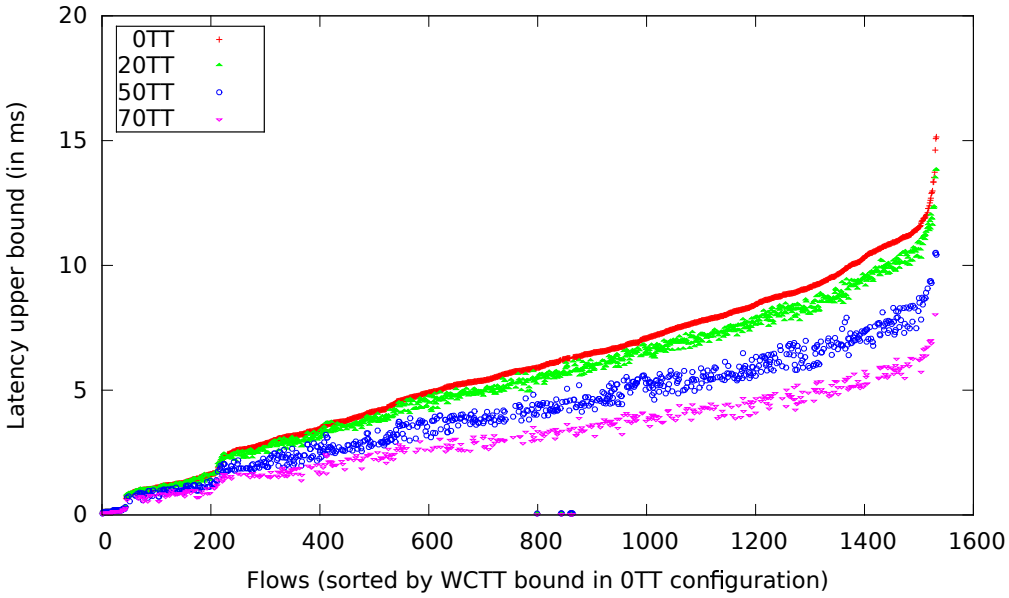


(a) Shuffling

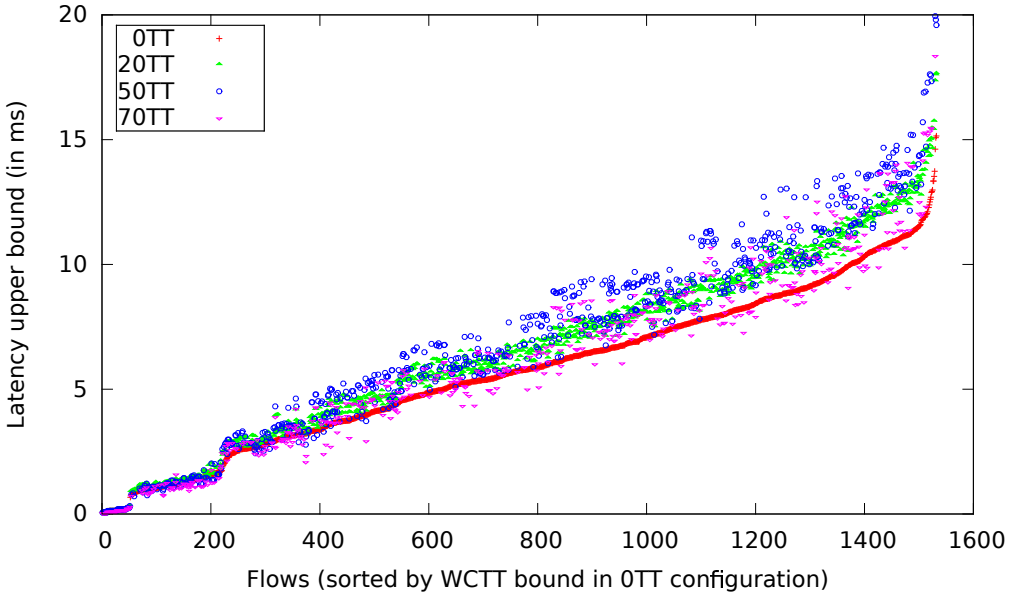


(b) Timely block

FIGURE 7.8 – **8S-500VL** Expérience A : bornes sur le délai pour chaque flux RC



(a) Shuffling



(b) Timely block

FIGURE 7.9 – **8S-500VL** Expérience B : bornes sur le délai pour chaque flux RC

classe a été imaginée pour les applications temps réel les plus contraignantes, l'utilisation d'offset permettant de contrôler le temps de traversée du réseau pour les messages TT. Cependant ce trafic TT a un impact direct sur le trafic *Rate-Constrained* proche du trafic utilisé dans l'AFDX. Dans un contexte temps réel il est nécessaire d'avoir des garanties temporelles sur les messages RC et il est donc important de connaître l'impact des flux TT sur les flux RC.

Pour répondre à ce problème, nous avons tout d'abord, à l'aide du calcul réseau, développé un nouveau modèle capable de calculer des bornes sur le temps de traversée des messages RC. Ce modèle permet de modéliser le trafic TT et peut être utilisé avec chacune des différentes politiques d'intégration proposées dans TTEthernet.

Une fois que nous avons été capables de borner le temps de traversée des messages RC nous avons imaginé un protocole expérimental pour évaluer l'impact qu'ont les flux TT sur le temps de traversée des flux RC. Pour cela nous avons imaginé deux configurations proches de celles utilisées en pratique par les industriels. Pour chacune de ces configurations nous avons fait varier la proportion de flux TT et RC et dans chaque cas nous avons calculé des bornes sur le temps de traversée des flux RC.

Nous avons pu apprendre de ces expérimentations un grand nombre de points :

— La politique d'intégration *timely block*, en prévenant les contentions pour les flux TT, entraîne une augmentation importante des bornes sur le temps de traversée en comparaison avec la politique de *shuffling*. Cela va jusqu'à doubler les bornes sur les délais lorsque la proportion de flux TT dépasse les 50%.

— La politique d'ordonnancement utilisée par TTPlan vise à lisser le trafic en répartissant les flux TT dans le temps. Ceci a pour conséquence de diminuer les bornes sur les délais des flux RC, lorsque la proportion de flux TT augmente. Cependant cet effet n'est vrai qu'en cas de *shuffling*, politique qui augmente le temps de traversée maximum des flux TT. Dans le cas de *timely block* il faut qu'une part importante du trafic soit TT pour que cet effet ne soit pas négligeable.

— La charge du réseau a un rôle majeur sur les interactions entre les flux TT et RC. En effet lorsque le nombre de flux est plus important les interactions entre flux TT et RC seront d'autant plus présentes. En cas de *timely block*, il y aura donc plus de bande passante non utilisée (afin de garantir les dates d'émission des messages TT) ce qui augmente les bornes sur les délais.

Un point que nous n'avons par contre pas étudié est la nécessité d'une synchronisation entre la partie application et la partie réseau pour les flux TT. En effet si une donnée est produite après la date d'émission du message TT elle devra attendre toute une période de ce flux alors qu'avec un flux RC elle devra uniquement attendre le délai de contention.

Chapitre 8

Modélisation en Calcul Réseau d'une architecture TSN possédant de multiples files Credit Based Shaper (TSN/CBS-BE)

8.1 TSN/CBS-BE : une extension du réseau AVB

TSN désigne un groupe de travail de l'IEEE, qui complète Ethernet, et les files d'attente Ethernet (802.1Q), afin de lui permettre de supporter des flux de données temps réel. Le mot ≪TSN≫ est aussi le nom utilisé pour un réseau implémentant ces nouvelles fonctionnalités.

Ce groupe est actuellement toujours en cours de travaux, la norme TSN n'est donc pas entièrement définie. De plus, l'architecture TSN admet beaucoup de paramètres, conduisant à de nombreux comportements différents. Dans la situation TSN la plus générique, chaque port de sortie possède 8 files d'attente.

Chaque file d'attente a une porte (ou *gate*) qui peut être dans l'état ouverte ou fermée. L'ensemble du système possède une période commune ou hyper-période et une table globale, la *Gate Control List* (GCL), qui définit l'état de chaque porte durant toute la durée de l'hyper-période.

Chaque file d'attente peut également posséder son propre algorithme de sélection de transmission, le plus ancien étant le *credit based shaper* (CBS) mais ce n'est pas le seul, d'autres algorithmes de sélection peuvent exister en aval de ces files.

Une file d'attente est dite *ready* si et seulement si elle n'est pas vide, si sa porte est ouverte, et si son algorithme de sélection de transmission permet l'émission d'un message. L'arbitrage entre les files d'attente *ready* est ensuite effectué en utilisant une politique de priorité statique.

L'architecture générale de TSN a été représentée Figure 8.1 (cette figure est issue de la norme [802.1Qbv, 2015]).

Ici nous allons chercher à modéliser un cas particulier de TSN que nous appelons TSN/CBS-BE où :

— Il existe n files, désignées CBS (ou AVB) qui utilisent l'algorithme *credit based shaper*.

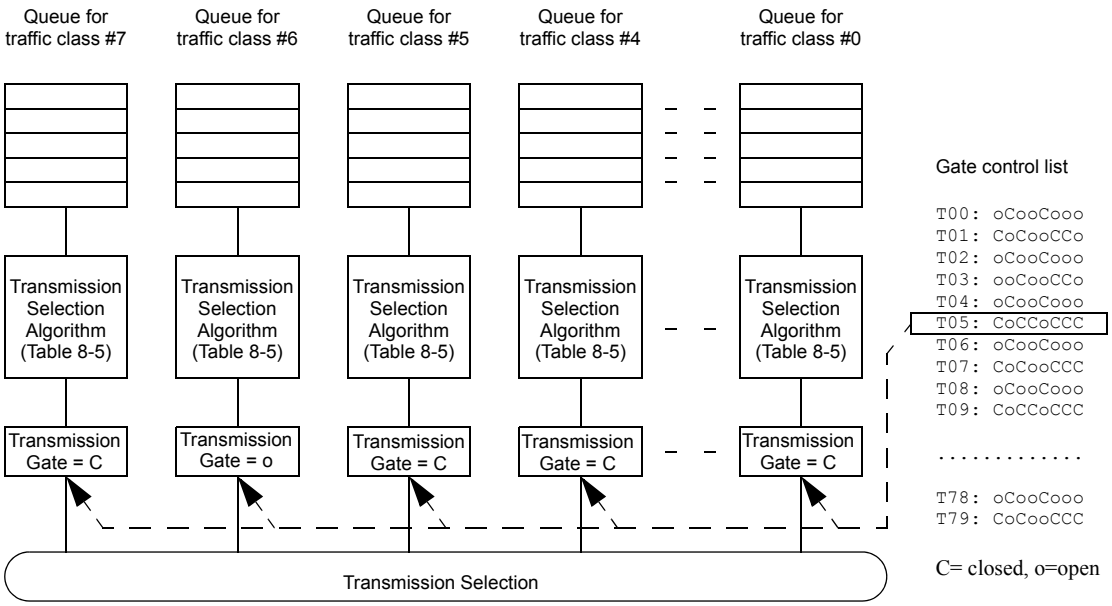


FIGURE 8.1 – Architecture TSN [802.1Qbv, 2015, Figure 8-12]

— Le reste des files sont des files *best effort* (BE) qui n'ont pas d'algorithme de sélection de transmission et sont moins prioritaires que les files CBS.

— La *Gate Control List* est telle que les portes de toutes les files sont toujours ouvertes.

Ce cas particulier est proche de la situation du réseau AVB (décrit au Chapitre 2) mais s'en différencie par la possibilité d'avoir plus de deux files utilisant l'algorithme CBS. Nous verrons par la suite que cette hypothèse est loin d'être anodine et augmente considérablement la difficulté du problème. Une représentation de ce cas est représentée Figure 8.2 avec trois files AVB et une file BE.

TSN est une solution innovante proposant énormément de flexibilité à l'utilisateur, dont la possibilité d'utiliser des flux TT, mais qui ne pourra être utilisée que s'il est possible de garantir que les contraintes temporelles sont respectées. Une modélisation en Calcul

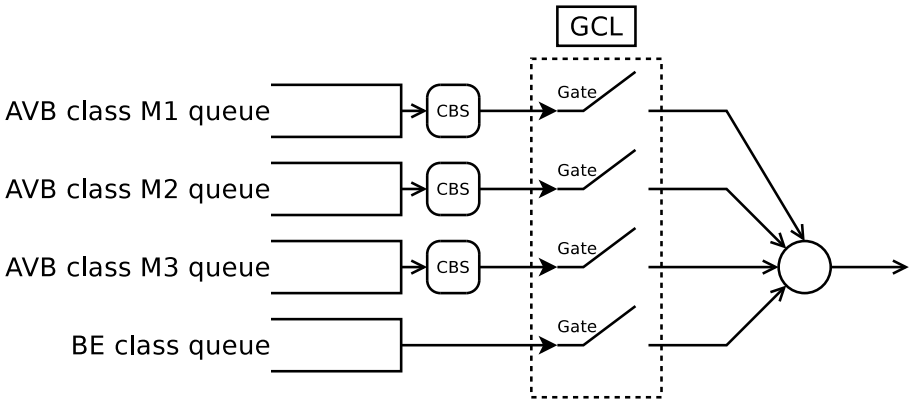


FIGURE 8.2 – Exemple d'architecture TSN/CBS-BE

Réseau du réseau AVB ayant déjà été réalisée avec succès [De Azua and Boyer, 2014] et grâce aux résultats concernant les flux TT que nous avons obtenus sur les réseaux CAN et TTEthernet (présentés dans les Chapitres 6 et 7) nous avons décidé de nous intéresser à TSN afin de proposer un modèle pouvant être utilisé pour borner le temps de traversée des messages.

Dans un premier temps, nous avons décidé de nous concentrer sur l'extension du *credit based shaper* à plus de deux files et de ne pas prendre en compte l'impact des flux TT. Ainsi dans ce chapitre, nous considérerons un cas particulier de TSN : TSN/CBS-BE. Le cas où un flux TT est présent et où les portes sont parfois ouvertes et parfois fermées est étudié dans le Chapitre 9.

8.1.1 Le principe de crédit dans TSN/CBS-BE

Comme nous l'avons vu précédemment, dans TSN/CBS-BE un certain nombre de files sont CBS. Ces files utilisent l'algorithme *credit based shaper*. Le but de cet algorithme est de permettre un meilleur partage du lien ; en effet lorsque la priorité statique est utilisée il y a un risque que les flux prioritaires s'accaparent toute la bande passante ou que la latence des autres files soit trop importante. On note l'ensemble de ces n files $Q_{CBS} = \{q_1, q_2, ..., q_n\}$. Chacune de ces files q_i est caractérisée par deux paramètres le *send slope* $sdSl_i < 0$ et le *idle slope* $idSl_i > 0$. De plus, à chaque file q_i est associé un crédit c_i initialisé à 0 et une porte qui peut être ouverte ou fermée.

Ces paramètres sont utilisés par le serveur TSN pour déterminer quel message doit être émis à un instant donné en respectant un certain nombre de règles :

R1 Lorsque le serveur est en attente, il sélectionne pour l'émission le message en tête de la file la plus prioritaire parmi les files *ready*. Une classe est *ready* lorsque :
- sa file est non vide,
- son crédit est positif ou nul dans le cas particulier des files CBS.

R2 Lorsqu'une classe CBS émet un message alors son crédit diminue à la vitesse $sdSl_i$.

R3 Lorsque la file d'une classe CBS est non vide mais n'est pas en train d'émettre alors son crédit augmente à la vitesse $idSl_i$.

R4 Lorsque la file d'une classe CBS est vide et si son crédit est positif alors il est remis à 0.

R5 Lorsque la file d'une classe CBS est vide et si son crédit est négatif alors son crédit augmente à la vitesse $idSl_i$.

L'ensemble de ces règles est illustré par la Figure 8.3 en reprenant l'exemple de la Figure 8.2 mais sans flux TT. Il y a donc trois files CBS : M1, M2 et M3. Nous allons ici nous intéresser à l'évolution du crédit de la classe M2 au cours du temps. On considère que le *send slope* est simplement l'opposé de l'*idle slope* pour la classe M2. Cette figure représente de haut en bas : les dates d'arrivée des trames, la sortie du lien et enfin la valeur du crédit de la classe M2 en fonction du temps.

À l'origine la valeur du crédit de la classe M2 est nulle, et sa file est vide. Jusqu'à l'arrivée du paquet M2-1 c'est **R4** qui s'applique le crédit reste nul. A l'arrivée de M2-1, M2 passe *ready* puisque sa file est non vide et son crédit est nul c'est donc **R1** qui s'applique. Les files des autres classes étant vides le paquet M2-1 est donc émis. Le crédit

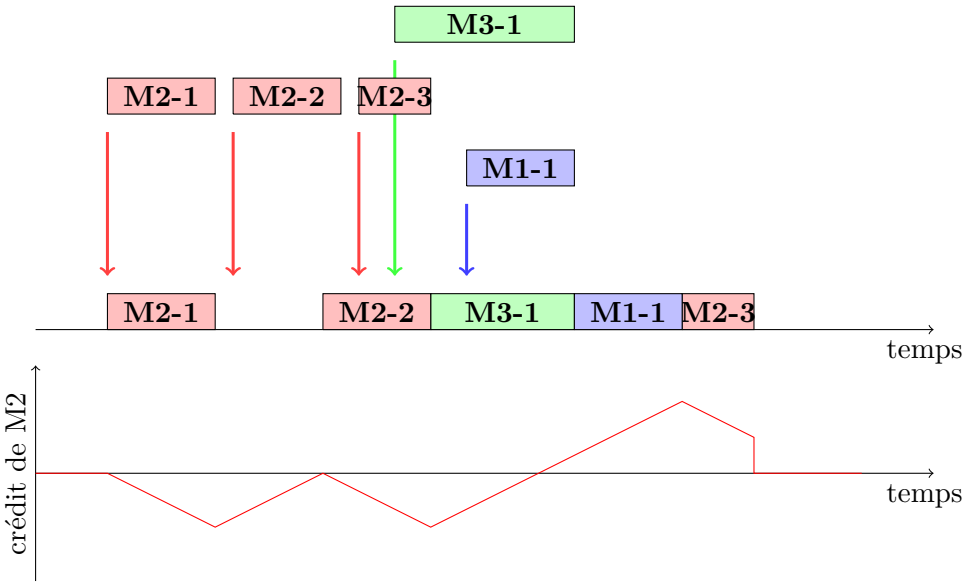


FIGURE 8.3 – Illustration de l'évolution du crédit

de la classe M2 diminue à la vitesse $sdSl_{M2}$ d'après **R2**. A la fin de l'émission de M2-1, la file de M2 est vide et son crédit est négatif, c'est **R5** qui s'applique, le crédit de la classe M2 augmente à la vitesse $idSl_{M2}$. Lorsque M2-2 arrive, le crédit de la classe M2 est toujours négatif donc c'est **R3** qui s'applique. Il est important de noter qu'il existe donc un intervalle de temps où il y a des paquets en attente mais pas d'émission sur le lien. Lorsque le crédit atteint 0 le paquet M2-2 peut être émis d'après **R1** le crédit évolue alors en en respectant **R2**. À la fin de l'émission de M2-2 il y a deux paquets en attente M2-3 et M3-1. M2 est plus prioritaire que M3 mais le crédit de M2 est négatif : la classe M2 n'est donc pas *ready* c'est donc M3-1 qui est émis (**R1**). Le crédit de M2 augmente (**R3**). Lorsqu'il devient nul l'émission de M3-1 n'est pas terminée, c'est donc **R3** qui s'applique, le crédit continue d'augmenter. Enfin lorsque M3-1 a terminé d'être émis c'est de nouveau **R1** qui permet de sélectionner le message qui va être émis. A ce moment M2 est *ready* mais ce n'est pas la classe la plus prioritaire qui l'est puisque M1 a un message en attente. C'est donc M1-1 qui est émis. Puis c'est enfin M2-3 qui peut être émis. A la fin de l'émission le crédit de la classe M2 est positif mais sa file est vide il est donc remis à 0 (**R4**).

Nous avons détaillé le fonctionnement de TSN/CBS-BE nous allons maintenant pouvoir nous intéresser à sa modélisation en Calcul Réseau. Mais avant cela il est nécessaire d'étudier les variations de crédit des classes CBS dans TSN/CBS-BE.

8.2 Borner le crédit dans TSN/CBS-BE

Dans la section précédente nous avons présenté les hypothèses ainsi que le fonctionnement du réseau TSN/CBS-BE. Nous avons en particulier évoqué le crédit des classes CBS.

Nous allons maintenant définir l'architecture de référence que nous considérerons par la suite :

Définition 26 (Architecture TSN/CBS-BE)**.** *Notre architecture TSN/CBS-BE de référence possède N_{CBS} files CBS, le reste des files sont BE. La vitesse de transmission du lien est noté C.*

Pour tout $1 \leq i \leq N_{CBS}$, on note $c_i(t)$ le crédit de la classe i à l'instant t.

De plus, on note L_i^{max} la taille maximale de n'importe quel message de la classe i et L_{BE}^{max} la taille maximale d'un message BE. Et l'on note $L_{>i}^{max} = \max\left(\max_{j\in[i+1,N_{CBS}]} L_j^{max}, L_{BE}^{max}\right)$, la taille maximale d'un message appartenant à une classe moins prioritaire que la classe i.

