Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace (ISAE)

Présentée et soutenue par :
William MANGOUA SOFACK

le lundi 30 juin 2014

Titre :

Amélioration des délais de traversée pire cas des réseaux embarqués à l'aide du calcul réseau

---

École doctorale et discipline ou spécialité :
ED MITT : Réseaux, télécom, système et architecture

Unité de recherche :
Équipe d'accueil ISAE-ONERA MOIS

Directeur(s) de Thèse :

M. Frédéric BONIOL (directeur de thèse)
M. Marc BOYER (co-directeur de thèse)

Jury :

M. Christian FRABOUL, Professeur, ENSEEIHT/INPT - Président du jury
M. Bruno GAUJAL, Directeur de recherche, INRIA - Rapporteur
M. Ye-Qiong SONG, Professeur, Université de Lorraine - Rapporteur
Mme Anne BOUILLARD, Maître de conférence, École normale supérieure
M. Marc FUMEY, Ingénieur, Thales avionics
M. Frédéric BONIOL, Ingénieur de recherche, ONERA - Directeur de thèse
M. Marc BOYER, Ingénieur de recherche, ONERA - Co-directeur de thèse

Remerciements

Mes premiers remerciements vont à mes encadrants Marc Boyer et Frédéric Boniol pour le temps qu'il m'ont consacré durant ce travail, leur soutien, leurs conseils, leur disponibilité et leurs aides précieuses qui ont été nécessaires pour aller jusqu'au bout de ce travail de thèse.

Merci à Josette Brial pour son accueil dans le groupe et pour avoir été la mamie de Ted et Hermann le jour de la soutenance.

Je tiens à remercier tous les membres du DTIM, particulièrement Claire Pagetti et Virginie Wiels pour leur soutien et leur accueil dans le groupe.

Je remercie Anne Bouillard, Eric Thierry et Giovanni Stea pour leurs conseils scientifiques.

Mes remerciements vont aussi aux membres de ma grande famille. Je pense en particulier à Djouaka ma maman, mon épouse Marie Ange et mes deux enfants Ted et Hermann pour leurs encouragements, leurs prières et leur présence malgré la distance.

Merci à tous ceux qui sont venus m'encourager et me soutenir lors de la soutenance.

Je ne saurais terminer sans remercier mes amis, les Bikunda.

Table des matières

Introduction générale

1 - Présentation du sujet

1.1 Nécessité du déterminisme dans les réseaux embarqués temps réel

Les systèmes embarqués sont de plus en plus des systèmes communicants. Par exemple, dans les avions modernes, on trouve plusieurs applications qui s'échangent des informations à travers le backbone avionique. Bien entendu, un comportement correct de ces applications se doit d'être garanti. Du point de vue du réseau, cela signifie que lorsqu'une information y est transmise, il faut que cette information arrive à destination dans les délais. Dans un contexte de temps réel embarqué, un résultat juste mais hors-délai est un résultat inutilisable. D'où la nécessité de se doter des moyens d'évaluation du délai pire cas d'un bout du réseau à l'autre bout.

De nombreux "réseaux temps réels" ont été développés au cours des années, tous utilisant des technologies plutôt spécifiques (cf. I.2.4, p.14). La technologie AFDX a été un changement de paradigme puisque dans son concept, il s'agit d'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.

Le réseau AFDX est basé sur un assemblage de topologies en étoile dans laquelle les commutateurs sont reliés par des liaisons Ethernet commuté full duplex. C'est un réseau dans lequel il n'y a pas de collision du fait de la suppression de la méthode d'accès au média CSMA/CD, et donc très peu de perte de paquet au niveau des liaisons. Cependant, puisque les commutateurs ont des files d'attente de capacité bornée, ils peuvent perdre des paquets suite à des conges-

tions. Ce qui revient à dire que le problème de perte de paquets a été déplacé au niveau des commutateurs. Un tel réseau n'est évidemment pas déterministe. Pour adapter le réseau AFDX au contexte embarqué avionique, la norme ARINC 664 propose d'imposer un contrôle de flux sur les paquets entrants en introduisant la notion de liens virtuel (Virtual Link : VL). Cette limitation ne garanti cependant pas le déterminisme du réseau. Finalement, pour garantir un comportement correct des applications qui s'exécutent sur le réseau, une garantie sur la borne du délai de traversée de bout en bout doit être fournie.

1.2 Évaluation du délai de bout en bout

Pour tout équipement embarqué, il faut faire passer au système avionique l'épreuve de la qualification. Les exigences de cette qualification sont d'autant plus hautes que les données sont critiques. La qualification du système avionique utilise comme hypothèse le fait que le réseau fournisse une certaine qualité de service et utilise des méthodes formelles pour en déduire le respect des exigences [59]. Les exigences que l'on cherche à assurer pour les applications s'appuie principalement sur les besoins des applications. On veut pouvoir garantir :

— un délai de traversée du réseau borné, pour préserver le temps de réponse des applications
— des pertes de trames très faibles, car les contraintes temps réel fortes interdisent les retransmissions.

Pour apporter ces garanties, qui ne découlent pas directement de l'architecture du réseau, il est nécessaire de mettre en place une méthode de preuve de déterminisme.

Lorsque plusieurs flux partagent un réseau AFDX, ce délai devient complexe à évaluer, car à la latence technologique (dépendant des équipement utilisés) et au temps de transmission proprement dit (variant en fonction du débit des liens) vient s'ajouter un temps d'attente dans les files de commutateurs qui dépend de la présence ou de l'absence d'autres flux utilisant le même support de transmission en sortie d'un commutateur [13]. Le délai de bout en bout pire cas doit donc tenir compte des circonstances les plus défavorables sur l'ensemble du parcours d'une trame dans un réseau. Différentes méthodes proposent de calculer une borne supérieure de ce délai pire cas. Dans le cas de l'AFDX embarqué dans l'A380 et l'A350, c'est l'approche par calcul réseau [49, 50] qui a permis de déterminer ces bornes pire cas, utilisées pour la qualification du réseau [59]. Cette approche a également servi à dimensionner tous les éléments du réseau, pour garantir que les files sont à même de stocker toutes les trames en attente, même dans le scénario le plus défavorable. Le choix de la méthode s'est portée sur le calcul réseau du fait que ses résultats offrent des garanties de type déterministe (par calcul de bornes maximales), avec une méthode de complexité réduite.

1.3 Position du problème

La problématique de cette thèse est donc la maîtrise des temps de communication dans des réseaux temps réel critiques. Il s'agit d'un problème récurrent, rencontré dans de nombreux domaines qui font appel à des systèmes embarqués.

La démarche pire cas issue du calcul réseau, permet l'établissement des garanties de déterminisme. Mais le pessimisme introduit par cette approche pire cas conduit à un sur-dimensionnement du réseau qui est, en moyenne, très peu chargé. Or, les besoins de communication dans les avions continuent de croître, et de nouveaux types de flux, parfois très gourmands en ressources font leur apparition (par exemple multimédia).

Le but de ce travail est de réduire le pessimisme induit dans la démarche pire cas issue du calcul réseau.

2 - Contributions

Nous faisons ici un résumé des contributions de notre travail de thèse.

— **Amélioration de la précision des calculs locaux en priorité statique non préemptive :** Partant du constat selon lequel les résultats existants sur l'agrégation dans un contexte de priorité non préemptive ne gèrent pas bien l'aspect non préemptif de la politique d'ordonnancement, nous proposons une modélisation plus fine permettant de mieux tenir compte de cet aspect. Cette modélisation nous permet alors d'obtenir de meilleurs résultats. Nous proposons en effet ici une généralisation des travaux existants accompagnée d'une meilleure précision des calculs locaux.
Ces travaux ont donné lieu à une publication dans une conférence internationale à comité de lecture, [95] puis dans un workshop international à comité de lecture, [96].

— **Expression du service résiduel en politique DRR :** La politique GPS (General Processor Sharing) est une politique permettant le partage équitable de ressource. À tout instant donné, la distribution de ressource est proportionnelle à des coefficients fixes donnés. Si cette politique s'applique bien avec un modèle de transmission de données fluides, elle nécessite une adaptation pour une transmission par paquets. La politique DRR (Deficit Round Robin) en est une. Nous étudions cette politique dans le cadre du calcul réseau.
Ces travaux ont donné lieu à une publication dans la session "Work in progress" d'une conférence internationale à comité de lecture, [97] puis dans une conférence internationale à comité de lecture, [36].

— **Intégration des politiques à priorité statique non préemptive et la politique DRR :** Un avantage de nos deux premières contributions est que nous proposons des services résiduels (la capacité non utilisée du serveur) qui sont des services stricts (un des type de servie en calcul réseau). Ce qui pourrait sembler un détail technique permet en fait de partager le service rédiduel au moyen d'une autre politique de service. Cette contribution montre comment il est possible d'intégrer politique à priorité statique non préemptive et politique DRR.
Ces résultats ont été utilisés par d'autres auteurs pour une conférence internationale à comité de lecture [35].

3 - Organisation du mémoire

Ce travail est présenté en trois chapitres.

Dans le premier chapitre, nous présentons le contexte technologique. Dans ce chapitre, nous traitons des réseaux embarqués : leurs problématiques, les politiques d'ordonnancement qui y sont utilisées et nous présentons aussi quelques exemples de ces réseaux.

Le deuxième chapitre est consacré à l'état de l'art sur les méthodes d'évaluation des bornes sur le délai pire cas dans les réseaux embarqués. Nous présentons un bref aperçu du modèle-checking, de l'approche par trajectoire, de l'approche basée sur les Event Stream et du calcul réseau.

Le troisième chapitre est consacré à l'exposé de nos contributions. Nous y présentons l'amélioration de la précision des calculs locaux en politique de priorité statique non préemptive et DRR et nous évoquons aussi l'intégration de ces deux politiques.