*Enfin, on note $idSl_i$ et $sdSl_i$ l'*idle slope *et le* send slope *de la classe i. La norme [802.1Qbv, 2015] donne la contrainte $sdSl_i = idSl_i - C$ mais cette contrainte est inutile pour certaines de nos preuves il sera donc précisé lorsque nous la considérons vraie.*

Dans cette section nous donnerons les conditions sur l'*idle slope* et le *send slope* qui permettent d'assurer que le crédit est borné ainsi que le calcul de ses bornes lorsqu'elles existent.

8.2.1 Cas particulier défini dans la norme

La norme [802.1Qbv, 2015] donne la contrainte $sdSl_i = idSl_i - C$. Dans cette partie (Section 8.2.1) nous nous placerons dans cette hypothèse. De plus nous sommes toujours dans l'hypothèse qu'il n'y a pas de flux TT et donc que les portes des flux CBS sont toujours ouvertes. Le cas avec flux TT sera étudié dans le Chapitre 9.

Dans un premier temps nous allons calculer une borne inférieure du crédit :

Théorème 18. *Dans les conditions de l'architecture de la Définition 26, le crédit $c_i(t)$ de la classe i peut être borné inférieurement par :*

$$c_i^{min} \triangleq \frac{L_i^{max}}{C} sdSl_i \leq c_i(t) \tag{8.1}$$

Démonstration. Le crédit d'une classe i diminue dans deux situations :

- s'il est positif et qu'il n'y a plus de message dans la file i, il est alors instantanément remis à 0,
- lorsque la classe i émet un message, il diminue alors à la vitesse $sdSl_i$.

De plus une classe ne peut commencer à émettre que si son crédit est positif ou nul.

Soit t un instant donné où la classe i émet un message, et notons s la date de début d'émission de ce message. Notons $c_i(t)$ le crédit de la classe i à l'instant t. Nous avons :

$$c_i(t) = c_i(s) + (t-s)sdSl_i.$$

Puisque une classe ne peut commencer à émettre que si son crédit est positif ou nul on a $c_i(s) \geq 0$. De plus, si L_i^{max} est la taille maximale de n'importe quel flux de la classe i alors $(t-s) \leq \frac{L_i^{max}}{C}$.

On en déduit :

$$c_i(t) = c_i(s) + (t-s)sdSl_i \geq \frac{L_i^{max}}{C} sdSl_i$$

□

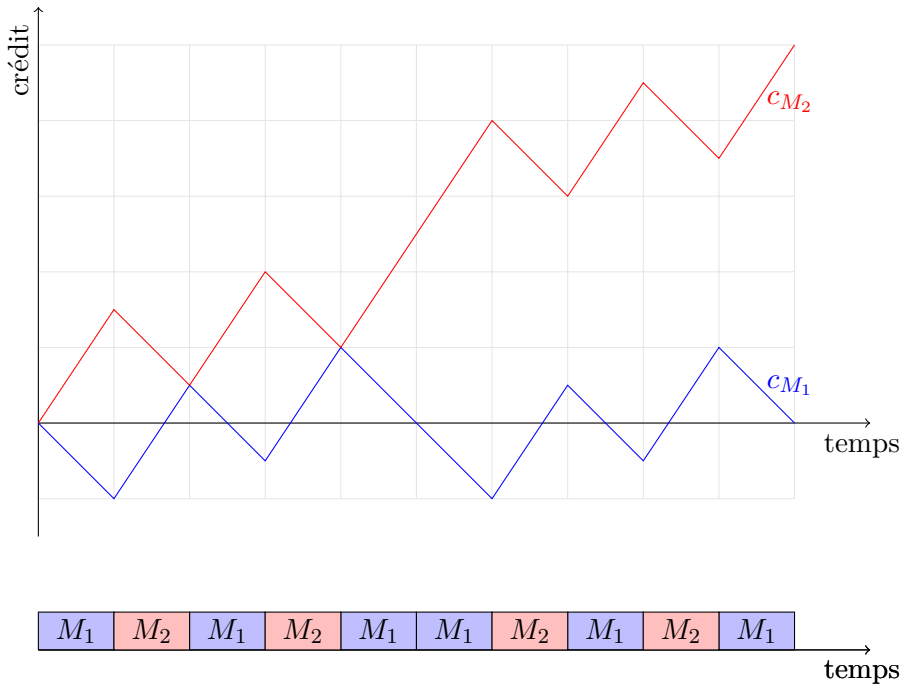


FIGURE 8.4 – Exemple de situation où un phénomène d'*overflow* se produit.

Nous allons maintenant donner une borne supérieure au crédit mais avant cela il nous faut trouver la condition qui garantisse qu'il n'y aura pas de phénomène d'*overflow*. C'est à dire que les crédits sont bien bornés. Cela n'est pas trivial en effet la condition $sdSl_i = idSl_i - C$ ne garantit pas qu'un tel phénomène ne se produira pas. Par exemple dans un cas très simple avec uniquement deux classes M_1 et M_2 telles que $idSl_1 = idSl_2 = 0,6C$ et $sdSl_1 = sdSl_2 = -0,4C$, on a bien la propriété $sdSl_i = idSl_i - C$. Cependant un exemple utilisant cette configuration a été illustré Figure 8.4. Dans cet exemple tous les paquets font la même taille, on fait l'hypothèse que les deux files ne sont jamais vides. Sur la Figure 8.4 nous avons représenté les paquets émis en respectant les règles énoncées dans la Section 8.1 et l'évolution du crédit qui en résulte pour chacune des deux classes. Or on observe rapidement que dans cette situation le crédit de la classe M_2 diverge et ne peut donc pas être borné. Le problème d'*overflow* est donc une question primordiale qui ne peut pas être ignorée. De plus si avec deux files la question de l'*overflow* reste assez intuitive elle devient beaucoup plus complexe lorsque l'on considère N files.

Théorème 19. *Dans les conditions de l'architecture de la Définition 26, si* $\forall i$, $sdSl_i = idSl_i - C$ *et si* $\sum_{i=1}^{N_{CBS}} idSl_i \leq C$, *alors le crédit de toute classe* i *est borné. De plus pour tout* $t \geq 0$, *une borne supérieure du crédit de la classe* i *est :*

$$c_i(t) \leq \frac{L^{max}_{\geq i}}{C} \sum_{j=1}^{i} idSl_j - \sum_{j=1}^{i-1} c_j^{min} \triangleq c_i^{max} \tag{8.2}$$

avec c_j^{min} *une borne inférieure du crédit de la classe* j *(par exemple celle du Théorème 18).*

Remarque Il est possible de donner une interprétation physique de la condition $\sum_{i=1}^{N_{CBS}} idSl_i \leq C$. En effet si l'on définit $\Phi_i = \frac{idSl_i}{C}$, on peut interpréter Φ_i comme une fraction de capacité offerte au flux i. La condition $\sum_{i=1}^{N_{CBS}} idSl_i \leq C$ revient à $\sum_{i=1}^{N_{CBS}} \Phi_i \leq 1$ soit un système sans sur-réservation.

Démonstration. Nous allons nous intéresser à la classe i et définir $c^H(t) = \sum_{j=1}^{i} c_j(t)$ la somme des crédits des classes de priorité plus haute ou égale à i. Nous allons montrer dans un premier temps que cette somme, sous certaines conditions, est bornée. Pour cela considérons un instant quelconque t.

De plus il nous faut définir deux ensembles $Q_{CBS}^{\leq i} = \bigcup_{1 \leq j \leq i} \{q_j\}$, l'ensemble des files avec une priorité supérieure ou égale à i, et $Q_{CBS}^{>i} = \bigcup_{i < j \leq N_{CBS}} \{q_j\} \cup Q_{BE}$ l'ensemble des files avec une priorité strictement inférieure à i.

Si toute les files de $Q_{CBS}^{\leq i}$ sont vides, on a alors $c^H(t) \leq 0$ (règles **R4** et **R5**).

Si aucun message n'est en cours d'émission à l'instant t, cela signifie que le crédit de chaque file CBS non vide est strictement négatif, ce qui implique entre autre $c^H(t) < 0$.

Si un message est en cours d'émission à l'instant t, alors notons $s < t$ la date de début d'émission de ce message.

Il y a deux possibilités, soit ce message appartient à $Q_{CBS}^{\leq i}$ soit il appartient à $Q_{CBS}^{>i}$.

Si le message appartient à $Q_{CBS}^{>i}$, alors soit $c^H(s) = 0$ et toutes les files de $Q_{CBS}^{\leq i}$ étaient vides à l'instant s soit $c^H(s) < 0$ (autrement, à l'instant s, c'est un message de $Q_{CBS}^{\leq i}$ qui aurait été choisi pour l'émission) donc dans toutes ces situations $c^H(s) \leq 0$.

De plus durant l'émission de ce message le crédit $c^H(t)$ a augmenté d'au plus :

$$c^H(t) - c^H(s) \leq \sum_{j=1}^{i} idSl_j(t-s).$$

Or par construction on sait qu'au plus un message de $Q_{CBS}^{>i}$ est émis entre s et t, si l'on note $L_{>i}^{max}$ la longueur maximale de ce message on obtient donc :

$$t - s \leq \frac{L_{>i}^{max}}{C}.$$

Puisque l'on a déjà prouvé que $c^H(s) \leq 0$ il est possible de déduire :

$$c^H(t) \leq \frac{L_{>i}^{max}}{C} \sum_{j=1}^{i} idSl_j.$$

Enfin si le message émis à l'instant t appartient à la classe $k \in Q_{CBS}^{\leq i}$, alors entre l'instant s et l'instant t le crédit de la classe k a diminué à la vitesse $sdSl_k$, alors que le crédit des autres classes $j \in Q_{CBS}^{\leq i}$ a lui augmenté d'au plus $idSl_j$. On en déduit donc pour le crédit c^H :

$$\begin{aligned} c^H(t) - c^H(s) &\leq sdSl_k(t-s) + \sum_{j=1, j \neq k}^{i} idSl_j(t-s) \\ &\leq \left(sdSl_k - idSl_k + \sum_{j=1}^{N_{CBS}} idSl_j - \sum_{j=i+1}^{N_{CBS}} idSl_j \right)(t-s). \end{aligned}$$

Puisque l'on sait que $sdSl_i = idSl_i - C$ alors si $\sum_{i=1}^{N_{CBS}} idSl_i \leq C$, on a :

$$c^H(t) - c^H(s) \leq -\sum_{j=i+1}^{N_{CBS}} idSl_j(t-s) \leq 0.$$

Pour résumer, à un instant t quelconque on a soit $c^H(t) \leq 0$ soit $c^H(t) \leq \frac{L^{max}_{>i}}{C} \sum_{j=1}^{i} idSl_j$ soit il existe un instant $s < t$ tel que $c^H(t) \leq c^H(s)$. Or le crédit ne pouvant augmenter que de façon continue et étant nul à $t = 0$ on peut en déduire :

$$c^H(t) \leq \frac{L^{max}_{>i}}{C} \sum_{j=1}^{i} idSl_j$$

En supposant que l'on connaît une borne inférieure, c'est-à-dire, $c_j(t) \geq c_j^{min}$, on obtient :

$$c_i(t) \leq \frac{L^{max}_{>i}}{C} \sum_{j=1}^{i} idSl_j - \sum_{j=1}^{i-1} c_j^{min}.$$

□

Corolaire 3. *Dans les conditions de l'architecture de la Définition 26, si* $\forall i$, $sdSl_i = idSl_i - C$ *et si* $\sum_{i=1}^{N_{CBS}} idSl_i \leq C$, *alors la somme des crédits* $c(t) \triangleq \sum_{j=1}^{N_{CBS}} c_j$ *est bornée par :*

$$c(t) \leq \frac{L^{max}_{BE}}{C} \sum_{j=1}^{N_{CBS}} idSl_j$$

Comment interpréter ce dernier résultat ? Il faut le comprendre ainsi : si $\forall i$, $sdSl_i = idSl_i - C$ et si $\sum_{i=1}^{N_{CBS}} idSl_i \leq C$ alors le crédit total diminue lorsqu'une trame CBS est émise. Le crédit total étant nul à $t = 0$ il est maximal après l'émission d'une trame BE si toutes les files CBS gagnent du crédit durant cette émission.

Dans cette section nous avons donc vu quelle était la condition pour garantir qu'il n'y aura pas d'*overflow* et nous avons calculé une borne inférieure et une borne supérieure pour le crédit de chaque classe i lorsque $sdSl_i = idSl_i - C$. Mais que se passe-t-il lorsque cette égalité n'est plus vraie ? C'est ce que nous allons voir dans la section suivante.

8.2.2 Condition de *non-overflow* dans le cas général sans TT

Remarque Je tiens tout d'abord à remercier tout particulièrement David Levadoux qui m'a aidé à rédiger la démonstration de la condition de non-*overflow* dans le cas général.

Dans cette section nous avons décidé de ne plus prendre en compte l'égalité $sdSl_i = idSl_i - C$ contrainte par la norme mais au contraire de nous placer dans une situation plus générale en considérant $sdSl_i$ et $idSl_i$ indépendants l'un de l'autre. Avec cependant toujours $idSl_i > 0$ et $sdSl_i < 0$. Dans ce cas, la condition de *non-overflow* (Théorème 19) n'est plus correcte et une nouvelle condition doit être définie.

Cette partie est théorique puisque l'égalité $sdSl_i = idSl_i - C$ est contrainte par la norme. De plus la condition de *non-overflow* du Théorème 19 avait une interprétation

"simple", la condition de *non-overflow* dans le cas général est plus fine mais a un interprétation plus difficile.

Afin de déterminer cette condition de *non-overflow* nous ne considérerons pas la règle **R5** de la Section 8.1, c'est-à-dire que nous considérerons que la variation du crédit de chaque classe est continue. Une façon de voir les choses et de considérer que les files d'attente de toutes les classes ne sont jamais vides. Cela ne réduit en rien le résultat puisque **R5** induit une diminution du crédit et que c'est la borne supérieure qui pose problème.

De plus nous ne prendrons pas en compte les flux BE. En effet lorsqu'un message BE est émis cela signifie que le crédit de chacune des classes est négatif. Les flux BE ne peuvent pas permettre au crédit d'une classe i de dépasser $\frac{L_{BE}^{max}}{C} idSl_i$ et ne sont donc pas pertinents lorsque l'on recherche une condition de *non-overflow*. Une façon d'interpréter ce point et de se représenter le flux BE comme un "effet de bord", lorsque l'on parle de *non-overflow* on veut en fait montrer que le crédit d'une classe ne peut pas diverger comme c'était le cas dans l'exemple de la Figure 8.4.

Pour définir cette relation il va nous falloir définir un ensemble de nouvelles notations. Définissons $(\mathbf{e_i})_{i=1,..,n}$ la base canonique de $\mathbb{R}^n$, $\mathbf{f_i}(0)$ un vecteur de $\mathbb{R}^n$ dont les coordonnées sont égales à 1 excepté la i-ième égale à 0 puis $\mathbf{f_i}(a) = \mathbf{f_i}(0) - a\mathbf{e_i}$ et $a_i = \frac{-sdSl_i}{idSl_i}$ et pour finir $\mathbf{a} = (a_1, a_2, ..., a_n)$.

$\mathbf{F}(\mathbf{a})$ est la matrice de $\mathbb{R}^{n\times n}$ faite des vecteurs $\mathbf{f_i}(a_i)$.

$$\mathbf{F}(\mathbf{a}) = \begin{pmatrix} -a_1 & 1 & ... & 1 & 1 \\ 1 & -a_2 & ... & 1 & 1 \\ ... & ... & ... & ... & ... \\ 1 & 1 & ... & -a_{n-1} & 1 \\ 1 & 1 & ... & 1 & -a_n \end{pmatrix}$$

Lemme 2. *Le problème de la condition de non-*overflow *peut s'exprimer comme :*

$$\forall x \in \mathbb{R}_+^{n*}, \mathbf{F}(\mathbf{a}).\mathbf{x} \notin \mathbb{R}_+^{n*} \tag{8.3}$$

Démonstration. Soit $\triangle t$ un intervalle de temps décomposé en $\triangle t = \sum_{j=1}^{n} \triangle t_j$, où $\triangle t_j$ représente le temps total d'émission de message de la classe j. On définit alors $\mathbf{x}$ par :

$$\mathbf{x} = \begin{pmatrix} \triangle t_1 \\ \triangle t_2 \\ ... \\ \triangle t_n \end{pmatrix}$$

Alors $\mathbf{F}(\mathbf{a}).\mathbf{x}$ est le vecteur de variation de crédit pondéré de chacune des classes durant une durée $\triangle t$ (où pour rappel aucun flux BE n'est échangé) et avec $\triangle t_j$ qui représente le temps total d'émission de message de la classe j. En effet :

$$\mathbf{F}(\mathbf{a}).\mathbf{x} = \begin{pmatrix} -a_1\triangle t_1 + (\triangle t - \triangle t_1) \\ -a_1\triangle t_1 + (\triangle t - \triangle t_2) \\ ... \\ -a_1\triangle t_1 + (\triangle t - \triangle t_n) \end{pmatrix} = \begin{pmatrix} \frac{1}{idSl_1}\left(sdSL_1\triangle t_1 + idSl_1(\triangle t - \triangle t_1)\right) \\ \frac{1}{idSl_2}\left(sdSL_2\triangle t_1 + idSl_2(\triangle t - \triangle t_2)\right) \\ ... \\ \frac{1}{idSl_n}\left(sdSL_n\triangle t_1 + idSl_n(\triangle t - \triangle t_n)\right) \end{pmatrix}$$

Donc :

$$\mathbf{F}(\mathbf{a}).\mathbf{x} = \begin{pmatrix} \frac{c_1(t+\triangle t)-c_1(t)}{idSl_1} \\ \frac{c_2(t+\triangle t)-c_2(t)}{idSl_2} \\ \dots \\ \frac{c_n(t+\triangle t)-c_n(t)}{idSl_n} \end{pmatrix}$$

Puisque $idSl_i > 0$, prouver l'équation (8.3) revient à prouver que quelle que soit la distribution d'émission de message CBS, il y aura toujours au moins une classe qui perdra du crédit (ou qu'aucune classe n'en gagnera), c'est-à-dire :

$$\begin{cases} \exists i,\ 1 \leq i \leq n | c_i(t+\triangle t) - c_i(t) < 0 \\ \text{ou} \\ \forall i,\ 1 \leq i \leq n | c_i(t+\triangle t) - c_i(t) \leq 0 \end{cases}$$

Si aucune classe ne gagne de crédit alors le crédit de chaque classe est borné supérieurement.

Si une classe gagne du crédit, alors au moins une classe en perd. Donc si en augmentant $\triangle t$ le crédit de la première diverge vers $+\infty$ alors le crédit de la seconde diverge vers $-\infty$. Or comme le crédit de chaque classe peut être borné inférieurement (Théorème 18), alors il est aussi possible de borner supérieurement le crédit de chaque classe. □

Avant de poursuivre il faut énoncer deux propriétés mathématiques qui nous seront nécessaires dans la suite.

Propriété 5.

$$(-1)^{n+1} \det(\mathbf{F}(\mathbf{0})) > 0 \tag{8.4}$$

$$\exists \mathbf{x} \in \mathbb{R}_+^{n*} | \mathbf{F}(\mathbf{b}).\mathbf{x} \in \mathbb{R}_+^{n*} \cup \mathbb{R}_-^{n*} \Leftrightarrow \det(\mathbf{F}(\mathbf{b})) \neq 0 \tag{8.5}$$

Avec $\mathbf{b} = (b_1, b_2, ..., b_n) \in \mathbb{R}_+^n$ *un vecteur quelconque.*

Remarques :

(8.4) Cette propriété est une conséquence de la forme très particulière de $\mathbf{F}(\mathbf{0})$. Elle permet de connaître le signe de la matrice $\mathbf{F}(\mathbf{0})$ en fonction de sa taille. Par exemple en dimension n= 2 cela donne $det(\mathbf{F}(\mathbf{0}))(-1)^3 > 0$ donc le déterminant de $\mathbf{F}(\mathbf{0})$ est négatif. $\mathbf{F}(\mathbf{0}) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, on peut calculer son déterminant qui vaut -1 qui est bien négatif.