Chapitre I

Contexte technologique

1 - Les systèmes embarqués temps réel

Les applications temps réel sont celles où en plus de l'exactitude des résultats, la maîtrise du temps est la principale contrainte à respecter et où le respect de cette contrainte est le facteur prépondérant pour estimer la qualité de service [37]. Pour une application temps réel, un résultat juste mais hors délai est un résultat inutilisable, c'est donc une faute temporelle [83]. Ainsi, les deux contraintes à vérifier pour qu'un système temps réel soit fonctionnel sont :

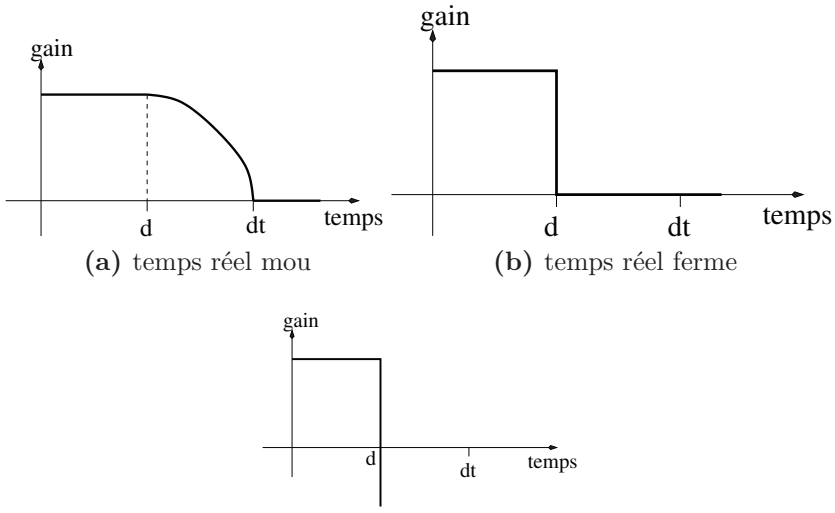
- l'exactitude logique (logical correctness) : sorties adéquates en fonction des entrées, assurant le comportement désiré pour le système suite à des événements et aux données communiquées ;
- l'exactitude temporelle (timeliness) : rencontre des contraintes temporelles. Les sorties sont présentées au bon moment.

On rencontre des contraintes temporelles dans les commandes de procédés, les systèmes embarqués, la surveillance des centrales nucléaires, la conduite d'expériences scientifiques, la robotique, la fourniture d'images et de son pour le multimédia, le suivi opératoire en milieu médical, et même le suivi d'informations boursières.

Les applications déclenchent des événements à occurrence périodique ou aléatoire et imposent au système informatique qui leur est associé de réagir avant un délai fixé ou à une date donnée.

L'échelle de temps peut varier selon les applications : la microseconde dans les radars, la seconde dans une interface homme-machine, une minute dans une chaine de fabrication, une heure pour la réaction chimique.

La rigueur avec laquelle doivent être respectées les échéances temporelles permet de définir une classification des systèmes temps réel, suivant trois catégories [118]. Comme le montre la figure I.1, la distinction est fonction de ce qu'on gagne (resp. de ce qu'on perd) à respecter les



(a) temps réel mou **(b)** temps réel ferme

(c) temps réel dur

Figure I.1 – Définition du temps réel : d désigne la date d'échéance et dt la date de terminaison effective.

échéances (suite à un non respect des échéances.)

— **Système temps réel à contraintes souples** : qualifiés aussi de systèmes temps réel mou, caractérise les systèmes pour lesquels un gain maximal est associé au respect des échéances, comme en temps réel dur, mais pour lequel un dépassement d'échéance entraine un gain qui décroit vers zéro après la date d'échéance (figure I.1a). De tels systèmes tolèrent un nombre spécifié de dépassement d'échéances temporelles. Ils sont typiquement présents dans les applications multimédia, où la perte d'images est acceptable dans une diffusion de vidéo.

— **Système temps réel ferme** : Le gain est maximal pour un respect des échéances, comme pour un système temps réel mou, et nul dans le cas contraire (figure I.1b).

— **Système temps réel à contraintes strictes** : Aussi qualifiés de systèmes temps réel durs, caractérise un système dans lequel le respect de contrainte entraine un gain maximal et le non respect une perte considérable pour le système (figure I.1c). Ainsi, leurs contraintes temporelles doivent impérativement être respectées, sous peine de provoquer des évènements pouvant être catastrophiques (perte du système entraînant des pertes humaines).

2 - Les réseaux embarqués

Un réseau est un ensemble d'équipements informatiques interconnectés. L'interconnexion signifie qu'ils s'échangent des informations. La définition d'un réseau nécessite que certains choix soient faits : notamment le mode d'interconnexion, qui peut être filaire ou sans fil ; la façon dont les équipements sont physiquement reliés, qui s'appelle la topologie ; les caractéristiques du médium de communication ; le sens des échanges des messages qui peut être unidirectionnel, bidirectionnel alterné ou bidirectionnel dans les deux sens ; la politique d'accès au médium qui défini les règles régissant l'accès au médium. Dans la suite, nous présentons quelques uns de ces choix possibles dans les réseaux embarqués.

2.1 Les topologies réseaux

La structure d'un réseau est définie par sa topologie. La définition de la topologie comprend deux aspects : la topologie physique, représentant la disposition effective des fils (média), et la topologie logique, précisant la façon dont les hôtes accèdent au média. Un réseau peut avoir un type de topologie physique et un type de topologie logique complètement différents. Les réseaux Token Ring utilisent une topologie physique en étoile et une topologie logique en anneau. Les réseaux FDDI utilisent une topologie physique et logique en anneau.

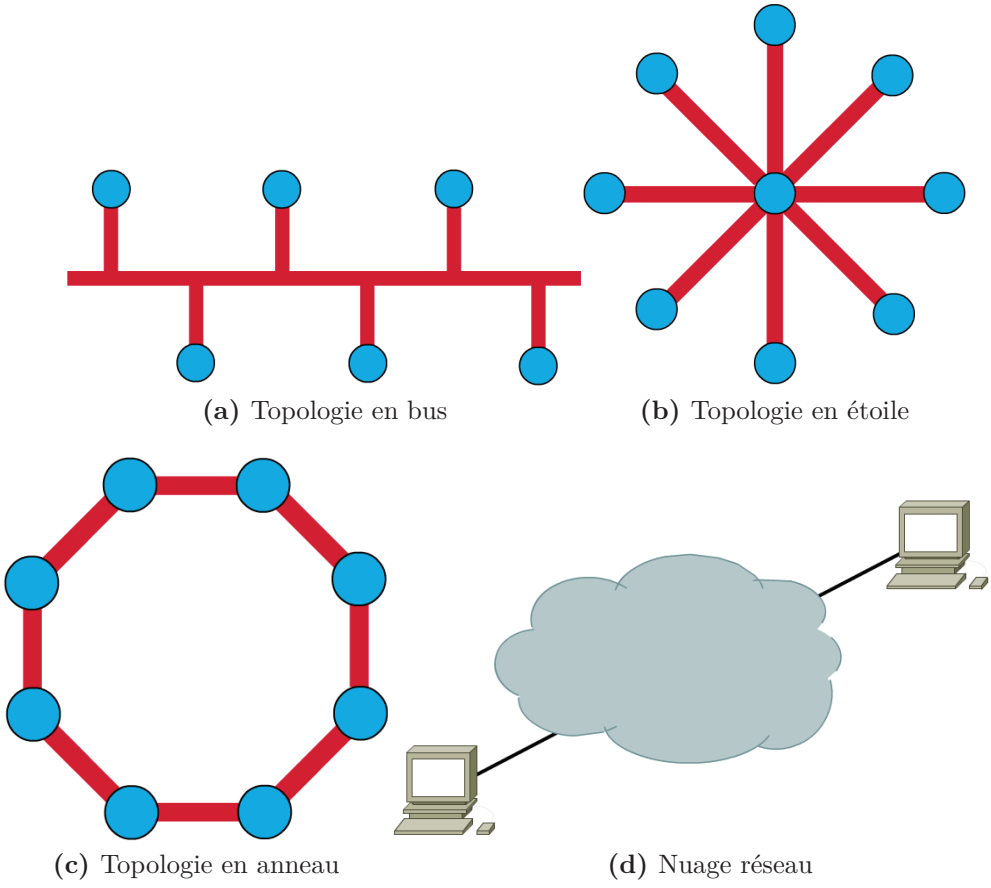
Les topologies physiques couramment utilisées sont la topologie en bus, la topologie en anneau, la topologie en étoile, et le nuage. Elles sont illustrées dans la figure I.2.

2.1.1 Topologie bus

Dans une topologie bus (figure I.2a), tous les nœuds sont directement reliés à un bus de données. Les nœuds ne sont pas directement reliés. Toute communication entre deux nœuds quelconques passe par le bus. Si le bus présente l'avantage d'interconnecter directement les nœuds du réseau, le réseau est perdu en cas de rupture du bus. Lorsqu'un nœud émet sur le bus, tous les autres nœuds reliés au bus peuvent voir l'information. Le bus est avantageux pour la transmission des messages de diffusion. Le bus est une ressource unique partagée. Donc l'utilisation du bus nécessite un mécanisme de gestion de partage, précisément une méthode d'accès.

2.1.2 Topologie en étoile

Dans une topologie en étoile (figure I.2b), on retrouve un nœud central auquel sont directement raccordés tous les autres nœuds du réseau. Aucune autre liaison n'est permise. Comme



(a) Topologie en bus **(b)** Topologie en étoile

(c) Topologie en anneau **(d)** Nuage réseau

Figure I.2 – Les topologies réseaux : Source CISCO

avec le bus, la topologie en étoile permet une interconnexion entre tous les nœuds du réseau via le nœud central. Ce nœud central constitue le talon d'Achille du réseau. Le réseau n'existe plus s'il est en panne. En fonction du type d'équipement central utilisé dans cette topologie, des collisions peuvent survenir. En effet, tandis qu'avec certains équipements le réseau constitue un seul domaine de collision (c'est le cas d'un hub), avec d'autres il est subdivisé en plusieurs domaines de collision (c'est le cas d'un commutateur) ou alors en plusieurs domaine de diffusion (c'est le cas d'un routeur). Dans une topologie en étoile, toutes les informations échangées passent par le nœud central. Ce comportement peut être pratique pour des raisons de sécurité ou de contrôle d'accès. Cette caractéristique peut aussi favoriser l'apparition des problèmes comme la congestion au niveau du nœud central.