(8.5) Dans la suite nous considérons le cas général à n dimensions que nous avons illustré Figure 8.5 dans le cas où n=2. Cette propriété découle une fois de plus de la forme particulière de $\mathbf{F}(\mathbf{b})$. Cette matrice est composée des n vecteurs $\mathbf{f_i}(b_i)$. En partant de l'origine ils permettent de définir n points. Sur la Figure 8.5 nous avons représenté ces vecteurs en bleu et rouge. Ces n points permettent de définir un hyperplan $\mathcal{P}$ de dimension n-1. Cet hyperplan est une droite en dimension 2, elle est représentée en vert sur la Figure. Si cet hyperplan passe par l'origine cela signifie que $\mathbf{F}(\mathbf{b})$ n'est pas une base de l'espace et donc que son déterminant est nul. Toute combinaison des $\mathbf{f_i}(b_i)$ appartient alors à $\mathcal{P}$. Dans le cas inverse une combinaison positive des

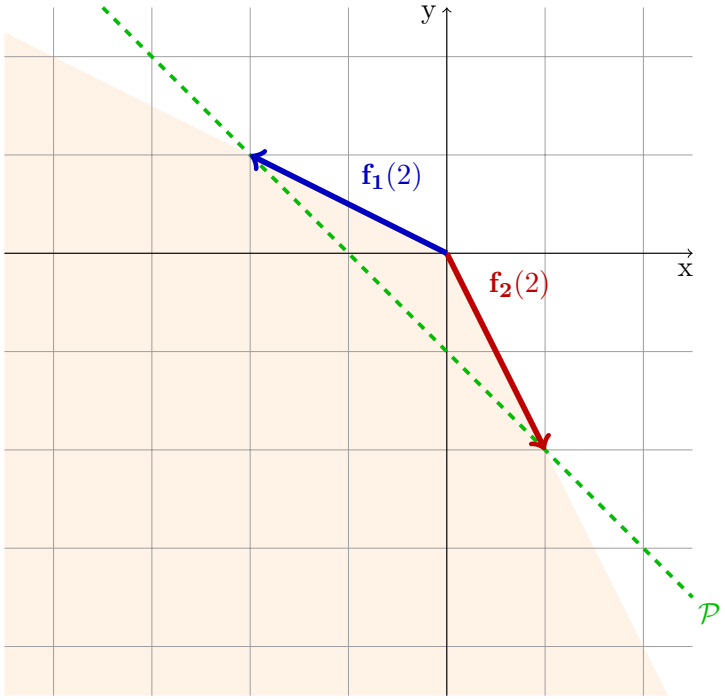


FIGURE 8.5 – Représentation de l'espace généré par $\mathbf{F}(\mathbf{b})$ en dimension 2.

$\mathbf{f_i}(b_i)$ permet d'obtenir un sous espace de dimension n. Ce dernier a été représenté en orange dans notre exemple. La propriété (8.4) permet d'affirmer qu'au moins un des points de cet ensemble appartient à $\mathbb{R}_+^{n*} \cup \mathbb{R}_-^{n*}$.

A l'aide de ces deux propriétés il nous est maintenant possible d'exprimer la condition de non-*overflow*.

Théorème 20. *Condition de* non-overflow *:*

$$\forall \mathbf{x} \in \mathbb{R}_+^{n*}, \mathbf{F}(\mathbf{a}).\mathbf{x} \notin \mathbb{R}_+^{n*} \Leftrightarrow (-1)^{n+1} \det(\mathbf{F}(\mathbf{a})) \leq 0 \tag{8.6}$$

Démonstration. Pour prouver cette équivalence on prouvera l'implication dans un sens puis dans l'autre.

— Nous allons tout d'abord prouver que :

$$\exists \mathbf{x} \in \mathbb{R}_+^{n*} | \mathbf{F}(\mathbf{a}).\mathbf{x} \in \mathbb{R}_+^{n*} \Rightarrow \forall t \in [0,1], \det(\mathbf{F}(t\mathbf{a}) \neq 0 \tag{8.7}$$

Si $\exists \mathbf{x} \in \mathbb{R}_+^{n*} | \mathbf{F}(\mathbf{a}).\mathbf{x} \in \mathbb{R}_+^{n*}$ alors pour tout i, $(\mathbf{f_i}(0) - a_i\mathbf{e_i}).\mathbf{x} \geq 0$ (avec au moins une de ces inégalité stricte).

On en déduit $\mathbf{f_i}(0).\mathbf{x} \geq a_i\mathbf{e_i}.\mathbf{x}$, or puisque $\mathbf{e_i}.\mathbf{x} \geq 0$ on a aussi $\mathbf{f_i}(0).\mathbf{x} \geq ta_i\mathbf{e_i}.\mathbf{x}$ pour tout $t \in [0,1]$. Donc on a enfin $\mathbf{f_i}(0).\mathbf{x} - ta_i\mathbf{e_i}.\mathbf{x} \geq 0$.

Donc $\mathbf{F}(t\mathbf{a}).\mathbf{x} \in \mathbb{R}_+^{n*}$ or d'après l'équation (8.5) nous pouvons conclure que $\det(\mathbf{F}(t\mathbf{a})) \neq 0$. On vient donc bien de prouver (8.7).

Nous allons maintenant prouver que :

$$\exists \mathbf{x} \in \mathbb{R}_+^{n*} | \mathbf{F}(\mathbf{a}).\mathbf{x} \in \mathbb{R}_+^{n*} \Rightarrow (-1)^{n+1} \det(\mathbf{F}(\mathbf{a})) > 0 \tag{8.8}$$

En utilisant (8.7) et sachant que $\det(\mathbf{F}(t\mathbf{a})$ est une fonction continue on peut déduire que $\det(\mathbf{F}(\mathbf{0}))$ et $\det(\mathbf{F}(\mathbf{a}))$ ont le même signe. Or le signe de $\det(\mathbf{F}(\mathbf{0}))$ est connue d'après l'équation (8.4), d'où $(-1)^{n+1} \det(\mathbf{F}(\mathbf{a})) > 0$. On vient donc bien de prouver (8.8).
Il est donc possible d'en déduire la contraposée :

$$(-1)^{n+1} \det(\mathbf{F}(\mathbf{a})) \leq 0 \Rightarrow \forall \mathbf{x} \in \mathbb{R}_+^{n*}, \mathbf{F}(\mathbf{a}).\mathbf{x} \notin \mathbb{R}_+^{n*} \tag{8.9}$$

— Nous allons tout d'abord prouver que :

$$\exists \mathbf{x} \in \mathbb{R}_+^{n*} | \mathbf{F}(\mathbf{a}).\mathbf{x} \in \mathbb{R}_-^{n*} \Rightarrow \forall t > 1, \det(\mathbf{F}(t\mathbf{a}) \neq 0 \tag{8.10}$$

Si $\exists \mathbf{x} \in \mathbb{R}_+^{n*} | \mathbf{F}(\mathbf{a}).\mathbf{x} \in \mathbb{R}_-^{n*}$ alors pour tout i, $(\mathbf{f_i}(0) - a_i \mathbf{e_i}).\mathbf{x} \leq 0$ (avec au moins une de ces inégalité stricte).
On en déduit $\mathbf{f_i}(0).\mathbf{x} \leq a_i \mathbf{e_i}.\mathbf{x}$, or puisque $\mathbf{e_i}.\mathbf{x} \geq 0$ on a aussi $\mathbf{f_i}(0).\mathbf{x} \leq t a_i \mathbf{e_i}.\mathbf{x}$ pour tout $t > 1$. Donc on a enfin $\mathbf{f_i}(0).\mathbf{x} - t a_i \mathbf{e_i}.\mathbf{x} \leq 0$.
Donc $\mathbf{F}(t\mathbf{a}).\mathbf{x} \in \mathbb{R}_-^{n*}$ or d'après l'équation (8.5) nous pouvons conclure que $\det(\mathbf{F}(t\mathbf{a})) \neq 0$. On vient donc bien de prouver (8.10).
Nous allons maintenant prouver que :

$$\exists \mathbf{x} \in \mathbb{R}_+^{n*} | \mathbf{F}(\mathbf{a}).\mathbf{x} \in \mathbb{R}_-^{n*} \Rightarrow (-1)^{n+1} \det(\mathbf{F}(\mathbf{a})) < 0 \tag{8.11}$$

Lorsque t tend vers l'infini on a $\lim_{t \to \infty} \frac{\mathbf{f_i}(t a_i)}{||\mathbf{f_i}(t a_i)||} = -\mathbf{e_i}$ et donc lorsqu'on considère la matrice $\det(\mathbf{F}(t\mathbf{a}))$ on obtient $\lim_{t \to \infty} \frac{\mathbf{F}(t\mathbf{a})}{||\mathbf{F}(t\mathbf{a})||} = -\mathbf{I_n}$ ($\mathbf{I_n}$ est la matrice identité de taille n).
Compte tenu de la continuité de $\det(\mathbf{F}(t\mathbf{a}))$ et en utilisant (8.10) nous pouvons en déduire que $\det(\mathbf{F}(\mathbf{a}))$ et $\det(-\mathbf{I_n}) = (-1)^n$ ont le même signe. Il est donc possible d'en déduire l'implication (8.11).
De plus d'après (8.5) on peut déduire que :

$$\forall \mathbf{x} \in \mathbb{R}_+^{n*}, \mathbf{F}(\mathbf{a}).\mathbf{x} \notin \mathbb{R}_+^{n*} \Rightarrow \begin{cases} \exists \mathbf{x} \in \mathbb{R}_+^{n*} | \mathbf{F}(\mathbf{a}).\mathbf{x} \in \mathbb{R}_-^{n*} \\ \text{ou} \\ \det(\mathbf{F}(\mathbf{a})) = 0 \end{cases} \tag{8.12}$$

On peut donc conclure en utilisant (8.12) et (8.11) :

$$\forall \mathbf{x} \in \mathbb{R}_+^{n*}, \mathbf{F}(\mathbf{a}).\mathbf{x} \notin \mathbb{R}_+^{n*} \Rightarrow (-1)^{n+1} \det(\mathbf{F_a}) \leq 0 \tag{8.13}$$

□

Nous avons donc exprimé une condition sur l'ensemble des $idlS_i$ et $sdSl_i$ qui permet de garantir que nous sommes bien dans une situation de *non-overflow* dans le cas général.

8.3 Modélisation de TSN/CBS-BE en Calcul Réseau

Dans cette section nous allons nous concentrer sur les courbes de service résiduel des classes CBS. Nous ne détaillerons pas les courbes d'arrivée qui sont les mêmes que celles présentes dans AVB et ont déjà été présentées dans [De Azua and Boyer, 2014].

Les courbes de services présentées ici sont une généralisation de celles présentées dans [De Azua and Boyer, 2014]. Celles ci ne prennent pas en compte la présence de flux TT (cela sera fait Chapitre 9) mais prennent en compte la possibilité qu'il y ait plus de deux classes CBS comme c'est le cas dans AVB.

Dans la suite nous nous intéresserons à la classe CBS i et calculerons ses différentes courbes de service. De plus dans cette section nous faisons l'hypothèse que le crédit de chaque classe peut être borné, et qu'une borne inférieure et une borne supérieure sont connues (on peut par exemple utiliser celles des Théorèmes 18 et 19).

8.3.1 Courbe de service simple

Théorème 21. *Dans les conditions de l'architecture de la Définition 26, si c_i^{max} est une borne supérieure du crédit de la classe i, alors une courbe de service simple de la classe CBS i ($i \in [1, N_{CBS}]$) est :*

$$\beta_i^{\min}(t) = \frac{C\ idSl_i}{idSl_i - sdSl_i}\left[t - \frac{c_i^{max}}{idSl_i}\right]^+. \tag{8.14}$$

Dans le cas où $sdSl_i = idSl_i - C$:

$$\beta_i^{\min}(t) = idSl_i\left[t - \frac{c_i^{max}}{idSl_i}\right]^+. \tag{8.15}$$

Démonstration. Soit t un instant quelconque. Notons $A_i(t)$ et $D_i(t)$ le flux d'arrivée et de départ de la classe CBS i. Soit s la date de début de la dernière *busy period* de la file i où le crédit de la file i était nul, c'est-à-dire pour tout x dans $(s, t]$, soit $c_i(x) < 0$ soit $D_i(x) < A_i(x)$.

A l'instant s, $D_i(s) = A_i(s)$ et $c_i(s) = 0$.

Notons $\triangle t$ l'intervalle $t - s$. Cet intervalle peut être décomposé en $\triangle t = \triangle t^+ + \triangle t^-$. $\triangle t^-$ représente la durée de transmission des messages de la classe CBS i. Durant $\triangle t^-$ le crédit de la classe i diminue à la vitesse $sdSl_i$. Enfin $\triangle t^+$ représente la durée où le crédit de i augmente. La variation de crédit durant l'intervalle $\triangle t$ satisfait :

$$c_i(t) - c_i(s) = c_i(t) = \triangle t^+ idSl_i + \triangle t^- sdSl_i = \triangle t\ idSl_i - \triangle t^-(idSl_i - sdSl_i).$$

Ainsi, sur tout intervalle $\triangle t$ nous obtenons la relation,

$$\triangle t^- = \frac{\triangle t\ idSl_i - c_i(t)}{idSl_i - sdSl_i}. \tag{8.16}$$

Par la suite en considérant (8.16), il est possible d'exprimer la variation du flux de départ de la classe i durant l'intervalle $\triangle t$:

$$D_i(t) - D_i(s) = C\triangle t^- \geq C\frac{\triangle t\ idSl_i - c_i^{max}}{idSl_i - sdSl_i}.$$

Comme le flux de départ $D_i(t)$ est une fonction positive croissante et sachant que $D_i(s) = A_i(s)$, nous avons :

$$D_i(t) - A_i(s) \geq \left[\frac{C\ idSl_i}{idSl_i - sdSl_i}\left(t - \frac{c_i^{max}}{idSl_i}\right)\right]^+ = \beta_i^{\min}(t).$$

Par la suite,

$$D_i(t) \geq \beta_i^{\min}(\triangle t) + A_i(s) \geq \inf_{0\leq s\leq t}\left\{A_i(s) + \beta_i^{\min}(t-s)\right\} = (A_i * \beta_i^{\min})(t)$$

Ainsi, $\beta_i^{\min}(t)$ est une courbe de service simple de la classe i. □

8.3.2 Courbe de service strict

Théorème 22. *Dans les conditions de l'architecture de la Définition 26, si c_i^{max} est une borne supérieure du crédit de la classe i, et si c_i^{min} est une borne inférieure du crédit de la classe i alors une courbe de service strict de la classe CBS i ($i \in [1, N_{CBS}]$) est :*

$$\beta_i^{\min}(t) = \frac{C\ idSl_i}{idSl_i - sdSl_i}\left[t - \frac{c_i^{max} - c_i^{min}}{idSl_i}\right]^+. \tag{8.17}$$

Dans le cas où $sdSl_i = idSl_i - C$:

$$\beta_i^{\min}(t) = idSl_i\left[t - \frac{c_i^{max} - c_i^{min}}{idSl_i}\right]^+. \tag{8.18}$$

Démonstration. Notons $A_i(t)$ et $D_i(t)$ le flux d'arrivée et de départ de la classe CBS i. Soit t et s deux instants tels que durant l'intervalle $(s,t]$ la file i ne soit jamais vide, c'est-à-dire pour tout x dans $(s,t]$, $D_i(x) < A_i(x)$.

Notons $\triangle t$ l'intervalle $t - s$. Cet intervalle peut être décomposé en $\triangle t = \triangle t^+ + \triangle t^-$. $\triangle t^-$ représente la durée de transmission des messages de la classe CBS i. Durant $\triangle t^-$ le crédit de i diminue à la vitesse $sdSl_i$. Enfin $\triangle t^+$ représente la durée où le crédit de i augmente. La variation de crédit durant l'intervalle $\triangle t$ satisfait :

$$c_i(t) - c_i(s) = \triangle t^+ idSl_i + \triangle t^- sdSl_i = \triangle t\ idSl_i - \triangle t^-(idSl_i - sdSl_i).$$

Ainsi, sur tout intervalle $\triangle t$ nous obtenons la relation,

$$\triangle t^- = \frac{\triangle t\ idSl_i - c_i(t) + c_i(s)}{idSl_i - sdSl_i}. \tag{8.19}$$

Par la suite en considérant (8.19), il est possible d'exprimer la variation du flux de départ de la classe i durant l'intervalle $\triangle t$:

$$D_i(t) - D_i(s) = C\triangle t^- \geq C\frac{\triangle t\ idSl_i - c_i^{max} + c_i^{min}}{idSl_i - sdSl_i}.$$

Comme le flux de départ $D_i(t)$ est une fonction positive croissante nous avons :

$$D_i(t) - D_i(s) \geq \left[\frac{C\ idSl_i}{idSl_i - sdSl_i}\left(t - \frac{c_i^{max} - c_i^{min}}{idSl_i}\right)\right]^+ = \beta_i^{\min}(t-s).$$

Ainsi, $\beta_i^{\min}(t)$ est une courbe de service strict de i. □

8.3.3 Courbe de *shaping*

Théorème 23. *Dans les conditions de l'architecture de la Définition 26, si c_i^{max} est une borne supérieure du crédit de la classe i, et si c_i^{min} est une borne inférieure du crédit de la classe i alors une courbe de* shaping *de la classe CBS i ($i \in [1, N_{CBS}]$) est :*

$$\sigma_i(t) = \frac{C\ idSl_i}{idSl_i - sdSl_i}\left[t + \frac{c_i^{max} - c_i^{min}}{idSl_i}\right]^+. \tag{8.20}$$

Dans le cas où $sdSl_i = idSl_i - C$:

$$\sigma_i(t) = idSl_i\left[t + \frac{c_i^{max} - c_i^{min}}{idSl_i}\right]^+. \tag{8.21}$$

Démonstration. Notons $A_i(t)$ et $D_i(t)$ le flux d'arrivée et de départ de la classe CBS i. Soit t et $t + \triangle t$ deux instants quelconques. $\triangle t$ peut être décomposé en $\triangle t = \triangle t^+ + \triangle t^-$. $\triangle t^-$ représente la durée de transmission des messages de la classe CBS i. Durant $\triangle t^-$ le crédit de i diminue à la vitesse $sdSl_i$ et $\triangle t^+$ représente la durée où le crédit de i augmente. On alors :

$$\begin{aligned}
c_i^{\min} - c_i^{\max} &\leq c(t + \triangle t) - c(t) \\
&\leq \triangle t^+\ idSl_i + \triangle t^- sdSl_i \\
&\leq (\triangle t - \triangle t^-) idSl_i + \triangle t^-.sdSl_i \\
&\leq \triangle t\ idSl_i + \triangle t^-(sdSl_i - idSl_i) \\
&\leq \triangle t\ idSl_i - \triangle t^-(idSl_i - sdSl_i) \\
&\leq \triangle t\ idSl_i - (D_i(t + \triangle t) - D_i(t))\ \frac{idSl_i - sdSl_i}{C}
\end{aligned}$$

Comme le flux de départ $D_i(t)$ est une fonction positive croissante nous avons :

$$D_i(t + \triangle t) - D_i(t) \leq \frac{C\ idSl_i}{idSl_i - sdSl_i}\left[\triangle t + \frac{c_i^{max} - c_i^{min}}{idSl_i}\right]^+ = \sigma_i(\triangle t)$$

Par la suite, on en déduit que pour tous instants t et $t + \triangle t$:

$$D_i(t + \triangle t) \leq \sigma_i(\triangle t) + D_i(t)$$

Et donc que :

$$D_i(t + \triangle t) \leq \inf_{u,s \geq 0, u+s=t+\triangle t} \{\sigma_i(u) + D_i(s)\} = (D_i * \sigma_i)(t + \triangle t)$$

Ainsi, $\sigma_i(t)$ est une courbe de *shaping* de la classe i. □

8.3.4 Courbe de service maximal

La courbe de service maximal exprimée dans l'article [De Azua and Boyer, 2014] est fausse et ne peut donc pas être adaptée. En effet, au cours de la démonstration (en utilisant nos notations) il est démontré qu'à tout instant t il existe un instant $s \leq t$ tel que :

$$D(t) \leq A(s) + \frac{C\ idSl_i}{idSl_i - sdSl_i}\left((t - s) - \frac{c_i^{min}}{idSl_i}\right) \tag{8.22}$$

Les auteurs en déduisent par la suite :

$$D(t) \leq \inf_{0 \leq s \leq t} \left\{ A(s) + \frac{C\ idSl_i}{idSl_i - sdSl_i} \left((t-s) - \frac{c_i^{min}}{idSl_i} \right) \right\} \tag{8.23}$$

Or si le résultat est vrai pour un s particulier, cela ne prouve pas qu'il reste correct pour tout $s \leq t$.

Nous avons donc exprimé les différentes courbes de services offertes par un serveur TSN/CBS-BE aux flux CBS dans le cas où les portes sont toujours ouvertes.

8.4 Conclusion

Dans ce chapitre nous avons tout d'abord présenté l'architecture générale du réseau TSN. Puis nous avons défini une classe particulière de réseau TSN : TSN/CBS-BE. Ce type réseau est caractérisé par différents points :

— Il existe n files, désignées CBS qui utilisent l'algorithme *credit based shaper*.

— Le reste des files sont des files *best effort* (BE) qui n'ont pas d'algorithme de sélection de transmission et sont moins prioritaires que les files CBS.

— La *Gate Control List* est telle que les portes de toutes les files sont toujours ouvertes.

Ce type de réseau est une généralisation du réseau AVB puisqu'il ajoute la possibilité d'utiliser plus de deux files CBS.

Dans ce chapitre nous nous avons considéré qu'il n'y avait pas de flux TT. Le cas où des flux TT sont présents est étudié dans le Chapitre 9.