2.1.3 Topologie en anneau

Dans une topologie en anneau les nœuds sont organisés de façon circulaire (figure I.2c), chaque nœud étant physiquement relié aux deux autres nœuds adjacents. En général, les informations circulent dans l'anneau dans un seul sens, la communication est unidirectionnelle ou bidirectionnelle alternée. Un nœud qui reçoit une information la transmet à son voisin situé dans le sens de transmission si nécessaire. La structure de l'anneau est perdue en cas de défaillance d'un nœud de l'anneau si la communication est unidirectionnelle.

2.1.4 Nuage réseau

Le nuage réseau ou plus simplement le nuage ou le réseau représente un moyen de communication quelconque entre des équipements informatique. Ce terme peut bien désigner un tout petit réseau tout comme une interconnexion de plusieurs réseaux. Le réseau Internet constitue un bon exemple. Il est généralement représenté par un nuage comme sur la figure I.2d.

2.2 Accès au médium

Dans un réseau, l'accès au médium désigne la méthode qu'utilisent les hôtes pour communiquer sur le média. Les deux types d'accès au médium les plus courants sont le broadcast et le passage de jeton. Il existe plusieurs classifications des méthodes d'accès au médium [94]. La figure I.3 présente une de ces classifications.

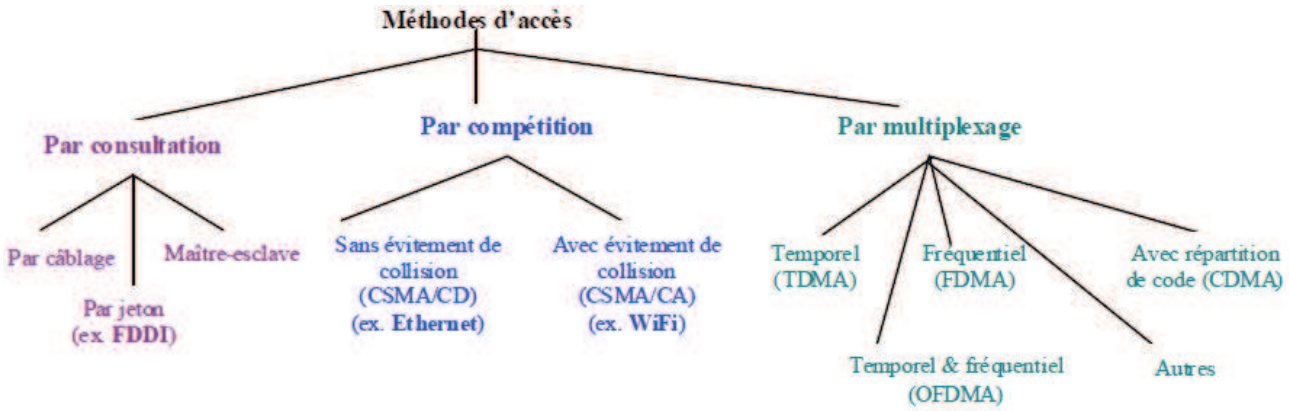


Figure I.3 – Classification des méthodes d'accès : Source [94]

2.2.1 Méthodes d'accès par compétition

Méthodes d'accès sans évitement de collision : CSMA/CD La méthode d'accès CSMA/CD (Carrier Sense Multiple Access with Collision Detection) assure la transmission, la réception et le décodage des paquets de données. Elle permet aussi d'assurer les fonctions de détection d'erreurs. Comme toutes les méthodes d'accès CSMA, la méthode CSMA/CD est une méthode à détection de porteuse. En effet, CS (Carrier Sense) signifie "détection de porteuse". Cela correspond au fait de s'assurer si le bus est occupé ou non avant d'initier une transmission. MA (Multiple Acces), qui veut dire à "accès multiple" signifie qu'il n'y a pas de priorité d'accès au bus entre les nœuds. Ceux-ci émettent lorsqu'ils considèrent le bus libre.

Avec la méthode CSMA/CD, tout équipement désirant transmettre écoute la porteuse jusqu'à ce que le médium de communication soit libre. Si la voie est libre, il commence alors à émettre tout en continuant d'écouter le médium dans le but de détecter d'éventuelles collisions. Si une collision se produit, les équipements responsables de la collision continuent d'émettre pendant quelques temps, pour permettre à tous les autres de détecter la collision. Puis, les équipements désirant transmettre attendent une période de temps aléatoire avant d'essayer d'émettre. Le temps aléatoire permet d'éviter qu'une collision ne se reproduise systématiquement à la suite d'une autre [130, 80, 119]. Lorsque la transmission de données reprend sur le réseau, les nœuds impliquées dans la collision n'ont pas la priorité de transmission des données [98]. Une fois qu'un nœud a vérifié l'adresse de destination transportée par les données, il s'assure que la trame ne contient aucune erreur. En cas de détection d'erreurs, la trame est supprimée.