Pour établir les différentes courbes de service à l'aide du Calcul Réseau, il était nécessaire que le crédit de chaque classe soit borné inférieurement et supérieurement et que l'on soit capable de calculer ces bornes. Nous avons donc commencé par expliquer les règles de transmission des messages ainsi que celles encadrant la variation du crédit des classes CBS. Ensuite nous avons montré sous quelle condition les crédits étaient bornés. Dans un premier temps en nous plaçant dans le cadre de la norme sans flux TT. Puis dans un second temps, nous avons exprimé la condition de *non-overflow*, c'est-à-dire la condition qui permet de garantir que les crédits sont bornés, dans le cas général sans flux TT. Et enfin nous avons calculé ces bornes dans le contexte TSN/CBS-BE.

Cela fait il nous a été possible de construire, à l'aide du Calcul Réseau, un modèle afin de borner le temps de traversée des messages. Pour cela nous avons exprimé les différentes courbes de services offertes par un serveur TSN/CBS-BE aux flux CBS, nous avons ainsi exprimé la courbe de service simple, strict et la courbe de *shaping*. Nous avons de plus montré que la démonstration de la courbe de service maximal exprimée dans l'article [De Azua and Boyer, 2014] était fausse et ne pouvait donc pas être adaptée.

Chapitre 9

Modélisation en Calcul Réseau d'une architecture TSN mixant des trafic Time-Triggered, Credit Based Shaper and Best-Effort

9.1 TSN/TT-CBS-BE : l'impact du flux *Time-Triggered*

Dans le Chapitre 8 nous avons introduit un cas particulier de TSN que nous avons désigné par TSN/CBS-BE où :

— Il existe n files, désignées CBS qui utilisent l'algorithme *credit based shaper*.

— Le reste des files sont des files *best effort* (BE) qui n'ont pas d'algorithme de sélection de transmission et sont moins prioritaires que les files CBS.

— La *Gate Control List* est telle que les portes de toutes les files sont toujours ouvertes.

Nous allons maintenant étudier le cas où des flux *Time-Triggered* sont présents et où les portes sont parfois ouvertes et parfois fermées et l'impact que l'ajout de ces derniers a sur les flux CBS. En effet, dans ce chapitre nous allons considérer un nouveau cas particulier désigné par TSN/TT-CBS-BE où :

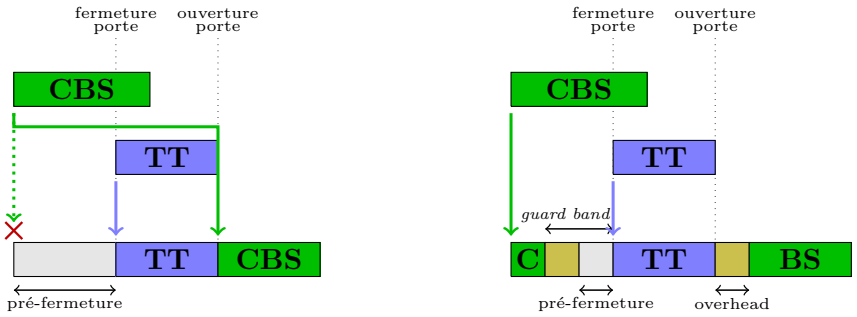
— La file d'attente la plus prioritaire est nommée *Time-Triggered* (TT) et ne possède pas d'algorithme de sélection de transmission.

— Il existe n files, désignées CBS qui utilisent l'algorithme *credit based shaper*.

— Le reste des files sont des files *best effort* (BE) qui n'ont pas d'algorithme de sélection de transmission et sont moins prioritaires que les files CBS et TT.

— La *Gate Control List* est telle que lorsque la porte de la file TT est ouverte les portes des autres files sont fermées. Et à l'inverse lorsque la porte de la file TT est fermée les portes des autres files sont ouvertes.

Dans TSN l'aspect *Time Triggered* est géré grâce à l'utilisation des portes. Ces portes sont définies de façon statique dans la *Gate Control List*. Ces dernières permettent de définir des fenêtre temporelles où seuls les flux TT peuvent être émis. Lorsque la porte d'une classe est fermée, il lui est impossible d'émettre un message. Ce concept de fenêtre s'il



(a) non-preemption.

(b) preemption with HOLD/RELEASE.

FIGURE 9.1 – Les politiques d'intégration CBS-TT dans TSN/TT-CBS-BE.

peut sembler similaire au flux TT de TTEthernet présenté dans le Chapitre 7 s'en éloigne par la nature de ce qu'il définit. Dans TTEthernet chaque flux TT, chaque message, avait sa propre date d'émission ; ici c'est l'ensemble du trafic TT qui se voit attribuer des fenêtres d'émission et non plus flux par flux. Cette différence a un impact majeur, en effet considérons deux messages *Time Triggered* A et B, ayant chacun une fenêtre réservée. Dans TTEthernet chacun a sa propre fenêtre d'émission (supposons que ce soit d'abord A puis B), l'ordre d'émission des deux message est donc connu *a priori* et ne dépend pas de l'ordre d'arrivée. Dans le cas de TSN à l'inverse les fenêtres TT sont partagées par l'ensemble des flux TT, les deux fenêtres ne sont pas affectées à un flux en particulier mais partagée par tous les flux TT avec un service FIFO. L'ordre d'émission dépendra donc de l'ordre d'arrivée des messages A et B.

9.1.1 Les différentes politiques d'intégration de TSN

Comme nous l'avons vu dans le Chapitre 8, TSN est en général un réseau non préemptif. Mais lorsqu'un message arrive dans une file asynchrone "juste avant" la fermeture des portes et pourrait être émis cela pose un problème. Le problème est le suivant : si la file n'est pas autorisée à émettre alors on perd de la bande passante, si la file est autorisée à émettre alors elle sera toujours en train d'émettre à la fermeture de sa porte.

L'addendum [802.1Qbu, 2016] traite ce problème, appelé politique d'intégration, en faisant référence à [802.3br, 2016] pour les détails sur la préemption. Deux modes d'intégration sont définis dans le contexte TSN/TT-CBS-BE, *non preemption* et *preemption with HOLD/RELEASE*. Elles sont illustrées dans la Figure 9.1.

Rappel sur la Préemption

La préemption définie dans [802.3br, 2016] Annexe R autorise à diviser une trame en plusieurs sous-trames, en ajoutant un *overhead*. Cependant, une sous-trame ne peut pas être inférieure à 64 octets. Ainsi, une trame de moins de 124 octets ne peut pas être divisée.

Les deux politiques d'intégration

Sachant qu'il n'est pas possible pour une classe d'émettre lorsque sa porte est fermée, la norme [802.1Qbv, 2015] prévoit qu'il n'est pas possible de commencer à émettre si l'on ne peut pas finir l'émission : "*a frame on a traffic class queue is not available for transmission [...] if there is an insufficient time available to transmit the entirety of the frame before the next gate-close event*". Mais "finir l'émission" n'a pas le même sens si la préemption présentée dans [802.1Qbu, 2016] est possible. Il y a donc deux solutions possibles dans TSN :

Non-preemption : ne pas commencer à émettre un message dont l'émission ne serait pas terminée à la prochaine fermeture des portes.

Preemption with HOLD/RELEASE : préempter tous les messages qui pourrait empiéter sinon sur la prochaine fenêtre TT. Mais la préemption "prend du temps", il faut anticiper la fermeture (la durée liée à la préemption est détaillée au paragraphe suivant). La solution de la norme est de définir une *guard band* dont le fonctionnement est le suivant : au début de la *guard band*, toutes les classes CBS et BE reçoivent une requête "M_P_HOLD.request(HOLD)". Dans la norme [802.3br, 2016] paragraphe 99.2, on nous explique que lorsque le paramètre "hold_req" reçoit la valeur "HOLD" cela cause la préemption dès que cela est possible et empêche l'émission de nouveaux messages. En choisissant correctement (ce point est détaillé au paragraphe suivant) la taille de la *guard band* il est possible de garantir que chaque message peut soit terminer d'être émis soit être préempté avant la date de fermeture et dans les deux cas ce message ne va pas empiéter sur la fenêtre TT. Le paramètre "hold_req" de chaque classe CBS reçoit la valeur "RELEASE" à la fin de la fenêtre TT.

Taille minimale de la *guard band* : Sachant qu'une sous trame a une taille minimum, il n'est pas toujours possible de préempter un trame à un instant donné. Soit parce que la taille totale de cette trame n'est pas suffisante, soit parce que la taille de la partie de la trame déjà émise n'a pas une taille suffisante pour devenir une sous trame, soit enfin parce que la taille de la partie de la trame qui n'a pas encore était émise n'a pas une taille suffisante pour devenir une sous trame. Ces différents phénomènes sont illustrés sur la Figure 9.2. Il faut donc que la *guard band* soit suffisamment grande pour que la préemption puisse toujours avoir lieu avant la fermeture des portes. Cette taille minimale est donnée dans [Thiele and Ernst, 2016] et est de 143 octets. De plus la préemption implique l'ajout d'un *overhead* qui permet de délimiter les sous-trames, l'*overhead* à considérer est de 8 octets.

Quelle que soit la politique choisie, celle ci peut entraîner l'apparition période où la porte d'une classe est ouverte mais où elle ne peut pas émettre. Nous avons décidé de désigner cette période la "pré-fermeture" : durant la "pré-fermeture" il n'est pas possible de commencer à émettre un nouveau paquet (car sinon ce dernier empiéterait sur la prochaine fenêtre TT). La "pré-fermeture" a été représentée en gris sur le Figure 9.1. Contrairement à la *guard band* qui a une taille fixe, la taille de cette "pré-fermeture" est variable et inconnue *a priori*. Il est cependant possible de la borner supérieurement. En cas de "Non-preemption" la longueur maximale de la "pré-fermeture" d'une classe CBS est de la taille maximale d'une trame qui peut interférer avec la fenêtre TT, c'est-à-dire la longueur

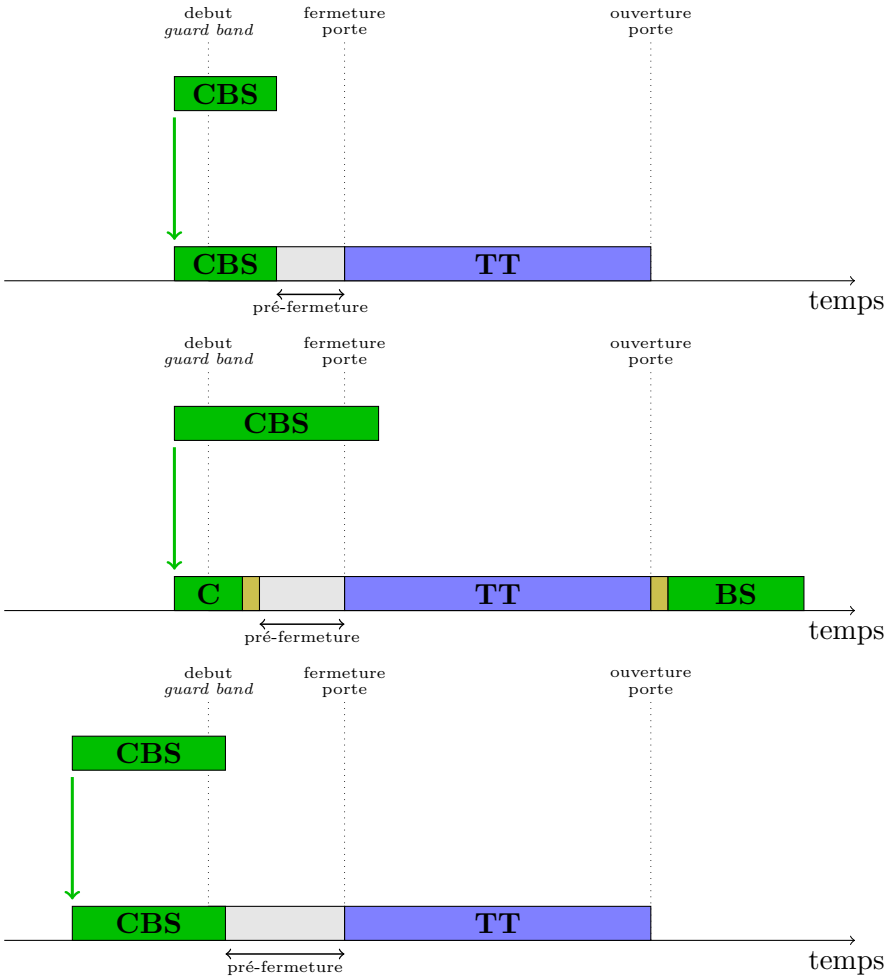


FIGURE 9.2 – Fonctionnement de la *guard band* dans TSN/TT-CBS-BE.

maximale d'une trame de cette classe. En cas de "Preemption with HOLD/RELEASE" la longueur de la "pré-fermeture" est elle aussi variable comme illustré sur la Figure 9.2, sa longueur maximale est égale à la longueur de la *guard band*.

9.1.2 L'impact des portes sur le *credit based shaper*

Comme nous l'avons vu précédemment, dans TSN/TT-CBS-BE un certains nombre de files sont CBS. Ces files utilisent l'algorithme *credit based shaper*.

Cependant l'ajout des portes a nécessité une modification de cet algorithme et de nouvelles règles ont dû être définies. Ainsi les règles ne sont plus celles définies au Chapitre 8. Les nouvelles règles définies dans la norme sont les suivantes (les modifications sont en italique) :

R1 Lorsque le serveur est en attente, il sélectionne pour l'émission le message en tête de la file la plus prioritaire parmi les files *ready*. Une classe est *ready* lorsque :

— sa file est non vide,

— son crédit est positif ou nul dans le cas particulier des files CBS,

— *sa porte est ouverte*,

— *il y a un temps suffisant avant la prochaine fermeture de sa porte pour transmettre le message en tête de sa file (ce temps dépend de la politique d'intégration utilisée).*

R2 Lorsqu'une classe CBS émet un message alors son crédit diminue à la vitesse $sdSl_i$. *Cela ne concerne pas l'*overhead *dû à la préemption.*

R3 Lorsque la file d'une classe CBS est non vide mais n'est pas en train d'émettre *(ou émet un* overhead*)* et si sa porte est ouverte alors son crédit augmente à la vitesse $idSl_i$.

R4 Lorsque la file d'une classe CBS est vide et si son crédit est positif alors il est remis à 0.

R5 Lorsque la file d'une classe CBS est vide, *si sa porte est ouverte* et si son crédit est négatif alors son crédit augmente à la vitesse $idSl_i$.

R6 *Lorsque la porte d'une classe CBS est fermée alors elle ne peut pas émettre de message et son crédit reste constant.*

Ces nouvelles règles définies dans la norme permettent de garantir le respect des portes : ainsi une classe CBS ne pourra pas émettre lorsque sa porte est fermée. Cependant en suivant ces règles, définies dans la norme, il est possible d'avoir un "fonctionnement inattendu" lorsque l'on utilise l'algorithme *credit based shaper*, l'utilisation de ces règles peut conduire à un fonctionnement qui n'est pas "équitable".

En effet avec les règles décrites précédemment, durant les "pré-fermetures" et les émissions d'*overhead* toutes les classes CBS non vides gagnent du crédit puisque à ces instants leurs portes sont ouvertes donc d'après la règle **R3** leurs crédits augmentent. Or ce fonctionnement a tendance à avantager les classes les plus prioritaires. En effet lorsqu'il sera de nouveau possible d'émettre si plusieurs classes ont un crédit positif ce sera la classe la plus prioritaire qui pourra émettre.

Sur la Figure 9.3 un exemple mettant en évidence ce phénomène a été représenté. Dans cet exemple nous sommes partis d'une situation de départ avec deux classes CBS (la classe A et la classe B) et une classe *best effort* (Figure 9.3a). Les *idle slope* et les *send slope* ont été choisis tels que la classe A se voit réserver au plus 25% de la bande passante, la classe B au plus 50% et la classe *best effort* aura donc au moins 25% de la bande passante.

Dans les situations suivantes la classe la plus prioritaire (la classe A) devient une classe *Time-Triggered*. Les portes des deux autres classes sont donc fermées 25% du temps afin de permettre à la classe A d'émettre, les 75% restant sont donc uniquement partagés entre la classe B et la classe BE. La classe B se voit donc réserver au plus 2/3 de la bande passante lorsque sa porte est ouverte et le flux BE aura donc en théorie au moins 1/3 de la bande passante hors fenêtre TT.

Sur la Figure 9.3b est représenté le résultat que l'on obtient en respectant les règles de la norme. Nous pouvons voir que le flux B monopolise le lien conduisant à une situation de famine pour la classe BE, alors que l'on s'attendait à ce qu'elle reçoive 1/3 de la bande passante laissée par TT. En effet la présence des "pré-fermetures" empêche d'émettre plus de messages. Cependant avec les règles de la norme (**R3**) le flux B regagne du crédit durant ces dernières et peut donc émettre en permanence.

Ainsi afin de conserver un fonctionnement plus proche de celui désiré lorsque l'on utilise l'algorithme *credit based shaper* nous avons proposé un nouvel ensemble de règles que l'on désignera par la suite par "règles équitables". Une façon d'interpréter ces nouvelles règles est de considérer que le crédit ne doit évoluer que lorsque les classes CBS ont réellement accès au medium. Cela signifie qu'il doit non seulement être gelé lorsque les portes sont fermées mais aussi durant les "pré-fermetures" et lors de l'émission d'un *overhead*. Cela nécessite de modifier les règles **R3**, **R5** et **R6** :

R'3 Lorsque la file d'une classe CBS est non vide mais n'est pas en train d'émettre et si sa porte est ouverte alors son crédit augmente à la vitesse $idSl_i$. *Cela ne concerne pas les "pré-fermetures" ni l'émission d'un* overhead *dû à la préemption.*

R'5 Lorsque la file d'une classe CBS est vide, si sa porte est ouverte et si son crédit est négatif alors son crédit augmente à la vitesse $idSl_i$. *Cela ne concerne pas les "pré-fermetures" ni l'émission d'un* overhead *dû à la préemption.*

R'6 Lorsque la porte d'une classe CBS est fermée alors elle ne peut pas émettre de message et son crédit reste constant. *De plus durant les "pré-fermetures" et lors de l'émission d'un* overhead *le crédit reste constant.*

Sur la Figure 9.3c est représenté le résultat que l'on obtient en respectant les "règles équitables". Nous pouvons voir que le flux B ne monopolise plus le lien. Les 2/3 tiers des messages émis sont bien des flux B et les flux BE représentent un message sur trois. Le crédit permet de s'assurer que le flux le plus prioritaire (ici B) ne s'accapare pas toute la bande passante.

Nous avons détaillé le fonctionnement du crédit dans TSN/TT-CBS-BE en suivant les règles énoncées dans la norme. Nous avons montré en quoi ces règles pouvaient conduire à un fonctionnement non désiré. Nous avons par la suite proposé un ensemble de règles alternatives qui permettent d'obtenir un résultat plus satisfaisant.

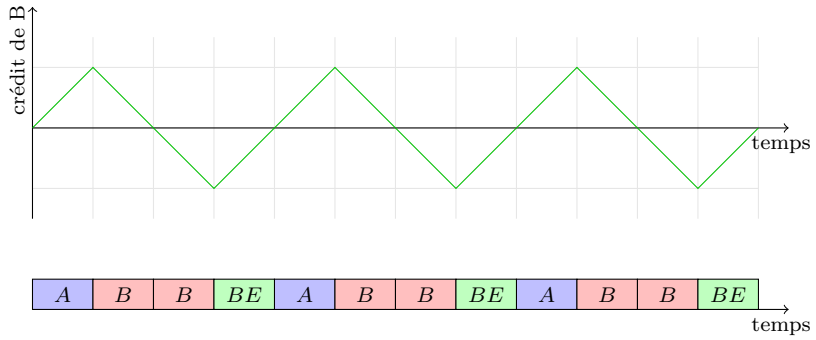
Dans la suite nous allons montrer comment il est possible à partir de ces règles de borner le crédit dans TSN/TT-CBS-BE.

9.2 Borner le crédit dans TSN/TT-CBS-BE

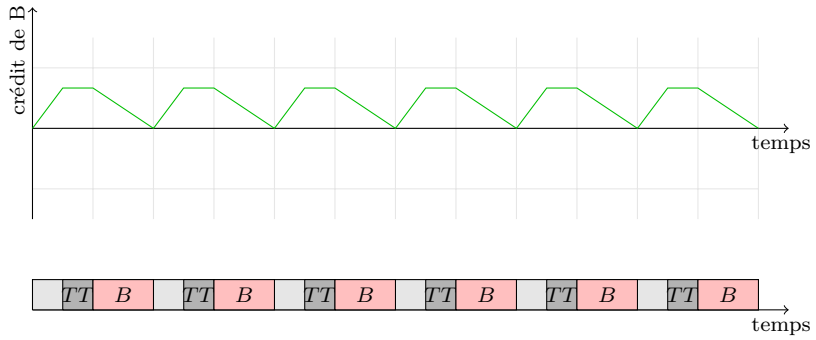
Dans la section précédente nous avons présenté les hypothèses ainsi que le fonctionnement du réseau TSN/TT-CBS-BE lorsque un flux TT est présent. Nous avons en particulier détaillé le fonctionnement du crédit des classes CBS. Dans cette section nous donnerons les conditions sur l'*idle slope* et le *send slope* qui permettent d'assurer que le crédit est borné ainsi que le calcul de ces bornes lorsqu'elles existent.

Nous allons tout d'abord définir l'architecture de référence que nous considérerons par la suite :

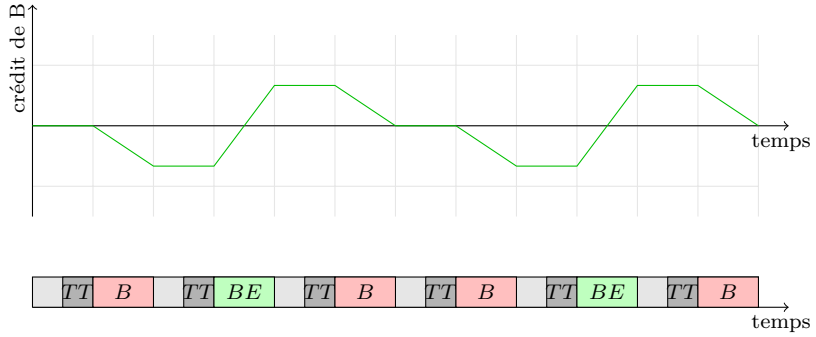
Définition 27 (Architecture TSN/TT-CBS-BE)**.** *Notre architecture TSN/TT-CBS-BE de référence possède une file TT,* N_{CBS} *files CBS, le reste des files sont BE. La vitesse de transmission du lien est noté* C*. Le système possède une hyper-période* P*, il y a* N_{TT} *fenêtres TT par hyper-période et la durée cumulée de ces fenêtres par hyper-période vaut* $\triangle_{TT}$*.*



(a) Situation sans flux TT.



(b) Situation avec flux TT en suivant les règles de la norme.



(c) Situation avec flux TT en suivant les “règles équitables”.

FIGURE 9.3 – Fonctionnement du crédit en fonction des règles utilisées.

Lorsque les portes de la file TT sont ouvertes, toutes les portes des autres files sont fermées et inversement.

Pour tout $1 \leq i \leq N_{CBS}$*, on note* $c_i(t)$ *le crédit de la classe* i *à l'instant* t*.*

De plus, on note L_i^{max} *la taille maximale de n'importe quel message de la classe* i *et* L_{BE}^{max} *la taille maximale d'un message BE. Et l'on note* $L_{>i}^{max} = \max\left(\max_{j\in[i+1,N_{CBS}]} L_j^{max}, L_{BE}^{max}\right)$*, la taille maximale d'un message appartenant à une classe moins prioritaire que la classe* i*.*

On note $idSl_i$ *et* $sdSl_i$ *l'*idle slope *et le* send slope *de la classe* i*. La norme [802.1Qbv, 2015] donne la contrainte* $sdSl_i = idSl_i - C$ *mais cette contrainte est inutile pour certaines de nos preuves il sera donc précisé lorsque nous la considérons vraie.*

Pour finir notons $Q_{CBS}^{\leq i}$ *l'ensemble des files CBS avec une priorité supérieure ou égale à la classe* i *et* $Q_{CBS}^{>i}$ *l'ensemble des files avec une priorité inférieure à la classe* i*.*

9.2.1 Condition de *non-overflow* dans le contexte TSN/TT-CBS-BE suivant les règles définies dans la norme

Dans le Chapitre 8 nous avons exprimé la condition de *non-overflow* sans flux TT. Nous allons maintenant nous placer dans le contexte TSN/TT-CBS-BE en supposant donc non seulement que $sdSl_i = idSl_i - C$ mais aussi qu'il y a maintenant un flux TT. De plus nous considérerons tout d'abord que nous appliquons les règles définies dans la norme.

Le crédit étant "gelé" lorsque les portes sont fermées, il peut sembler dans un premier temps qu'il n'y a pas de différence avec la situation décrite dans le Chapitre 8. Mais ce serait oublier l'existence des "pré-fermetures" et des *overhead*. En effet durant ces instants le crédit de toutes les classes peut potentiellement augmenter (si elles sont non vides) or cela peut entraîner un phénomène d'*overflow*. Par exemple, dans un cas très simple avec uniquement deux classes M_1 et M_2 telles que $idSl_1 = idSl_2 = 0,5C$ et $sdSl_1 = sdSl_2 = -0,5C$, on a bien les propriétés $sdSl_i = idSl_i - C$. Dans cet exemple tous les paquets font la même taille, nous faisons de plus l'hypothèse que les deux files ne sont jamais vides. Sur Figure 9.4 nous avons représenté les paquets émis en respectant les règles énoncées dans la Section 9.1 et l'évolution du crédit qui en résulte pour chacune des deux classes. Or nous observons rapidement que dans cette situation le crédit de la classe M_2 diverge et ne peut donc pas être borné et alors que pourtant $idSl_1 + idSl_2 \leq C$. La condition de *non-overflow* du Chapitre 8 n'est donc plus correcte.

Nous allons donc exprimer la condition de *non-overflow* et calculer la borne sur le crédit lorsque cette condition est vérifiée.

Pour la suite il nous faut introduire deux notions la *phase de pseudo-fermeture* et la *durée de pseudo-fermeture.*

Définition 28 (Phase de pseudo-fermeture)**.** *Nous désignons par* phase de pseudo-fermeture *de* $Q_{CBS}^{\leq i}$*, l'ensemble instant du temps où les portes CBS sont ouvertes mais où les interaction avec une fenêtre TT peut empêcher l'émission d'un message de* $Q_{CBS}^{\leq i}$*.*

La phase de pseudo-fermeture *de* $Q_{CBS}^{\leq i}$ *est donc en cas de :*

Non-Preemtion *l'ensemble des instants où au moins une classe de* $Q_{CBS}^{\leq i}$ *est non vide et où toutes les classes non vides de* $Q_{CBS}^{\leq i}$ *sont en situation de "pré-fermeture".*

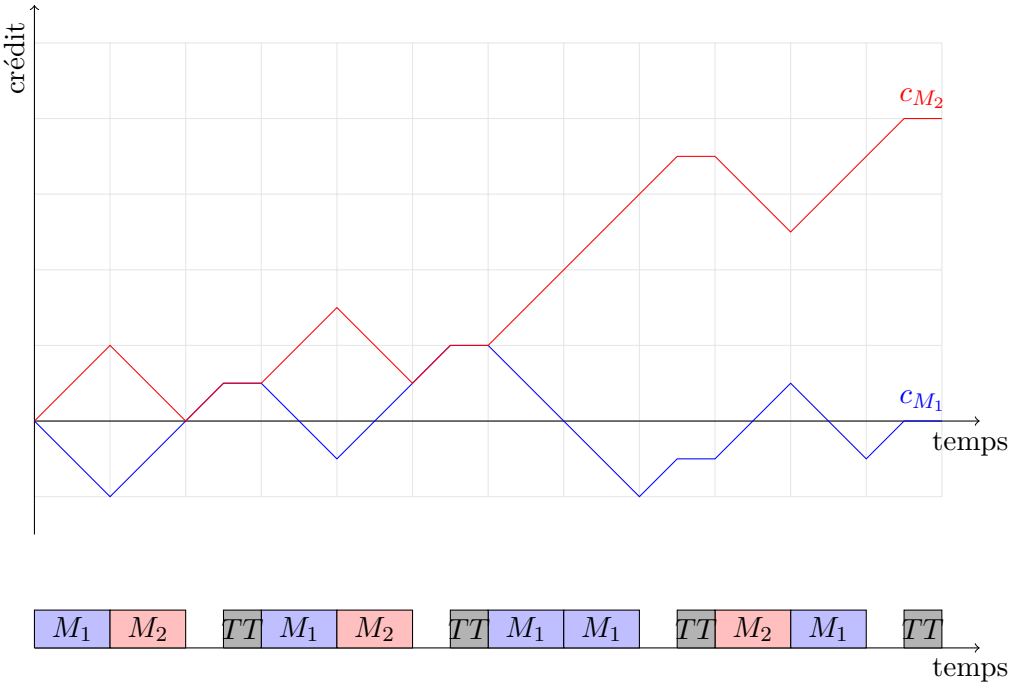


FIGURE 9.4 – Exemple de situation où un phénomène d'*overflow* se produit à cause de la fermeture des portes.

Preemption with HOLD/RELEASE *l'ensemble des instants où au moins une classe de* $Q_{CBS}^{\leq i}$ *est non vide et où toutes les classes non vides de* $Q_{CBS}^{\leq i}$ *sont en situation de "pré-fermeture" ainsi que l'ensemble des instant où un* overhead *est émis.*

Définition 29 (Durée maximale de pseudo-fermeture)**.** *Nous désignons par* durée maximale de pseudo-fermeture *de* $Q_{CBS}^{\leq i}$*, et notons* $T_{i,max}^{pf}$*, une borne supérieur de la durée de l'ensemble des* phases de pseudo-fermeture *de* $Q_{CBS}^{\leq i}$ *durant une hyper-période.*

Remarque : Nous avons déjà détaillé dans la Section 9.1.1 une façon de borner la durée de "pré-fermeture' pour chaque classe en fonction de la politique d'intégration, en prenant le maximum de ces durées on obtient bien une *durée maximale de pseudo-fermeture.*

Théorème 24. *Dans les conditions de l'architecture de la Définition 27.*

Soit $i \leq N_{CBS}$*, si* $\sum_{k=1}^{i} idSl_k \leq C\left(1 - \frac{T_{i,max}^{pf}}{P - \triangle_{TT}}\right)$ *où* $T_{i,max}^{pf}$ *désigne durée maximale de pseudo-fermeture de* $Q_{CBS}^{\leq i}$ *alors le crédit de la classe* i *est borné.*

De plus pour tout $t \geq 0$*, une borne supérieure du crédit de la classe* i *est :*

$$c_i(t) \leq \left(\frac{L_{>i}^{max}}{C} + T_{i,max}^{pf}\right) \sum_{j=1}^{i} idSl_j - \sum_{j=1}^{i-1} c_j^{min} \triangleq c_i^{max} \tag{9.1}$$

avec c_j^{min} *une borne inférieure du crédit de la classe* j *(par exemple celle du Théorème 18).*

Démonstration. Nous allons nous intéresser à la classe i et définir $c^H(t) = \sum_{j=1}^{i} c_j(t)$ la somme des crédits des classes de priorité plus haute ou égale à i.

Maintenant définissons $idSl_{pf,i} = C - \sum_{k=1}^{i} idSl_k$ et associons à cet *idle slope* un crédit (virtuel) $c_{pf,i}$. Les règles de variation de ce crédit sont les suivantes :

R0-PF : $c_{pf,i}(0) = 0$.

R1-PF : Si les portes CBS sont fermées $c_{pf,i}$ reste constant.

R2-PF : Si les portes CBS sont ouvertes et que l'on est dans une *phase de pseudo-fermeture* de $Q_{CBS}^{\leq i}$ alors $c_{pf,i}$ diminue à la vitesse $idSl_{GB} - C$.

R3-PF : Si les portes CBS sont ouvertes et que l'on n'est pas dans une *phase de pseudo-fermeture* de $Q_{CBS}^{\leq i}$ alors $c_{pf,i}$ augmente à la vitesse $idSl_{GB}$.

Prouvons tout d'abord que $c_{pf,i}(t+P) \geq c_{pf,i}(t)$ et que sur une durée $\triangle t \leq P$ la variation de crédit est bornée inférieurement par $(idSl_{GB} - C)T_{i,max}^{pf}$.

Le crédit de $c_{pf,i}$ diminue durant la *phase de pseudo-fermeture* de $Q_{CBS}^{\leq i}$, or la durée de pseudo-fermeture durant P est au minimum égale à $T_{i,max}^{pf}$. Le crédit de $c_{pf,i}$ diminue donc d'au plus $(idSl_{GB} - C)T_{i,max}^{pf}$ donc :

$$c_{pf,i}(t + \triangle t) - c_{pf,i}(t) \geq (idSl_{GB} - C)T_{i,max}^{pf} = -\sum_{k=1}^{i} idSl_k T_{i,max}^{pf}$$

De plus durant la durée P le crédit de $c_{pf,i}$ augmente au minimum de $idSl_{GB}(P - T - T_{i,max}^{pf})$ donc :

$$\begin{aligned} c_{pf,i}(t+P) - c_{pf,i}(P) &\geq idSl_{GB}(P - T - T_{i,max}^{pf}) + (idSl_{GB} - C)T_{i,max}^{pf} \\ &\geq idSl_{GB}(P - T) - CT_{i,max}^{pf} \\ &\geq (C - \sum_{k=1}^{i} idSl_k)(P - T) - CT_{i,max}^{pf} \end{aligned}$$

Or $\sum_{k=1}^{i} idSl_k \leq C\left(1 - \frac{T_{i,max}^{pf}}{P - \triangle_{TT}}\right)$ donc :

$$c_{pf,i}(t+P) \geq c_{pf,i}(P)$$

Considérons un instant quelconque t. Nous allons maintenant montrer que $c_{pf,i} + c^H$ est une fonction décroissante en t ou que $c^H(t) \leq \frac{L_{>i}^{max}}{C} \sum_{j=1}^{i} idSl_j$.

Si à l'instant t les portes sont fermées alors $c_{pf,i} + c$ est constant.

Si à l'instant t les portes sont ouvertes et si t est dans une *phase de pseudo-fermeture*. Alors $c_{pf,i}(t)$ diminue à la vitesse $idSl_{GB} - C = -\sum_{k=1}^{i} idSl_k$ et $c^H(t)$ augmente au plus à la vitesse $\sum_{k=1}^{i} idSl_k$ donc $c_{pf,i} + c$ est une fonction décroissante (potentiellement au sens large).

Si à l'instant t les portes sont ouvertes et si t n'est pas dans une *phase de pseudo-fermeture*. Alors $c_{pf,i}(t)$ augmente à la vitesse $idSl_{GB} = C - \sum_{k=1}^{i} idSl_k$.

— Si un message de $Q_{CBS}^{\leq i}$ est en cours d'émission alors le crédit d'au moins une classe de $Q_{CBS}^{\leq i}$ (la classe j) diminue à la vitesse $idSl_j - C$. Le crédit des autres classes augmente d'au plus $idSl_k$ Soit une variation de $c^H(t)$ d'au plus $\sum_{k=1}^{i} idSl_k - C$. Donc $c_{pf,i} + c$ est une fonction décroissante (potentiellement au sens large).

— Si un message de $Q_{CBS}^{\leq i}$ n'est en cours d'émission à l'instant t, alors soit aucun message n'est en cours d'émission à l'instant t ce qui implique que $c^H(t) \leq 0$. Soit un message de $Q_{CBS}^{>i}$ est en cours d'émission à l'instant t, alors notons $s < t$ la date de début d'émission de ce message. On sait alors que $c^H(s) \leq 0$ sinon un message de $Q_{CBS}^{\leq i}$ aurait été émis à l'instant s. De plus par construction on sait qu'au plus un message de $Q_{CBS}^{>i}$ est émis entre s et t on obtient donc $t - s \leq \frac{L_{>i}^{max}}{C}$. Durant l'émission de ce message le crédit c augmente au plus à la vitesse $\sum_{k=1}^{i} idSl_k$ donc il est possible d'en déduire que $c^H(t) \leq \frac{L_{>i}^{max}}{C} \sum_{k=1}^{i} idSl_k$.

Pour résumer, nous avons montré qu'à un instant quelconque t soit la fonction $c_{pf,i}+c^H$ est une fonction décroissante soit $c^H(t) \leq \frac{L_{>i}^{max}}{C} \sum_{j=1}^{i} idSl_j$.

Nous avons aussi montré que durant un intervalle de longueur P, la fonction $c_{pf,i}$ décroît d'au plus $-T_{i,max}^{pf} \sum_{j=1}^{i} idSl_j$ et que sur l'ensemble de l'intervalle elle aura augmenté. Nous pouvons donc en déduire que durant un intervalle de longueur P la fonction c^H augmente d'au maximum $T_{i,max}^{pf} \sum_{j=1}^{i} idSl_j$.

Il est donc possible d'en déduire que :

$$c^H(t) \leq \left(\frac{L_{>i}^{max}}{C} + T_{i,max}^{pf} \right) \sum_{j=1}^{i} idSl_j$$

En supposant que l'on connaît une borne inférieure, c'est-à-dire, $c_j(t) \geq c_j^{min}$, on obtient :

$$c_i(t) \leq \left(\frac{L_{>i}^{max}}{C} + T_{i,max}^{pf} \right) \sum_{j=1}^{i} idSl_j - \sum_{j=1}^{i-1} c_j^{min}$$

□

9.2.2 Condition de *non-overflow* dans le contexte TSN/TT-CBS-BE en suivant les "règles équitables"

Dans la section précédente nous avons exprimé une condition de *non-overflow*, ainsi qu'une borne supérieure sur le crédit de chaque flux, avec l'hypothèse $sdSl_i = idSl_i - C$, avec un flux TT et en appliquant les règles définies dans la norme.

Nous allons maintenant nous intéresser au cas avec l'hypothèse $sdSl_i = idSl_i - C$, avec un flux TT mais en appliquant les "règles équitables".

Du point de vue du crédit la différence est double. En effet avec ces règles le crédit d'une classe n'évolue pas lorsque :

— les données émises correspondent à un *overhead*,

— la classe en question est dans une "pré-fermeture".

Nous allons maintenant donner la condition de *non-overflow* ainsi qu'une borne supérieure du crédit pour toute classe CBS.

Théorème 25. *Dans les conditions de l'architecture de la Définition 27. Si* $\forall i$, $sdSl_i = idSl_i - C$ *et si* $\sum_{i=1}^{N_{CBS}} idSl_i \leq C$, *alors le crédit de toute classe* i *est borné. De plus pour tout* $t \geq 0$, *une borne supérieure du crédit de la classe* i *est :*

$$c_i(t) \leq \frac{L_{>i}^{max}}{C} \sum_{j=1}^{i} idSl_j - \sum_{j=1}^{i-1} c_j^{min} \triangleq c_i^{max} \tag{9.2}$$

avec c_j^{min} une borne inférieure du crédit de la classe j (par exemple celle du Théorème 18).

Démonstration. Nous allons nous intéresser à la classe i et définir $c^H(t) = \sum_{j=1}^{i} c_j(t)$ la somme des crédits des classes de priorité plus haute ou égale à i. Nous allons montrer dans un premier temps que cette somme, sous certaines conditions, est bornée. Pour cela considérons un instant quelconque t.

Si les portes CBS sont fermées à la date t alors le crédit est gelé donc $c^H(t)$ n'a pas évolué depuis la dernière date de fermeture. Il existe donc une date t_o où les portes CBS sont ouvertes qui vérifie $c^H(t_o) = c^H(t)$ Considérons pour la suite que les portes CBS sont ouvertes à la date t.

Si la date t appartient à une "pré-fermeture" ou à un *overhead* notons $s < t$ la date de début de cette *guard band* ou de cet *overhead*. Le crédit c^H est donc gelé entre s et t donc $c^H(t) = c^H(s)$.

Considérons maintenant que t n'appartient pas à une "pré-fermeture" ou à un *overhead*.

Si toute les files de $Q_{CBS}^{\leq i}$ sont vides, on a alors $c^H(t) \leq 0$.

Si aucun message n'est en cours d'émission à l'instant t, cela signifie que le crédit de chaque file CBS non vide est strictement négatif, ce qui implique entre autre $c^H(t) < 0$.

Si un message est en cours d'émission à l'instant t, alors notons $s < t$ la date de début d'émission de ce message.

Il y a deux possibilités, soit ce message appartient à $Q_{CBS}^{\leq i}$ soit il appartient à $Q_{CBS}^{>i}$.

Si le message appartient à $Q_{CBS}^{>i}$, alors soit $c^H(s) = 0$ et toutes les files de $Q_{CBS}^{\leq i}$ étaient vides à l'instant s soit $c^H(s) < 0$ (autrement, à l'instant s, c'est un message de $Q_{CBS}^{\leq i}$ qui aurait été choisi pour l'émission) donc dans toutes ces situations $c^H(s) \leq 0$.