Méthodes d'accès avec évitement de collision : CSMA/CA CSMA/CA (Carrier Send Multiple Access with Collision Avoidance, signifiant à prévention de collision) est un mécanisme de dialogue basé sur l'évitement de collision [80, 119] :

- Une station qui veut émettre écoute le réseau jusqu'à ce qu'il soit libre pendant un certain temps.
- La station envoie ensuite un message court précisant le destinataire, le volume de données à transmette.
- Le destinataire envoie un Ok pour recevoir les données.
- L'émetteur envoie les données.
- Le destinataire envoie un acquittement des données
- La ligne est de nouveau libre

La différence majeure entre CSMA/CA et CSMA/CD est la possibilité de détection de collisions. Dans la technique CSMA/CD, la collision est détectée à l'émission car les stations ont la possibilité de continuer à écouter leurs transmissions en cours. En revanche, cette collision ne pourra pas être détectée qu'au niveau du récepteur dans le cas du mécanisme CSMA/CA [46, 22].

Méthodes d'accès avec résolution de collision : CSMA/CR Tout comme avec les autres méthodes CSMA, un nœud qui désire émettre avec la méthode d'accès CSMA/RC (Carrier Sense Multiple Access with Collision Resolution) écoute la porteuse jusqu'à ce que le médium soit libre. Le nœud commence à transmettre ses données tout en écoutant la porteuse. La différence avec les autres méthodes de la famille CSMA est qu'en cas de détection de collision, c'est la priorité des nœuds concernés par la collision qui permet de déterminer le nœud qui continuera sa transmission. Les autres nœuds attendront la libération du canal pour de nouveau tenter une transmission. Le réseaux CAN est un exemple d'un tel réseau (cf. section I.2.7). Notons que certains auteurs présentent CAN comme un réseau CSMA/CA [7], ce qui est à notre avis un contre-sens.

2.2.2 Méthodes d'accès par consultation

Méthodes d'accès par jeton Les réseaux de passage de jeton font circuler une petite trame, appelée jeton, autour du réseau. La possession du jeton confère le droit de transmettre des données. Si le nœud qui reçoit un jeton n'a aucune information à transmettre, il passe le jeton à la station d'extrémité suivante. Chaque station peut conserver le jeton pour un délai maximal qui varie selon la technologie mise en place.

Lorsqu'un jeton parvient à un hôte qui a des informations à transmettre, il le saisit et commence sa transmission. Il n'y a aucun jeton sur le réseau pendant que la trame d'information circule sur l'anneau. Les autres stations de l'anneau ne peuvent pas transmettre pendant ce temps. Elles doivent attendre que le jeton soit disponible. Aucune collision ne survient dans les réseaux à jeton [80, 119].

Méthodes d'accès maître-esclave Dans les méthodes maître-esclave, on est en présence de deux types de nœud : les nœuds maîtres et des nœuds esclaves. Le rôle d'un maître est de donner le droit d'expression aux stations esclaves. Ainsi, une station esclave ne peut émettre si elle n'a pas obtenu l'autorisation d'un poste maître. Les règles appliquées par le nœud maître pour autoriser les nœuds esclaves à émettre sont diverses. En particulier, les deux manières suivantes d'autoriser les nœuds esclaves sont utilisées dans les réseaux que l'on trouve dans les installations industrielles automatisées [123, 94].

— Méthode par scrutation régulière (pooling) : Les nœuds esclaves sont classés selon un ordre. Le nœud maître scrute un par un ces nœuds esclaves dans cet ordre en proposant à chacun d'émettre. Une station esclave désirant émettre attend d'avoir reçu l'invitation d'une station maître. Si une station esclave reçoit le droit de transmettre alors qu'elle n'a pas de trame à émettre, alors elle rend immédiatement ce droit de transmission à la station maître. Le mode par scrutation peut ainsi entrainer une perte de temps, du fait de la scrutation, s'il y a plusieurs stations qui n'ont pas de données à transmettre.
— Méthode utilisant une table d'arbitre : dans cette méthode, le nœud maître est appelé arbitre du réseau et possède une table qui lui indique à quel moment exactement il faut scruter chaque nœud esclave. Dans ce cas, la scrutation est interprétée par le nœud esclave comme une demande à émettre (et non une invitation à émettre, comme dans le cas de la méthode à scrutation). Un des réseaux de terrain qui utilise cette méthode est le réseau WorldFIP [9, 125].

En général, le maître parle à un moment donné à l'esclave. L'esclave doit répondre dans un temps donné. Un esclave n'a pas le droit d'initier un dialogue. Le maître peut parler à plusieurs esclaves. Un dialogue entre 2 esclaves passe par le maître. Le calculateur central cadence les dialogues. Ce mécanisme garanti une cohérence des dialogues, pas de collisions et est déterministe. L'inconvénient majeur réside dans la longueur du temps de dialogues.

2.2.3 Méthodes d'accès temporel (TDMA : Time Division Multiple Access)

La méthode TDMA permet la transmission de plusieurs signaux d'information sur le même canal en organisant un partage de manière temporelle. L'allocation du support de communication fonctionne de manière cyclique. Une période correspond à un cycle. Un cycle est divisé en plusieurs sous-cycles dont chacun est affecté à la transmission des informations d'un nœud précis. Chaque nœud connaît sa position exacte dans un cycle et n'a le droit d'émettre que pendant son sous-cycle associé [57, 38].