De plus durant l'émission de ce message le crédit $c^H(t)$ a augmenté d'au plus :

$$c^H(t) - c^H(s) \leq \sum_{j=1}^{i} idSl_j(t-s).$$

Or par construction on sait qu'au plus un message de $Q_{CBS}^{>i}$ est émis entre s et t, si l'on note $L_{>i}^{max}$ la longueur maximale de ce message on obtient donc :

$$t - s \leq \frac{L_{>i}^{max}}{C}.$$

Puisque l'on a déjà prouvé que $c^H(s) \leq 0$ il est possible de déduire :

$$c^H(t) \leq \frac{L_{>i}^{max}}{C} \sum_{j=1}^{i} idSl_j.$$

Enfin si le message émis à l'instant t appartient à la classe $k \in Q_{CBS}^{\leq i}$, alors entre l'instant s et l'instant t le crédit de la classe k a diminué à la vitesse $sdSl_k$, alors que le

crédit des autres classes $j \in Q_{CBS}^{\leq i}$ a lui augmenté d'au plus $idSl_j$. On en déduit donc pour le crédit c^H :

$$\begin{aligned} c^H(t) - c^H(s) &\leq sdSl_k(t-s) + \sum_{j=1, j\neq k}^{i} idSl_j(t-s) \\ &\leq \left(sdSl_k - idSl_k + \sum_{j=1}^{N_{CBS}} idSl_j - \sum_{j=i+1}^{N_{CBS}} idSl_j \right)(t-s). \end{aligned}$$

Puisque l'on sait que $sdSl_i = idSl_i - C$ alors si $\sum_{i=1}^{N_{CBS}} idSl_i \leq C$, on a :

$$c^H(t) - c^H(s) \leq - \sum_{j=i+1}^{N_{CBS}} idSl_j(t-s) \leq 0.$$

Pour résumer, à un instant t quelconque on a soit $c^H(t) \leq 0$ soit $c^H(t) \leq \frac{L_{>i}^{max}}{C} \sum_{j=1}^{i} idSl_j$ soit il existe un instant $s < t$ tel que $c^H(t) \leq c^H(s)$. Or le crédit ne pouvant augmenter que de façon continue et étant nul à $t = 0$ on peut en déduire :

$$c^H(t) \leq \frac{L_{>i}^{max}}{C} \sum_{j=1}^{i} idSl_j$$

En supposant que l'on connaît une borne inférieure, c'est-à-dire, $c_j(t) \geq c_j^{min}$, on obtient :

$$c_i(t) \leq \frac{L_{>i}^{max}}{C} \sum_{j=1}^{i} idSl_j - \sum_{j=1}^{i-1} c_j^{min}.$$

□

Remarque : La condition de *non-overflow* du Théorème 25 est la même que celle du Théorème 19. En effet en suivant les "règles équitables", le crédit des classes CBS est gelé durant les fenêtres TT ainsi que durant les "pré-fermetures" et les émissions d'*overhead*.

9.3 Modélisation de TSN/TT-CBS-BE en Calcul Réseau

Dans cette section nous allons nous concentrer sur les courbes de service résiduel des classes CBS. Nous ne détaillerons pas les courbes d'arrivée qui sont les mêmes que celles présentes dans AVB et ont déjà été présentées dans [De Azua and Boyer, 2014].

Les courbes de service qui seront présentées ici prennent en compte les deux politiques d'intégration disponibles dans TSN et peuvent être utilisées avec les règles de la norme ainsi que les "règles équitables".

9.3.1 Temps de de gel de crédit pour les classes CBS

Commençons par nous intéresser au temps de gel du crédit sur un intervalle de temps puisque ce temps de fermeture aura un impact direct sur le service offert au flux CBS par le serveur.

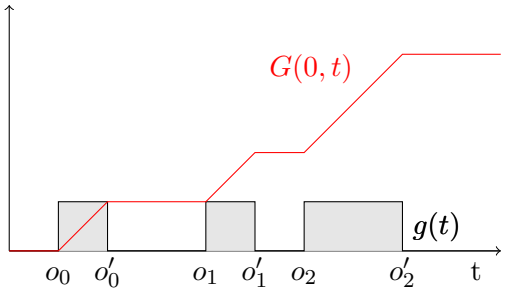


FIGURE 9.5 – Fonctions de temps de fermeture

Pour modéliser ce temps de fermeture nous introduisons deux fonctions g et G :

$$g(t) = \begin{cases} 0 & \text{si le crédit n'est pas gelé à l'instant t} \\ 1 & \text{si le crédit est gelé à l'instant t} \end{cases} \tag{9.3}$$

$$G(t, t+\triangle t) = \int_t^{t+\triangle t} g(x)\,\mathrm{d}x \tag{9.4}$$

Dans le cas des règles définies par la norme, $g(t)$ représente l'état de la porte à l'instant t, et $G(t, t+\triangle t)$ le temps cumulé où la porte a été fermée entre t et $t+\triangle t$. Ces deux fonctions ont été représentées sur la Figure 9.5.

Pour la suite nous ferons l'hypothèse que le système possède une hyper-période P et qu'il y a N fenêtres TT par hyper-période, la i-ième fenêtre commençant à l'instant o_i et termine à l'instant o'_i (Figure 9.5). Nous supposons de plus que $i \leq j \Rightarrow o_i \leq o'_i \leq o_j \leq o'_j$.

Théorème 26. *Soit $t, d \in \mathbb{R}^+$, alors :*

$$G(t, t+d) \leq \delta_{TT,u}(d) \stackrel{\triangle}{=} \max_{0 \leq i \leq N-1} \{G(o_i, o_i + d)\}\,. \tag{9.5}$$

De plus : $\forall d, \exists t$ tel que : $G(t, t+d) = \delta_{TT,u}(d)$.

Ce théorème borne supérieurement le temps de fermeture des portes. Cette borne est la meilleure possible puisqu'elle peut être atteinte.

Remarque Sur un intervalle de temps $[s, t]$, il existe $d_f = G(s, t)$, la durée de fermeture des portes sur cet intervalle (bien sûr, $d_f \leq t - s$). La durée maximale du temps de fermeture des portes $\delta_{TT,u}$ vérifie : $d_f \leq \delta^h_{TT,u}(t-s)$.

Démonstration. Pour tout $t \geq 0$, $g(t) = 0$ ou $g(t) = 1$

— Cas $g(t) = 0$:
Alors il existe $i \in [0, N-1]$ tel que la prochaine fenêtre TT après t est la i-ième

fenêtre TT : $o'_{i-1} \leq t \leq o_i$

$$\begin{aligned} G(t, t+d) &= \int_t^{t+d} g(x)\,\mathrm{d}x \\ &= \int_t^{o_i} g(x)\,\mathrm{d}x + \int_{o_i}^{t+d} g(x)\,\mathrm{d}x \\ &= \int_t^{o_i} 0\,\mathrm{d}x + \int_{o_i}^{t+d} g(x)\,\mathrm{d}x \\ &\leq \int_{o_i}^{o_i+d} g(x)\,\mathrm{d}x = G(o_i, o_i+d) \end{aligned}$$

— Cas $g(t) = 1$:
Alors il existe $i \in [0, N-1]$ tel que t soit dans la i-ième fenêtre TT : $o_i \leq t \leq o'_i$

$$\begin{aligned} G(t, t+d) &= \int_t^{t+d} g(x)\,\mathrm{d}x \\ &= \int_t^{o_i+d} g(x)\,\mathrm{d}x + \int_{o_i+d}^{t+d} g(x)\,\mathrm{d}x \\ &\leq \int_t^{o_i+d} g(x)\,\mathrm{d}x + \int_{o_i+d}^{t+d} 1\,\mathrm{d}x \\ &\leq \int_t^{o_i+d} g(x)\,\mathrm{d}x + \int_{o_i}^{t} 1\,\mathrm{d}x \\ &\leq \int_t^{o_i+d} g(x)\,\mathrm{d}x + \int_{o_i}^{t} g(x)\,\mathrm{d}x \\ &\leq G(o_i, o_i+d) \end{aligned}$$

Donc pour tout $t \geq 0$, il existe $i \in [0, N-1]$ tel que :

$$G(t, t+d) \leq G(o_i, o_i+d) \leq \max_{0 \leq i \leq N-1} \{G(o_i, o_i+d)\}. \tag{9.6}$$

□

Théorème 27. *Soit $t, d \in \mathbb{R}^+$, alors :*

$$G(t, t+d) \geq \delta_{TT,l}(d) \stackrel{\triangle}{=} \min_{0 \leq i \leq N-1} \{G(o'_i, o'_i+d)\}. \tag{9.7}$$

De plus : $\forall d, \exists t$ tel que : $G(t, t+d) = \delta_{TT,l}(d)$.

Ce théorème borne inférieurement le temps de fermeture des portes. Cette borne est la meilleure possible puisqu'elle peut être atteinte.

Démonstration. Pour tout $t \geq 0$, $g(t) = 0$ ou $g(t) = 1$

— Cas $g(t) = 0$:
Alors il existe $i \in [0, N-1]$ tel que la fenêtre TT présente à l'instant t soit la i-ième

fenêtre TT : $o'_i \leq t \leq o_{i+1}$

$$\begin{aligned} G(t, t+d) &= \int_t^{t+d} g(x)\,\mathrm{d}x \\ &= \int_{o'_i}^{t} 0\,\mathrm{d}x + \int_t^{t+d} g(x)\,\mathrm{d}x \\ &= \int_{o'_i}^{t} g(x)\,\mathrm{d}x + \int_t^{o'_i+d} g(x)\,\mathrm{d}x \\ &+ \int_{o'_i+d}^{t+d} g(x)\,\mathrm{d}x \\ &\geq \int_{o'_i}^{o'_i+d} g(x)\,\mathrm{d}x = G(o'_i, o'_i+d) \end{aligned}$$

— Cas $g(t) = 1$:
Alors il existe $i \in [0, N-1]$ tel que t soit dans la i-ième fenêtre TT : $o_i \leq t \leq o'_i$

$$\begin{aligned} G(t, t+d) &= \int_t^{t+d} g(x)\,\mathrm{d}x \\ &= \int_t^{o'_i} g(x)\,\mathrm{d}x + \int_{o'_i}^{t+d} g(x)\,\mathrm{d}x \\ &= \int_t^{o'_i} 1\,\mathrm{d}x + \int_{o'_i}^{t+d} g(x)\,\mathrm{d}x \\ &= \int_{t+d}^{o'_i+d} 1\,\mathrm{d}x + \int_{o'_i}^{t+d} g(x)\,\mathrm{d}x \\ &\geq \int_{t+d}^{o'_i+d} g(x)\,\mathrm{d}x + \int_{o'_i}^{t+d} g(x)\,\mathrm{d}x \\ &\geq G(o'_i, o'_i+d) \end{aligned}$$

Donc pour tout $t \geq 0$, il existe $i \in [0, N-1]$ tel que

$$G(t, t+d) \geq G(o'_i, o'_i+d) \geq \min_{0 \leq i \leq N-1} \left\{ G(o'_i, o'_i+d) \right\}. \tag{9.8}$$

□

Il nous est donc possible de borner la durée de fermeture des portes pour un système TSN/TT-CBS-BE. Lorsque l'on applique les règles de la norme ce temps correspond à la durée du gel du crédit.

À l'inverse lorsque l'on applique les "règles équitables", il faut ajouter à cette durée le temps de "pré-fermeture" ainsi que le temps d'*overhead*. Or les dates de début de "pré-fermeture" et d'*overhead* ne sont pas connues *a priori* cependant elle peuvent être bornées.

Dans le cas où o_i et o'_i sont inconnus mais peuvent être bornés ($\underline{o_i} \leq o_i \leq \overline{o_i}$ et

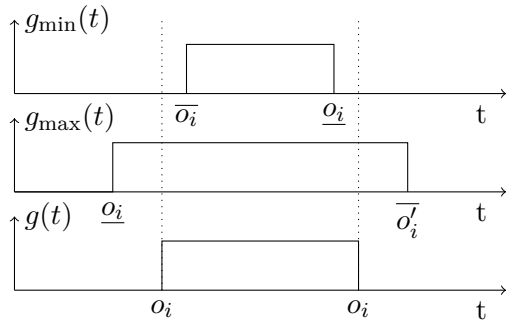


FIGURE 9.6 – Lorsque les dates de début et de fin du gel du crédit sont inconnues : $g_{\min}$ et $g_{\max}$

$\underline{o'_i} \leq o'_i \leq \overline{o'_i}$) comme représenté sur la Figure 9.6 il est possible de définir :

$$g_{min}(t) \leq g(t) \leq g_{max}(t)$$
$$G_{min}(t, t+\triangle t) = \int_t^{t+\triangle t} g_{min}(x)\,\mathrm{d}x$$
$$G_{max}(t, t+\triangle t) = \int_t^{t+\triangle t} g_{max}(x)\,\mathrm{d}x$$

Il est alors possible d'adapter les Théorèmes (26, 27) :

Théorème 28. *Soit $t, d \in \mathbb{R}^+$, alors :*

$$\delta_{TT,u}(d) \leq G(t, t+d) \leq \delta_{TT,u}(d) \tag{9.9}$$

Avec

$$\delta_{TT,u}(d) \stackrel{\triangle}{=} \max_{0\leq i\leq N-1} \left\{G_{max}(\underline{o_i}, \underline{o_i}+d)\right\}, \tag{9.10}$$
$$\delta_{TT,l}(d) \stackrel{\triangle}{=} \min_{0\leq i\leq N-1} \left\{G_{min}(\underline{o'_i}, \underline{o'_i}+d)\right\}. \tag{9.11}$$

Ce théorème permet de borner les temps de gel de crédit. Mais comme ces temps ne sont pas exactement connus on ne peut pas prouver que la borne est atteinte.

Démonstration. Soit $d \geq 0$.

— Pour tout $0 \leq i \leq N-1$:

$$\begin{aligned}
G(o_i, o_i + d) &= \int_{o_i}^{o_i+d} g(x)\,\mathrm{d}x \\
&\leq \int_{o_i}^{o_i+d} g_{max}(x)\,\mathrm{d}x \\
&\leq \int_{o_i}^{\underline{o_i}+d} g_{max}(x)\,\mathrm{d}x + \int_{\underline{o_i}+d}^{o_i+d} g_{max}(x)\,\mathrm{d}x \\
&\leq \int_{o_i}^{\underline{o_i}+d} g_{max}(x)\,\mathrm{d}x + \int_{\underline{o_i}+d}^{o_i+d} 1\,\mathrm{d}x \\
&\leq \int_{o_i}^{\underline{o_i}+d} g_{max}(x)\,\mathrm{d}x + \int_{\underline{o_i}}^{o_i} 1\,\mathrm{d}x \\
&\leq \int_{o_i}^{\underline{o_i}+d} g_{max}(x)\,\mathrm{d}x + \int_{\underline{o_i}}^{o_i} g_{max}(x)\,\mathrm{d}x \\
&\leq G_{max}(\underline{o_i}, \underline{o_i} + d)
\end{aligned}$$

Il est donc possible de déduire que :

$$\max_{0 \leq i \leq N-1} \{G(o_i, o_i + d)\} \leq \max_{0 \leq i \leq N-1} \{G_{max}(\underline{o_i}, \underline{o_i} + d)\}$$

— Pour tout $0 \leq i \leq N-1$:

$$\begin{aligned}
G(o'_i, o'_i + d) &= \int_{o'_i}^{o'_i+d} g(x)\,\mathrm{d}x \\
&\geq \int_{o'_i}^{o'_i+d} g_{min}(x)\,\mathrm{d}x \\
&\geq \int_{o'_i}^{\underline{o'_i}+d} g_{min}(x)\,\mathrm{d}x + \int_{\underline{o'_i}+d}^{o'_i+d} g_{max}(x)\,\mathrm{d}x \\
&\geq \int_{o'_i}^{\underline{o'_i}+d} g_{min}(x)\,\mathrm{d}x + \int_{\underline{o'_i}+d}^{o'_i+d} 0\,\mathrm{d}x \\
&\geq \int_{o'_i}^{\underline{o'_i}+d} g_{min}(x)\,\mathrm{d}x + \int_{\underline{o'_i}}^{o'_i} 0\,\mathrm{d}x \\
&\geq \int_{o'_i}^{\underline{o'_i}+d} g_{min}(x)\,\mathrm{d}x + \int_{\underline{o_i}}^{o_i} g_{min}(x)\,\mathrm{d}x \\
&\geq G_{min}(\underline{o'_i}, \underline{o'_i} + d)
\end{aligned}$$

Et donc on en déduit que :

$$\min_{0 \leq i \leq N-1} \{G(o'_i, o'_i + d)\} \geq \min_{0 \leq i \leq N-1} \left\{G_{min}(\underline{o'_i}, \underline{o'_i} + d)\right\}$$

□

A ce stade nous sommes donc capables de borner inférieurement et supérieurement la durée de gel de crédit même lorsque les dates d'ouvertures et de fermetures ne sont pas connues précisément. Nous allons voir comment l'adapter dans le cas de TSN/TT-CBS-BE.

Non-preemption : une "pré-fermeture" est établie avant chaque fenêtre TT. Dans le pire cas la durée de la "pré-fermeture" est le temps de transmission maximal d'une trame CBS/BE (L_{max}). Donc les offsets sont :

$$\underline{o_i} = o_{i,GCl} - \frac{L_{max}}{C} \qquad \underline{o'_i} = o'_{i,GCl}$$
$$\overline{o_i} = o_{i,GCl} \qquad \overline{o'_i} = o'_{i,GCl}$$

Preemption with HOLD/RELEASE : une "pré-fermeture" réduite est établie avant chaque fenêtre TT. Dans le pire cas la durée de la "pré-fermeture"est le la durée de la *guard band* (L_{GB}). De plus un *overhead* apparaît après la fenêtre TT en cas de préemption, sa longueur est d'au plus L_{over}. Donc les offsets sont :

$$\underline{o_i} = o_{i,GCl} - \frac{L_{GB}}{C} \qquad \underline{o'_i} = o'_{i,GCl}$$
$$\overline{o_i} = o_{i,GCl} \qquad \overline{o'_i} = o'_{i,GCl} + \frac{L_{over}}{C}$$

Il nous est donc possible de borner la durée de gel de crédit pour un système TSN/TT-CBS-BE quelle que soit la politique d'intégration utilisée et quelles que soient les règles suivies.

9.3.2 Calcul des courbes de service résiduel

Les courbes de service présentées ici sont une généralisation de celles présentées dans le Chapitre 8. Celles ci prennent en compte la présence de flux TT mais aussi la possibilité qu'il y ait plus de deux classes CBS comme c'est le cas dans AVB.

Dans la suite nous nous intéresserons à la classe CBS i et calculerons ces différentes courbes de service. De plus dans cette section nous faisons l'hypothèse que le crédit de chaque classe peut être borné.

Courbe de service simple

Théorème 29. *Si c_i^{max} est une borne supérieure du crédit de la classe i, et si $\delta_{TT,u}(t)$ est une borne supérieure du temps de gel de crédit (bornes qui dépendent de la politique d'intégration) alors la courbe de service simple de la classe CBS i ($i \in [1, N_{CBS}]$) est :*

$$\beta_i^{\min}(t) = \frac{C\ idSl_i}{idSl_i - sdSl_i}\left[t - \delta_{TT,u}(t) - \frac{c_i^{max}}{idSl_i}\right]^+_{\uparrow}. \tag{9.12}$$

Dans le cas où $sdSl_i = idSl_i - C$:

$$\beta_i^{\min}(t) = idSl_i\left[t - \delta_{TT,u}(t) - \frac{c_i^{max}}{idSl_i}\right]^+_{\uparrow}. \tag{9.13}$$

Démonstration. Soit t un instant quelconque. Notons $A_i(t)$ et $D_i(t)$ le flux d'arrivée et de départ de la classe CBS i. Soit s la date de début de la dernière *busy period* de la file i où le crédit de la file i était nul, c'est-à-dire pour tout x dans $(s,t]$, soit $c_i(x) < 0$ soit $D_i(x) < A_i(x)$.

À l'instant s, $D_i(s) = A_i(s)$ et $c_i(s) = 0$.

Notons $\triangle t$ l'intervalle $t-s$. Cet intervalle peut être décomposé en $\triangle t = \triangle t^+ + \triangle t^- + \triangle t^0$. $\triangle t^0$ est la somme de tous les intervalles où le crédit est gelé sur $(s,t]$. Durant $\triangle t^0$ le crédit de i est par définition constant. $\triangle t^-$ représente la durée de transmission des messages de la classe CBS i. Durant $\triangle t^-$ le crédit de i diminue à la vitesse $sdSl_i$. Enfin $\triangle t^+$ représente la durée où le crédit de i augmente. La variation de crédit durant l'intervalle $\triangle t$ satisfait :

$$\begin{aligned} c_i(t) - c_i(s) = c_i(t) &= \triangle t^+ idSl_i + \triangle t^- sdSl_i \\ &= (\triangle t - \triangle t^0)\ idSl_i - \triangle t^-\ (idSl_i - sdSl_i). \end{aligned}$$

Ainsi, nous obtenons une relation entre le temps de service de la classe i et la durée de gel de crédit sur tout intervalle $\triangle t$,

$$\triangle t^- = \frac{(\triangle t - \triangle t^0)\ idSl_i - c_i(t)}{idSl_i - sdSl_i}. \tag{9.14}$$

De plus $\delta_{TT,u}(t)$ est une borne supérieure du temps de gel de crédit sur une durée t donc :

$$\triangle t^0 \leq \delta_{TT,u}(\triangle t). \tag{9.15}$$

Par la suite en considérant (9.14) et (9.15), il est possible d'exprimer la variation du flux de départ de la classe i durant l'intervalle $\triangle t$:

$$D_i(t) - D_i(s) = C\ \triangle t^- \geq C\ \frac{(\triangle t - \delta_{TT,u}(\triangle t))\ idSl_i - c_i^{max}}{idSl_i - sdSl_i}.$$

Comme le flux de départ $D_i(t)$ est une fonction positive croissante et sachant que $D_i(s) = A_i(s)$, nous avons :

$$D_i(t) - A_i(s) \geq \left[\frac{C\ idSl_i}{idSl_i - sdSl_i}\left(t - \delta_{TT,u}(\triangle t) - \frac{c_i^{max}}{idSl_i}\right)\right]^+_{\uparrow} = \beta_i^{\min}(t).$$

Par la suite,

$$D_i(t) \geq \beta_i^{\min}(\triangle t) + A_i(s) \geq \inf_{0\leq s\leq t}\left\{A_i(s) + \beta_i^{\min}(t-s)\right\} = (A_i * \beta_i^{\min})(t)$$

Ainsi, $\beta_i^{\min}(t)$ est une courbe de service simple de i. □

Courbe de service strict

Théorème 30. *Si c_i^{max} est une borne supérieure du crédit de la classe i, si c_i^{min} est une borne inférieure du crédit de la classe i et si $\delta_{TT,u}(t)$ est une borne supérieure du temps*

de gel de crédit sur une durée t (bornes qui dépendent de la politique d'intégration) alors la courbe de service strict de la classe CBS i ($i \in [1, N_{CBS}]$) est :

$$\beta_i^{\min}(t) = \frac{C\ idSl_i}{idSl_i - sdSl_i}\left[t - \delta_{TT,u}(t) - \frac{c_i^{max} - c_i^{min}}{idSl_i}\right]_{\uparrow}^{+}. \tag{9.16}$$

Dans le cas où $sdSl_i = idSl_i - C$:

$$\beta_i^{\min}(t) = idSl_i\left[t - \delta_{TT,u}(t) - \frac{c_i^{max} - c_i^{min}}{idSl_i}\right]_{\uparrow}^{+}. \tag{9.17}$$

Démonstration. Notons $A_i(t)$ et $D_i(t)$ le flux d'arrivée et de départ de la classe CBS i. Soit t et s deux instants tels que durant l'intervalle $(s,t]$ la file i ne soit jamais vide, c'est-à-dire pour tout x dans $(s,t]$, $D_i(x) < A_i(x)$.

Notons $\triangle t$ l'intervalle $t-s$. Cet intervalle peut être décomposé en $\triangle t = \triangle t^+ + \triangle t^- + \triangle t^0$. $\triangle t^0$ est la somme de tous les intervalles où le crédit est gelé sur $(s,t]$. Durant $\triangle t^0$ le crédit de i est par définition constant. $\triangle t^-$ représente la durée de transmission des messages de la classe CBS i. Durant $\triangle t^-$ le crédit de i diminue à la vitesse $sdSl_i$. Enfin $\triangle t^+$ représente la durée où le crédit de i augmente. La variation de crédit durant l'intervalle $\triangle t$ satisfait :

$$\begin{aligned} c_i(t) - c_i(s) &= \triangle t^+ idSl_i + \triangle t^- sdSl_i \\ &= (\triangle t - \triangle t^0)\ idSl_i - \triangle t^-\ (idSl_i - sdSl_i). \end{aligned}$$

Ainsi, nous obtenons une relation entre le temps de service de la classe i et la durée de gel de crédit sur tout intervalle $\triangle t$,

$$\triangle t^- = \frac{(\triangle t - \triangle t^0)\ idSl_i - c_i(t) + c_i(s)}{idSl_i - sdSl_i}. \tag{9.18}$$

De plus $\delta_{TT,u}(t)$ est une borne supérieure du temps de gel de crédit sur une durée t donc :

$$\triangle t^0 \leq \delta_{TT,u}(\triangle t). \tag{9.19}$$

Par la suite en considérant (9.18) et (9.19), il est possible d'exprimer la variation du flux de départ de la classe i durant l'intervalle $\triangle t$:

$$D_i(t) - D_i(s) = C\ \triangle t^- \geq C\ \frac{(\triangle t - \delta_{TT,u}(\triangle t))\ idSl_i - c_i^{max} + c_i^{min}}{idSl_i - sdSl_i}.$$

Comme le flux de départ $D_i(t)$ est une fonction positive croissante nous avons :

$$D_i(t) - D_i(s) \geq \left[\frac{C\ idSl_i}{idSl_i - sdSl_i}\left(t - \delta_{TT,u}(t-s) - \frac{c_i^{max} - c_i^{min}}{idSl_i}\right)\right]_{\uparrow}^{+} = \beta_i^{\min}(t-s).$$

Ainsi, $\beta_i^{\min}(t)$ est une courbe de service strict de i. □

Courbe de shaping

Théorème 31. *Si c_i^{max} est une borne supérieure du crédit de la classe i, si c_i^{min} est une borne inférieure du crédit de la classe i et si $\delta_{TT,l}(t)$ est une borne inférieure du temps de gel de crédit sur une durée t (bornes qui dépendent de la politique d'intégration) alors la courbe de* shaping *de la classe CBS i ($i \in [1, N_{CBS}]$) est :*

$$\sigma_i(t) = \frac{C\ idSl_i}{idSl_i - sdSl_i}\left[t - \delta_{TT,l}(t) + \frac{c_i^{max} - c_i^{min}}{idSl_i}\right]_{\uparrow}^{+}. \tag{9.20}$$

Dans le cas où $sdSl_i = idSl_i - C$:

$$\sigma_i(t) = idSl_i\left[t - \delta_{TT,l}(t) + \frac{c_i^{max} - c_i^{min}}{idSl_i}\right]_{\uparrow}^{+}. \tag{9.21}$$

Démonstration. Notons $A_i(t)$ et $D_i(t)$ le flux d'arrivée et de départ de la classe CBS i. Soit t et $t+\triangle t$ deux instants quelconque. $\triangle t$ peut être décomposé en $\triangle t = \triangle t^+ + \triangle t^- + \triangle t^0$. $\triangle t^0$ est la somme de tous les intervalles où le crédit est gelé sur $(s,t]$. Durant $\triangle t^0$ le crédit de i est par définition constant. $\triangle t^-$ représente la durée de transmission des messages de la classe CBS i. Durant $\triangle t^-$ le crédit de i diminue à la vitesse $sdSl_i$. Enfin $\triangle t^+$ représente la durée où le crédit de i augmente. On alors :

$$\begin{aligned}
c_i^{\min} - c_i^{\max} &\leq c(t+\triangle t) - c(t) \\
&\leq \triangle t^+\ idSl_i + \triangle t^-\ sdSl_i + \triangle t^0\ 0 \\
&\leq (\triangle t - \triangle t^- - \triangle t^0)\ idSl_i + \triangle t^-\ sdSl_i \\
&\leq (\triangle t - \triangle t^0)\ idSl_i + \triangle t^-(sdSl_i - idSl_i) \\
&\leq (\triangle t - \delta_{TT,l}(\triangle t))\ idSl_i - \triangle t^-(idSl_i - sdSl_i) \\
&\leq (\triangle t - \delta_{TT,l}(\triangle t))\ idSl_i - (D_i(t+\triangle t) - D_i(t))\ \frac{idSl_i - sdSl_i}{C}
\end{aligned}$$

Comme le flux de départ $D_i(t)$ est une fonction positive croissante nous avons :

$$D_i(t+\triangle t) - D_i(t) \leq \frac{C\ idSl_i}{idSl_i - sdSl_i}\left[t - \delta_{TT,l}(\triangle t) + \frac{c_i^{max} - c_i^{min}}{idSl_i}\right]_{\uparrow}^{+} = \sigma_i(t)$$

Par la suite, on en déduit que pour tous instants t et $t+\triangle t$:

$$D_i(t+\triangle t) \leq \sigma_i(\triangle t) + D_i(t)$$

Et donc que :

$$D_i(t+\triangle t) \leq \inf_{u,s\geq 0, u+s=t+\triangle t}\{\sigma_i(u) + D_i(s)\} = (D_i * \sigma_i)(t+\triangle t)$$

Ainsi, $\sigma_i(t)$ est une courbe de *shaping* de la classe i. □

9.4 Conclusion

Dans ce chapitre nous avons complété la modélisation qui avait été effectuée au Chapitre 8.

Pour cela nous avons tout d'abord présenté les règles, définies par la norme, qui encadrent la transmission des messages ainsi que la variation du crédit des classes CBS lorsque l'on rajoute un aspect *Time-Triggered* via un système de porte du réseau TSN/TT-CBS-BE. Nous avons aussi détaillé les différentes politiques d'intégration disponibles.

Par la suite nous avons montré que ces règles pouvaient conduire à un fonctionnement inattendu du réseau TSN/TT-CBS-BE. Nous avons donc proposé de nouvelles règles qui nous semblent mieux correspondre au comportement désiré.

Pour établir les différentes courbes de service, il était nécessaire que le crédit de chaque classe soit borné inférieurement et supérieurement et que l'on soit capable de calculer ces bornes. C'est ce que nous avons fait, dans un premier temps en nous plaçant dans le cadre de la norme. Puis dans un second temps, nous avons calculé ces bornes, avec nos règles alternatives.

Il était de plus nécessaire d'être capable de borner la durée durant laquelle le crédit des classes CBS est gelé. C'est donc ce que nous avons fait dans la section suivante.

Cela fait, il nous a été possible de construire, à l'aide du Calcul Réseau, un modèle afin de borner le temps de traversée des messages. Pour cela nous avons exprimé les différentes courbes de services offertes par un serveur TSN/TT-CBS-BE aux flux CBS.

Troisième partie

Bilan

Chapitre 10

Bilan

10.1 Résumé des contributions

L'objectif de ce travail était de mettre en place un modèle capable de borner les temps de traversée du réseaux. Afin d'y parvenir nous nous sommes basés sur la méthode d'analyse du Calcul Réseau. Nous nous sommes intéressés en particulier à la modélisation des interactions qui existent entre les messages synchrones et les messages asynchrones. Les modèles présentés dans ce manuscrit prennent en compte les dates d'émission sur le réseau des messages synchrones lors du calcul des bornes supérieures de temps de traversée du réseau des messages asynchrones.

Ce travail a permis tout d'abord dans le Chapitre 6 de proposer une nouvelle façon d'envisager l'utilisation des dates d'émission avec un bus faiblement synchronisé. Il avait déjà été montré que l'utilisation d'offsets réduisait fortement les délais de traversée du bus en réduisant le nombre de contentions. Cependant jusqu'à présent seules deux solutions étaient proposées. La première consistait à synchroniser tous les nœuds du système, il est alors possible d'établir un ordonnancement global et d'éviter les contentions. Cette première solution permettait de minimiser le délai de traversée mais en contre partie était très coûteuse. La deuxième solution consistait à ne pas synchroniser les nœuds et de se limiter à un ordonnancement local. Cela permet tout de même d'éviter les contentions intra-nœuds ce qui diminue le temps de traversée des messages, mais de façon moins efficace qu'avec une horloge commune. L'idée d'un système faiblement synchronisé était d'obtenir un compromis entre ces deux solutions. Un délai réduit en évitant au maximum les contentions mais sans avoir un coût de synchronisation trop important.

Afin d'évaluer l'intérêt que peut avoir l'utilisation d'un système faiblement synchronisé sur le pire temps de traversée d'un message il est nécessaire d'être capable de borner le délai dans une telle situation. Or il n'existait pas jusqu'à présent de modèle capable de prendre en compte de façon satisfaisante cette synchronisation faible. Nous avons donc développé, à l'aide du calcul réseau, un modèle d'un bus faiblement synchronisé. Ce modèle permet non seulement de modéliser un bus parcouru uniquement par des flux synchrones mais aussi des flux asynchrones. De plus nous avons montré comment ce modèle permettait de prendre en compte les possibles erreurs sur le bus ainsi que les messages liés à la synchronisation entre les horloges.

Ce modèle établi il a été possible de mener un nombre important d'expérimentations afin d'évaluer le gain apporté par l'utilisation d'un bus faiblement synchronisé. Ces expériences ont révélé qu'une synchronisation faible permettait une diminution très significative des délais (jusqu'à 40%). Nous avons par la suite montré que cette diminution des délais permet une augmentation de la charge maximale du lien utilisable. La synchronisation faible permet d'utiliser en moyenne près de 70% du lien alors que sans utiliser d'offsets la part maximale du lien que l'on peut utiliser n'atteindra en moyenne même pas les 50%.

Par la suite, dans le Chapitre 7, nous nous sommes intéressés au réseau TTEthernet. Le réseau TTEthernet avec ses trois classes de trafic offre énormément de flexibilité en terme d'organisation de la communication. Cependant les interactions qui existent entre ces différents trafics sont multiples et complexes et il est donc difficile de prévoir l'impact qu'elles peuvent avoir sur le temps de traversée du réseau. En particulier nous avons cherché à évaluer l'impact du trafic *Time-Triggered* sur le trafic *Rate-Constrained*. Dans un contexte temps réel il est nécessaire d'avoir des garanties temporelles sur les messages *Rate-Constrained* et il est donc primordial de connaître l'impact des flux *Time-Triggered* sur les flux *Rate-Constrained*.

Pour répondre à ce problème, nous avons tout d'abord, à l'aide du calcul réseau, développé un nouveau modèle capable de calculer des bornes sur le temps de traversée des messages *Rate-Constrained*. Ce modèle permet de modéliser le trafic *Time-Triggered* et peut être utilisé avec chacune des différentes politiques d'intégration proposées dans TTEthernet.

Ce modèle établi il a été possible de mener un nombre important d'expérimentations afin d'évaluer l'impact qu'ont les flux *Time-Triggered* sur le temps de traversée des flux *Rate-Constrained*. Nous avons pu apprendre de ces expérimentations un grand nombre de points :

— La politique d'intégration *timely block*, en prévenant les contentions pour les flux TT, entraîne une augmentation importante des bornes sur le temps de traversée en comparaison avec la politique de *shuffling*. Cela va jusqu'à doubler les bornes sur les délais lorsque la proportion de flux TT dépasse les 50%.

— La politique d'ordonnancement utilisée par TTPlan vise à lisser le trafic en répartissant les flux TT dans le temps. Ceci a pour conséquence de diminuer les bornes sur les délais des flux RC, lorsque la proportion de flux TT augmente. Cependant cet effet n'est vrai qu'en cas de *shuffling*, politique qui augmente le temps de traversée maximum des flux TT. Dans le cas de *timely block* il faut qu'une part importante du trafic soit TT pour que cet effet ne soit pas négligeable.

— La charge du réseau a un rôle majeur sur les interactions entre les flux TT et RC. En effet lorsque le nombre de flux est plus important les interactions entre flux TT et RC seront d'autant plus présentes. En cas de *timely block* il y aura donc d'autant plus de bande passante non utilisée afin de garantir les dates d'émission des messages TT ce qui augmente les bornes sur les délais.

Enfin dans les Chapitres 8 et 9 nous nous sommes intéressés au réseau TSN. TSN désigne un groupe de travail de l'IEEE, qui complète Ethernet. Ce groupe est actuelle-

ment toujours en cours de travaux, la norme TSN n'est donc pas entièrement définie. De plus, l'architecture TSN admet beaucoup de paramètres, conduisant à de nombreux comportements différents. Il n'existe donc pour l'instant que peu d'études sur le sujet, d'où l'importance d'être capable de bien comprendre le fonctionnement du réseau TSN afin de mettre en place une modélisation de ce dernier.

Dans ces deux chapitres nous nous sommes donc basés sur la norme TSN, afin d'en dégager les règles de fonctionnements que nous avons présenté en détail. Nous avons proposé différentes façon d'utiliser ces norme à travers deux classes particulières de TSN : TSN/CBS-BE et TSN/TT-CBS-BE. Nous avons de plus montré en quoi la norme telle qu'elle existe actuellement pouvait conduire à un fonctionnement inattendu et nous avons proposé une possible modification. Nous avons de plus modéliser en calcul réseau, l'ensemble de ces deux classes.

La première contribution a consisté à définir une classe particulière du réseau TSN que nous avons désigné par TSN/CBS-BE. Ce type de réseau est une généralisation de réseau AVB puisqu'il ajoute la possibilité d'avoir plus de deux files CBS. Nous avons détaillé le fonctionnement du crédit, et avons par la suite démontré sous quelles conditions ce dernier était borné et quelles étaient ses bornes dans ce cas. Cela fait il nous a été possible de construire, à l'aide du Calcul Réseau, un modèle afin de borner le temps de traversée des messages, pour cela nous avons calculé les différentes courbes de service pour chaque flux CBS.

Dans un second temps nous avons considéré une nouvelle classe particulière du réseau TSN que nous avons désigné par TSN/TT-CBS-BE. Ce type de réseau ajoute non seulement la possibilité d'avoir plus de deux files CBS mais il ajoute aussi une file *Time-Triggered*. Nous avons tout d'abord expliqué comment était géré l'aspect *Time-Triggered* dans TNS. Puis nous avons présenté les modifications qu'entraînait la présence d'un flux *Time-Triggered* pour l'ensemble des autres flux. Nous avons alors montré que la norme actuelle pouvait conduire à un fonctionnement inattendu. Nous avons donc proposé un nouvel ensemble de règles et nous avons montré en quoi ces nouvelles règles nous semble mieux correspondre au comportement attendu. Nous avons par la suite développé un modèle complet permettant d'analyser le fonctionnement de TSN/TT-CBS-BE quelle que que soit la politique d'intégration utilisée dans le cas de la norme mais aussi dans le cas de nos règles alternatives. Pour cela nous avons non seulement borné supérieurement le crédit dans TSN/TT-CBS-BE mais nous avons aussi borné le temps durant lequel le crédit est gelé suite aux interactions avec les flux *Time-Triggered*.

Si l'on trace le bilan de tout ceci, ces travaux ont permis de développer de nouveaux résultats de calcul réseau afin de prendre en compte la synchronisation dans un réseau. Nous avons de plus montré, sur un bus, qu'une synchronisation faible permet une réduction très importante du délai maximal des messages à faible coût. Nous avons aussi montré que l'utilisation de flux synchrones pouvait grandement impacter le reste du trafic. Nous avons pu montrer que cette influence est complexe, d'une part l'utilisation de flux synchrones permet de lisser le trafic et donc réduire les délais et d'autre part les politiques d'intégration augmentent les délais des flux asynchrones. Ce travail permet de modéliser finement les interactions qui existent entre flux synchrones et flux asynchrones.

10.2 Perspectives

Un point important qui n'a été qu'évoqué au cours de ce travail est la nécessité d'une synchronisation entre la partie application et la partie réseau. En effet dans cette étude nous nous sommes concentrés sur les temps de traversée du réseau cependant en pratique le temps de traversée qui importe est le temps de traversée d'application à application. Dans le cas de messages asynchrones cela ne pose jamais de soucis puisque dès qu'un message est produit par la partie application il peut être transmis à la partie réseau. Les messages synchrones eux doivent respecter des dates d'émission au niveau de la partie réseau, si la date de production du message ne correspond pas à la date d'émission du message alors celui ci devra attendre jusqu'à sa prochaine date d'émission. Ce délai supplémentaire doit être pris en compte.

Cependant nous avons montré comment, dans le cas du bus CAN, une synchronisation faible permet d'obtenir une réduction très importante de l'ensemble des temps de traversée des messages. Il n'est pas nécessaire de synchroniser parfaitement tous les éléments sur le bus pour bénéficier de la réduction des délais, il suffit de synchroniser faiblement les applications entre elles. Une piste de travail qui nous semble très prometteuse à ce stade est d'étendre ce résultat que l'on a obtenu sur un bus à un réseau commuté. Ainsi la synchronisation des seules applications serait suffisante pour bénéficier des avantages des flux synchrones et la synchronisation des commutateurs ne serait plus nécessaire. En effet si les dates d'émission sur le réseau de l'ensemble des messages sont connues il devrait être possible d'en déduire les dates d'émission au niveau des commutateurs à une certaine précision près. Ce cas est similaire à celui observé lorsque plusieurs nœuds faiblement synchronisé communique sur un même lien où la date d'émission des messages est alors connue à une précision près. Or nous pensons donc qu'il est possible d'adapter notre modèle développé pour le cas d'un bus faiblement synchronisé, à un réseau commuté dont seules les applications sont synchronisées. C'est-à-dire que la date d'émission des messages au niveau de chaque calculateur serait connue, et celles au niveau des calculateurs le serait à une certaine précision près. L'intérêt serait de ne pas avoir à utiliser de commutateurs dédiés capable d'être synchronisés au reste du réseau (ce qui a un coût) tout en bénéficiant des avantages procurés par les messages synchrones.

Enfin la norme TSN possède de très nombreuses possibilités, sa flexibilité permet de définir de nombreux types de réseau. C'est ce que nous avons fait au cours de cette étude en proposant deux cas d'utilisation : TSN/CBS-BE et TSN/TT-CBS-BE. Ces types de réseau sont une généralisation de réseau AVB puisqu'ils ajoutent la possibilité d'avoir plus de deux files CBS ainsi que la possibilité d'avoir une file *Time-Triggered*. Mais de nombreuses autres utilisations sont possibles, le *credit based shaper* n'est pas le seul algorithme de *shaping* qui existe, on peut par exemple citer le *burst Limiting shaper* et le *peristaltic shaper*. De plus dans TSN/TT-CBS-BE nous avons considéré le cas où les fenêtres *Time-Triggered* étaient exclusives, c'est à dire que les fenêtres des classes CBS étaient ouvertes lorsque celles de la classe *Time-Triggered* était fermée et inversement. Or cela n'est pas imposé par la norme, il serait possible d'avoir un flux plus prioritaire que les classes CBS qui partage le lien avec elles par exemple. Dans cette étude nous avons proposé un cas d'utilisation de TSN et avons montré qu'il était possible de le modéliser à l'aide du calcul réseau. Toutes ces autres utilisations nécessitent elles aussi une modélisation afin d'être capable d'évaluer leurs intérêts.