La durée des sous-cycles alloués aux nœuds peuvent être identiques ou différentes et alors fonction de l'importance et la quantité du flux de données générée par chaque nœud. Dans le premier cas, on parle de TDMA synchrone et dans le second, de TDMA statistique [94]. L'inconvénient majeur du TDMA synchrone est que lorsqu'un nœud n'a pas de données à émettre, le support reste libre. Le TDMA statistique améliore l'utilisation de la bande passante en fixant de manière dynamique la durée d'utilisation du médium par station. La signalisation (gestion des demandes, notification à chaque station de la durée quelle peut utiliser) rend ce TDMA plus complexe à mettre en œuvre.

2.3 Le sens des échanges

On distingue trois modes de transmission selon le sens des échanges des messages.

2.3.1 Liaison simplex : unidirectionnelle

Dans une liaison simplexe, les informations circulent toujours dans le même sens : de l'émetteur vers le destinataire.

2.3.2 Liaison half-duplex : bidirectionnelle alternée

Une liaison semi-duplex ou bidirectionnelle alternée caractérise une liaison dans laquelle les données circulent dans un sens ou dans l'autre sens mais jamais dans les deux simultanément. Chaque extrémité émet à son tour en bénéficiant de toute les capacités de la liaison.

2.3.3 Liaison full-duplex : bidirectionnelle

Une liaison full-duplex se caractérise par le fait que les deux extrémités peuvent émettre en même temps. La présence de deux canaux, dont chacun dédié à un seul sens de communication, permet d'avoir des échangent dans les deux sens simultanément.

2.4 Politiques d'ordonnancement

Les nœuds d'un réseau sont reliés par des liaisons possédant des caractéristiques physiques diverses pouvant entrainer des problèmes. Par exemple, un engorgement peut survenir à un nœud qui relie des liaison de capacité différentes. Il peut en être de même pour un nœud ayant plusieurs entrées ou ayant une capacité inférieure à celle des liaisons qu'il relie. Certains nœuds du réseau sont capables de stocker temporairement, dans leurs files d'attente, les données qu'ils ne peuvent transmettre immédiatement, pour les transmette plus tard quand les conditions le permettront. Une fois que les voies de transmissions se libèrent, quels sont les messages de la file d'attente qui seront les premiers à être transmis ? C'est justement le rôle de la politique d'ordonnancement de choisir un message parmi plusieurs prêts à être transmis pour l'émettre.

Cette question de l'ordonnancement a été très étudiée dans le cadre de l'ordonnancement de tâches : lorsque plusieurs tâches doivent être exécutées sur un processeur, quelle tâche sélectionner à quel moment pour permettre à chacun de respecter ses échéances ? Il y des similitudes et des différences entre ordonnancement de tâches et de messages, mais les travaux les plus importants ont été menés autour de l'ordonnancement de tâches. La taxonomie que nous allons présenter ici est principalement issue des travaux sur l'ordonnancement des tâches, mais son vocabulaire est aussi pertinent pour l'ordonnancement de messages.

2.4.1 Typologie des algorithmes d'ordonnancement

En ligne vs hors ligne On distingue les algorithmes hors ligne et les algorithmes en ligne. Dans le cas d'un algorithme d'ordonnancement hors ligne, tous les paramètres temporels des tâches sont donnés d'avance. Une planification complète de l'ensemble de toutes les tâches est faite et l'analyse s'appuie sur cette planification statique.

À l'inverse, les algorithmes d'ordonnancement en ligne sont plus dynamiques et s'adaptent aux changements des paramètres temporels. De tels algorithmes sont capables de sélectionner, à un instant donné, la prochaine tâche à exécuter sur la base des paramètres temporelles disponibles. Cela permet de prendre en compte de nouvelles informations temporelles en cours d'exécution et de les intégrer lors du prochain instant de décision visant à sélectionner la tâche à exécuter. Ces algorithmes permettent l'arrivée imprévisible des tâches et autorisent la création progressive de la séquence d'ordonnancement [47].

Non préemptif vs Préemptif Cette section compare les systèmes de tâches non-préemptives avec les systèmes de tâches préemptives. Le terme préemptif signifie qu'une tâche en cours d'exécution peut être interrompue à tout moment par une autre tâche plus prioritaire. Même si la

plupart des algorithmes d'ordonnancement sont préemptifs, il existe des situations où l'ordonnancement non-préemptif est préférable. Par exemple le surcoût engendré par la préemption n'est pas toujours négligeable par rapport aux durées d'exécutions des tâches et aux temps de communication entre processeurs. De plus, pour la majorité des systèmes embarqués, la préemption est trop chère en temps et espace à cause du coût lié au changement de contexte.

Les ordonnancements de messages ne sont pas, en général, préemptif : lorsque qu'un message a été sélectionné pour être émis, on ne peut pas suspendre son émission : on ne peut que l'annuler, alors que la préemption suppose une possibilité de reprise.