Glossaire

AFDX : *Avionics Full DupleX Switched Ethernet*
AVB : *Audio Video Bridging*
BAG : *Bandwidth Allocation Gap*
BE : *Best Effort*
CAN : *Controller Area Network*
CAN-FD : *Controller Area Network with Flexible Data rate*
CBS : *Credit Based Shaper*
CSMA/CD : *Carrier Sense Multiple Access / Collision Detection*
FIFO : *First In, First Out*
MIMO : *Multiple Input, Multiple Output*
MFS : *Maximum Frame Size*
NP-SP : *Non-Preemptive Static-Priority*
PCF : *Protocol Control Frame*
PTP : *Precision Time Protocol*
RC : *Rate Constrained*
SFA : *Separated Flow Analysis*
TDMA : *Time Division Multiple Access*
TFA : *Total Flow Analysis*
TT : *Time-Triggered*
TTP : *Time-Triggered Protocol*
TSN : *Time Sensitive Network*
VL : *Virtual Link*

Bibliographie

[802.1BA, 2011] 802.1BA (2011). Local and metropolitan area networks - Audio Video Bridging (AVB) Systems. IEEE Standard IEEE 802.1BA, IEEE. http://www.ieee802.org/1/pages/802.1ba.html.

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

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

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

[802.1Qbu, 2016] 802.1Qbu (2016). IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks – Amendment 26 : Frame Preemption. IEEE Standard 802.1Qbu, IEEE.

[802.1Qbv, 2015] 802.1Qbv (2015). IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks–Amendment 25 : Enhancements for Scheduled Traffic. IEEE Standard 802.1Qbv, IEEE.

[802.3br, 2016] 802.3br (2016). IEEE Standard for Ethernet Amendment 5 : Specification and Management Parameters for Interspersing Express Traffic. IEEE Standard 802.3br, IEEE.

[AFDX, 2005] AFDX (2005). Aircraft data network part 7 : Avionics full duplex switched ethernet (AFDX) network. Technical report.

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

[Alur and Dill, 1994] Alur, R. and Dill, D. L. (1994). A theory of timed automata. *Theoretical computer science*, 126(2) :183–235.

[ARINC 825, 2011] ARINC 825 (2011). General standardization of CAN (Controller Area Network) bus protocol for airborne use. *ARINC Specification 825-2*.

[Bauer et al., 2009] Bauer, H., Scharbarg, J.-L., and Fraboul, C. (2009). Applying and optimizing trajectory approach for performance evaluation of AFDX avionics network. In *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*, pages 1–8. IEEE.

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

[Bondorf, 2015] Bondorf, Steffen abd Schmitt, J. B. (2015). Calculating accurate end-to-end delay bounds – you better known your cross-traffic. In *Proc. of the 9th EAI Int. Conf. on Performance Evaluation Methodologies and Tools (ValueTools 2015)*, Berlin, Germany.

[Bondorf, 2017] Bondorf, S. (2017). Better bounds by worse assumptions – improving network calculus accuracy by adding pessimism to the network model. In *Proc. of the IEEE Int. Conference on Communications (ICC 2017), Symposium on Communications QoS, Reliability and Modeling (CQRM)*, Paris. IEEE.

[Bondorf and Schmitt, 2016] Bondorf, S. and Schmitt, J. (2016). Improving cross-traffic bounds in feed-forward networks – there is a job for everyone. In *Proc. of the 18th Int. GI/ITG Conf. on "Measurement, Modelling and Evaluation of Computing Systems" and "Dependability and Fault Tolerance" (MMB&DFT 2016)*, Munster, Deutchland.

[Bosch et al., 1991] Bosch, R. et al. (1991). CAN specification version 2.0. *Rober Bousch GmbH, Postfach*, 300240 :72.

[Bouillard et al., 2018] Bouillard, A., Boyer, M., and Le Corronc, E. (2018). *Deterministic Network Calculus : From Theory to Practical Implementation*. Number ISBN : 978-1-119-56341-9. John Wiley and Sons.

[Bouillard et al., 2009] Bouillard, A., Jouhet, L., and Thierry, E. (2009). Service curves in Network Calculus : dos and don'ts. Research Report RR-7094, INRIA.

[Bouillard et al., 2010] Bouillard, A., Jouhet, L., and Thierry, E. (2010). Tight performance bounds in the worst-case analysis of feed-forward networks. In *Proceedings of the 29th Conference on Computer Communications (INFOCOM 2010)*, pages 1–9.

[Bouillard and Stea, 2014] Bouillard, A. and Stea, G. (2014). Exact worst-case delay for FIFO-multiplexing feed-forward networks. *IEEE/ACM Transactions on Networking*.

[Bouillard and Thierry, 2016] Bouillard, A. and Thierry, É. (2016). Tight performance bounds in the worst-case analysis of feed-forward networks. *Discrete Event Dynamic Systems*, 26(3) :383–411.

[Boyer et al., 2016] Boyer, M., Daigmorte, H., Navet, N., and Migge, J. (2016). Performance impact of the interactions between time-triggered and rate-constrained transmissions in TTEthernet. In *Proceedings of the 8th European Congress on Embedded Real Time Software and Systems*.

[Boyer et al., 2013a] Boyer, M., Dufour, G., and Santinelli, L. (2013a). Continuity for network calculus. In *Proceedings of the 21st International conference on Real-Time Networks and Systems*, pages 235–244. ACM.

[Boyer et al., 2011] Boyer, M., Migge, J., and Navet, N. (2011). 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. ACM.

[Boyer et al., 2013b] Boyer, M., Navet, N., Fumey, M., Migge, J., and Havet, L. (2013b). 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.

[Boyer and Roux, 2016] Boyer, M. and Roux, P. (2016). Embedding network calculus and event stream theory in a common model. In *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*, pages 1–8. IEEE.

[Boyer et al., 2012] Boyer, M., Stea, G., and Sofack, W. M. (2012). Deficit round robin with network calculus. In *Performance Evaluation Methodologies and Tools (VALUETOOLS), 2012 6th International Conference on*, pages 138–147. IEEE.

[Buttle, 2012] Buttle, D. (2012). Real-time in the prime-time. In *Keynote speech at the 24th Euromicro Conference on Real-Time Systems*.

[Chan, 2007] Chan, T. S. (2007). Time-Division Multiple Access. *Handbook of Computer Networks : LANs, MANs, WANs, the Internet, and Global, Cellular, and Wireless Networks, Volume 2*, pages 769–778.

[Chang, 2000] Chang, C.-S. (2000). *Performance Guarantees in Communication Networks*. Springer Science & Business Media.

[Cruz, 1991a] Cruz, R. L. (1991a). A calculus for network delay. I. Network elements in isolation. *IEEE Transactions on information theory*, 37(1) :114–131.

[Cruz, 1991b] Cruz, R. L. (1991b). A calculus for network delay. II. Network analysis. *IEEE Transactions on information theory*, 37(1) :132–141.

[Daigmorte and Boyer, 2016] Daigmorte, H. and Boyer, M. (2016). Traversal time for weakly synchronized CAN bus. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, pages 35–44. ACM.

[Daigmorte and Boyer, 2017] Daigmorte, H. and Boyer, M. (2017). Evaluation of admissible CAN bus load with weak synchronization mechanism. In *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, pages 277–286. ACM.

[Daigmorte et al., 2017] Daigmorte, H., Boyer, M., and Migge, J. (2017). Reducing CAN latencies by use of weak synchronization between stations. In *Proceedings of the 16th International CAN Conference*.

[Daigmorte et al., 2018] Daigmorte, H., Boyer, M., and Zhao, L. (2018). Modelling in network calculus a TSN architecture mixing Time-Triggered, Credit Based Shaper and Best-Effort queues. Technical report.

[Davis and Burns, 2009] Davis, R. I. and Burns, A. (2009). Robust priority assignment for messages on Controller Area Network (CAN). *Real-Time Systems*, 41(2) :152–180.

[Davis et al., 2007] Davis, R. I., Burns, A., Bril, R. J., and Lukkien, J. J. (2007). Controller Area Network (CAN) schedulability analysis : Refuted, revisited and revised. *Real-Time Systems*, 35(3) :239–272.

[Davis et al., 2011] Davis, R. I., Kollmann, S., Pollex, V., and Slomka, F. (2011). Controller Area Network (CAN) schedulability analysis with FIFO queues. In *Real-Time Systems (ECRTS), 2011 23rd Euromicro Conference on*, pages 45–56. IEEE.

[Davis et al., 2013] Davis, R. I., Kollmann, S., Pollex, V., and Slomka, F. (2013). Schedulability analysis for Controller Area Network (CAN) with FIFO queues priority queues and gateways. *Real-Time Systems*, 49(1) :73–116.

[Davis and Navet, 2012] Davis, R. I. and Navet, N. (2012). Controller Area Network (CAN) schedulability analysis for messages with arbitrary deadlines in FIFO and work-conserving queues. In *Factory Communication Systems (WFCS), 2012 9th IEEE International Workshop on*, pages 33–42. IEEE.

[De Azua and Boyer, 2014] De Azua, J. A. R. and Boyer, M. (2014). Complete Modelling of AVB in Network Calculus Framework. In *Proceedings of the 22Nd International Conference on Real-Time Networks and Systems*, RTNS '14, pages 55 :55–55 :64, New York, NY, USA. ACM.

[Ermont et al., 2006] Ermont, J., Scharbarg, J.-L., and Fraboul, C. (2006). Worst-case analysis of a mixed CAN/Switched Ethernet architecture. In *Proc. of the Real-Time and Network System Conference.*

[Ethernet, 2015] Ethernet (2015). 802.3 Standard for Ethernet. IEEE.

[FlexRay Consortium, 2005] FlexRay Consortium (2005). FlexRay Protocol Specification V2. 1 Rev. *FlexRay Consortium.*

[Frances et al., 2006] Frances, F., Fraboul, C., and Grieu, J. (2006). Using network calculus to optimize AFDX network. In *Proceeding of the 3thd European congress on Embedded Real Time Software (ERTS06)*, Toulouse.

[George et al., 1996] George, L., Rivierre, N., and Spuri, M. (1996). Preemptive and non-preemptive real-time uniprocessor scheduling. Technical report, INRIA.

[Georges et al., 2011] Georges, J.-P., Divoux, T., and Rondeau, E. (2011). 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. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

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

[Grenier et al., 2006] Grenier, M., Goossens, J., and Navet, N. (2006). Near-Optimal Fixed Priority Preemptive Scheduling of Offset Free Systems. In *14th International Conference on Real-Time and Networks Systems (RTNS'06)*, pages 35–42, Poitiers/France.

[Grenier et al., 2008] Grenier, M., Havet, L., and Navet, N. (2008). Pushing the limits of CAN-scheduling frames with offsets provides a major performance boost. In *4th European Congress on Embedded Real Time Software (ERTS 2008).*

[Gresser, 1993] Gresser, K. (1993). An Event Model for Deadline Verification of Hard Real-Time Systems. In *Proc. of the Fifth Euromicro Workshop on Real-Time Systems.*, pages 118–123.

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

[Hartwich et al., 2012] Hartwich, F. et al. (2012). CAN with flexible data-rate. In *Proc. iCC*, pages 1–9.

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

[Kemayo et al., 2015] Kemayo, G., Benammar, N., Ridouard, F., Bauer, H., and Richard, P. (2015). Improving AFDX end-to-end delays analysis. In *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*, pages 1–8. IEEE.

[Kemayo et al., 2012] Kemayo, G., Ridouard, F., Bauer, H., and Richard, P. (2012). A Forward end-to-end delays analysis to packet switched networks. In *Proc. of the 22nd Int. Conf. on Real-Time Networks and Systems (RTNS 2014)*, Versailles, France.

[Kemayo et al., 2013] Kemayo, G., Ridouard, F., Bauer, H., and Richard, P. (2013). Optimistic problems in the trajectory approach in FIFO context. In *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*, pages 1–8. IEEE.

[Klehmet et al., 2008] Klehmet, U., Herpel, T., Hielscher, K.-S., and German, R. (2008). Delay bounds for CAN communication in automotive applications. In *Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2008 14th GI/ITG Conference-*, pages 1–15. VDE.

[Kopetz et al., 2005] Kopetz, H., Ademaj, A., Grillinger, P., and Steinhammer, K. (2005). The time-triggered Ethernet (TTE) design. In *Proc. of the 8th IEEE Int. Symp. on Object-Oriented Real-Time Distributed Computing (ISORC 2005)*. IEEE.

[Kopetz and Grunsteidl, 1993] Kopetz, H. and Grunsteidl, G. (1993). TTP-A time-triggered protocol for fault-tolerant real-time systems. In *Fault-Tolerant Computing, 1993. FTCS-23. Digest of Papers., The Twenty-Third International Symposium on*, pages 524–533. IEEE.

[Krákora et al., 2004] Krákora, J., Waszniowski, L., Pisa, P., and Hanzálek, Z. (2004). Timed Automata Approach to Real Time Distributed System Verification. In *Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on*, pages 407–410. IEEE.

[Le Boudec and Thiran, 2001] Le Boudec, J.-Y. and Thiran, P. (2001). *Network calculus : a theory of deterministic queuing systems for the internet*, volume 2050. Springer Science & Business Media.

[Lee et al., 2005] Lee, K., Eidson, J. C., Weibel, H., and Mohl, D. (2005). IEEE 1588-standard for a precision clock synchronization protocol for networked measurement and control systems. In *Conference on IEEE*, volume 1588, page 2.

[Leen and Heffernan, 2002] Leen, G. and Heffernan, D. (2002). TTCAN : a new time-triggered controller area network. *Microprocessors and Microsystems*, 26(2) :77–94.

[Leung and Whitehead, 1982] Leung, J. Y.-T. and Whitehead, J. (1982). On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance evaluation*, 2(4) :237–250.

[Li et al., 2014] Li, X., Cros, O., and George, L. (2014). The Trajectory approach for AFDX FIFO networks revisited and corrected. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014 IEEE 20th International Conference on*, pages 1–10. IEEE.

[Lian et al., 2002] Lian, F.-L., Moyne, J., and Tilbury, D. (2002). Network Design Consideration for Distributed Control Systems. *IEEE Transactions on Control Systems Technology*, 10(2) :297–307.

[Liu and Layland, 1973] Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1) :46–61.

[Martin, 2004] Martin, S. (2004). *Maîtrise de la dimension temporelle de la qualité de service dans les réseaux*. PhD thesis, Université Paris XII Val de Marne.

[Mifdaoui and Leydier, 2017] Mifdaoui, A. and Leydier, T. (2017). Beyond the Accuracy-Complexity Tradeoffs of CompositionalAnalyses using Network Calculus for Complex Networks. In *10th International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (co-located with RTSS 2017)*, pages pp. 1–8, Paris, France.

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

[Scharbarg et al., 2009] Scharbarg, J.-L., Ridouard, F., and Fraboul, C. (2009). A probabilistic analysis of end-to-end delays on an AFDX avionic network. *IEEE transactions on industrial informatics*, 5(1) :38–49.

[Schmitt and Zdarsky, 2006] Schmitt, J. B. and Zdarsky, F. A. (2006). 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*. ACM.

[Sofack, 2014] Sofack, W. M. (2014). *Amélioration des délais de traversée pire cas des réseaux embarqués à l'aide du calcul réseau).* PhD thesis, Institut Supérieur de l'Aéronautique et de l'Espace, Toulouse, France.

[Sofack and Boyer, 2012] Sofack, W. M. and Boyer, M. (2012). Non preemptive static priority with network calculus : Enhancement. In *International GI/ITG Conference on Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*, pages 258–272. Springer.

[Steiner et al., 2009] Steiner, W., Bauer, G., Hall, B., Paulitsch, M., and Varadarajan, S. (2009). TTEthernet dataflow concept. In *Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium on*, pages 319–322. IEEE.

[Talbot and Ren, 2009] Talbot, S. C. and Ren, S. (2009). Comparision of fieldbus systems can, ttcan, flexray and lin in passenger vehicles. In *Distributed Computing Systems Workshops, 2009. ICDCS Workshops' 09. 29th IEEE International Conference on*, pages 26–31. IEEE.

[Thiele and Ernst, 2016] Thiele, D. and Ernst, R. (2016). Formal Worst-Case Performance Analysis of Time-Sensitive Ethernet with Frame Preemption. In *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*, pages 1–9. IEEE.

[Tindell and Burns, 1994] Tindell, K. and Burns, A. (1994). Guaranteed message latencies for distributed safety-critical hard real-time control networks. *Dept. of Computer Science, University of York*.

[Tindell et al., 1994] Tindell, K., Hanssmon, H., and Wellings, A. J. (1994). Analysing Real-Time Communications : Controller Area Network (CAN). In *RTSS*, pages 259–263.

[Witsch et al., 2006] Witsch, D., Vogel-Heuser, B., Faure, J.-M., and Marsal, G. (2006). Performance analysis of industrial Ethernet networks by means of timed model-checking. In *12th IFAC Symposium on Information Control Problems in Manufacturing, INCOM 2006, Saint-Etienne (France), May 2006*, pages pp–101.

[Yomsi et al., 2012] Yomsi, P. M., Bertrand, D., Navet, N., and Davis, R. I. (2012). Controller Area Network (CAN) : Response time analysis with offsets. In *Factory Communication Systems (WFCS), 2012 9th IEEE International Workshop on*, pages 43–52. IEEE.

[Zeng et al., 2010] Zeng, H., Di Natale, M., Giusto, P., and Sangiovanni-Vincentelli, A. (2010). Using Statistical Methods to Compute the Probability Distribution of Message Response Time in Controller Area Network. *IEEE Transactions on Industrial Informatics*, 6(4) :678–691.

Résumé

Les systèmes embarqués complexes (avions, satellites, drones...) contiennent de plus en plus de calculateurs. Désormais ce sont des dizaines voire des centaines de calculateurs qui communiquent à travers un réseau partagé. Une fonction est réalisée par la collaboration d'un ensemble de calculateurs qui s'échangent un nombre croissant d'informations.

Dans un contexte de temps réel embarqué, il faut non seulement garantir que ces informations échangées sont correctes mais il faut aussi garantir qu'elles vérifient leurs contraintes temporelles. Du point de vue du réseau cela signifie qu'une information doit être échangée en respectant les délais qui lui sont imposés. Ceci implique de pouvoir borner le temps de traversée du réseau de chaque message afin de vérifier qu'il arrive dans les temps. Or les systèmes embarqués étant de plus en plus complexes et le nombre d'informations échangées étant en constante augmentation, cette borne est de plus en plus difficile à calculer. De plus il est important que cette borne soit le moins pessimiste possible afin d'éviter que le système soit surdimensionné.

L'objectif de ce travail est de mettre en place un modèle capable de calculer ces bornes. Afin d'y parvenir nous nous sommes basés sur la méthode d'analyse du Calcul Réseau. Ce travail s'est en particulier attardé sur la modélisation des interactions qui existent entre les messages synchrones et les messages asynchrones. Les modèles présentés dans ce manuscrit prennent en compte les dates d'émission sur le réseau des messages synchrones lors du calcul des bornes supérieures de temps de traversée des messages asynchrones. Les principales contributions apportées par ce manuscrit sont :

1. la présentation d'une nouvelle façon d'envisager l'utilisation des dates d'émission sur le bus CAN : la synchronisation faible. Ainsi que la modélisation complète d'un tel système et enfin l'évaluation du gain apporté par cette solution.
2. une modélisation complète du réseau TTEthernet permettant d'évaluer finement l'impact des flux synchrones sur le temps de traversée des flux asynchrones.
3. une présentation de l'utilisation de la synchronisation dans le réseau TSN ainsi qu'un modèle complet permettant d'analyser cette nouvelle technologie.

Abstract

Complex embedded systems (planes, satellites, drones ...) contain more and more calculators. From now on, these are tens or even hundreds of calculators that communicate through a shared network. A function is achieved by the collaboration of a set of devices that exchange a growing number of information.

In an embedded real-time context, it must be ensured that these informations exchanged are correct but it must also be ensured that they verify their temporal constraints. From the network point of view, this means that informations must be exchanged respecting their deadlines. This implies being able to upper bound the traversal time of the network of each message in order to verify that it arrives in time. However, as embedded systems are more and more complex and as the amount of information exchanged is constantly increasing, this bound is increasingly difficult to compute. Furthermore, it is important that this upper bound to be the least pessimistic possible to avoid an oversized system. The goal of this work is to develop new methods of analysis in order to be able to compute these bounds.

In order to achieve this, we used the Network Calculus method of analysis. This work focuses on the modeling of interactions between synchronous messages and asynchronous messages. The models presented in this work take into account the transmission dates on the network of synchronous messages when calculating the upper bounds of traversal time of the asynchronous messages. The main contributions are :

1. the presentation of a new way of considering the use of the dates of emission on the CAN bus : the weak synchronization. As well as the complete modeling of such a system and finally the evaluation of the gain provided by this solution.
2. a complete modeling of the TTEthernet network allowing to evaluate the impact of the synchronous flows on the traversal time of the asynchronous flows.
3. a presentation of the use of synchronization in the TSN network and a complete model for analyzing this new technology.