2.4.2 Ordonnancement des messages

Dans les applications temps réels et réparties, on retrouve des contraintes temporelles sur l'exécution des tâches qui s'échangent des messages. Les tâches pouvant se situer sur des processeurs différents, la transmission de ces messages se fait sur un médium de communication et doit respecter certaines contraintes. Une tâche qui attend un message est conditionnée par la bonne transmission de celle-ci dans les délais. C'est comme l'attente d'acquisition d'une ressource. Les contraintes temporelles de la tâche pourraient ne plus être respectées si le message n'arrive pas à temps. Dans une application temps réel, si un message arrive au-delà du délai fixé, il est alors considéré comme perdu. Un message doit être correct du point de vue contenu, mais aussi du point de vue temporel. Il est donc nécessaire que le support de communication garantisse un respect de délai de communication des messages temps réels, permettant ainsi de garantir aux tâches une bonne réception des messages qui leurs sont destinées dans les délais.

Communication temps réel Une communication temps réel est définie par des contraintes de temps explicites, c'est-à-dire qu'elle doit commencer ou se terminer dans un intervalle de temps déterminé. En général, les contraintes temporelles associées à un message temps réel sont de deux types : un délai de transfert borné et une gigue bornée. Les réseaux qui disposent des mécanismes adéquats pour garantir le respect de ces contraintes sont appelés réseaux temps réel [48]. De tels réseaux doivent posséder les propriétés temps réel suivantes :

- prédiction des temps de réponse des messages à contraintes strictes ;
- borne maximale connue pour le délai de transfert ;
- haut degré d'ordonnançabilité, c'est-à-dire garantie a priori du respect des contraintes temporelles d'un nombre élevé de messages ;
- possibilité d'échanges périodiques ou apériodiques de messages ;
- possibilité d'échanges de messages avec des contraintes de temps strictes ou non ;

— bonnes performances et surcoût peu important ;
— tolérance aux fautes ;
— fiabilité des transferts ;
— fonctionnement dans des conditions environnementales pouvant être sévères ;
— niveau élevé de validation et de testabilité.

Caractéristique des messages temps réels Les paramètres les plus utilisés pour spécifier les caractéristiques des sources de messages échangés dans une application temps réel et répartie sont les suivants [48].

— La criticité. Comme pour les tâches, les messages temps réel peuvent être à contraintes strictes (tout manquement au respect des contraintes temporelles peut conduire à une détérioration du service, voire à des catastrophes), à contraintes relatives (un manquement occasionnel au respect des contraintes temporelles ne conduit pas à la dégradation du service), à contraintes mixtes.
— Le type. Un message peut être périodique (dit aussi synchrone) ou apériodique (dit aussi asynchrone). La période d'un message périodique désigne l'intervalle de temps séparant deux demandes d'émission successives du message par sa source. Un message apériodique peut demander à être transmis à n'importe quel moment. Cependant, pour pouvoir garantir, dans certains cas, ou optimiser, dans d'autres, le respect des contraintes de temps associées à un message apériodique, un certain nombre d'informations concernant les instants de demande de transmission du message doivent être connues, à savoir : l'intervalle de temps minimal séparant deux demandes successives de transmission du message, les durées des avalanches du message, l'intervalle de temps minimal séparant deux avalanches successives, etc. Toutes ces informations ne sont pas toujours nécessaires. Ainsi, pour le respect des contraintes de temps des messages apériodiques avec contraintes strictes, l'intervalle minimal entre deux demandes successives de transmission du message suffit.
— La longueur. C'est la longueur du message en bits.
— Les relations entre messages. Les messages générés par une même source ont, de ce fait, une relation de précédence. Les messages générés par des sources différentes appartenant à une même station ou à des stations différentes peuvent être indépendants ou bien avoir des relations de dépendance. Par exemple, les applications multimédias doivent synchroniser le son et l'image et spécifient que les messages issus de ces deux sources soient transmis dans une même fenêtre temporelle.
— La destination. Un message émis par une source peut être destiné à une seule station, à

un groupe de stations ou à toutes les stations.

2.5 Les tâches temps réel

La caractérisation d'une tâche peut varier d'un modèle d'ordonnancement à l'autre et suivant la nature de celle-ci. Comme paramètres d'une tâche, on peut citer

- $R(t_i)$ **(Ready time ou Release time)** : c'est la date à laquelle la tâche t_i peut commencer son exécution, elle est appelée aussi date de demande d'activation,
- $S(t_i)$ **(Start time),** $E(t_i)$ **(End time)** : sont respectivement la date à laquelle la tâche t_i est exécutée sur le processeur appelée aussi la date de début d'exécution et la date à laquelle la tâche t_i finit son exécution appelée aussi la date de fin d'exécution,
- $RT(t_i)$ **(Response time)** : ceci représente $E(t_i) - S(t_i)$,
- $C(t_i)$ **(Computing time)** : c'est la durée d'exécution d'une tâche t_i. Ce paramètre représente une borne supérieure du temps d'exécution c'est à dire que la tâche peut se terminer plus tôt. Pour être valable, la valeur de ce paramètre ne doit pas être trop surestimée et doit être sûre (jamais dépassée) ;
- $D(t_i)$ **(Deadline)** : c'est l'échéance ou la date au plus tard. Elle représente l'instant auquel l'exécution d'une tâche doit être terminée et dont le dépassement provoque une transgression de la contrainte temporelle. Deux types de deadlines existent :
  - **relative** $D(t_i) + R(t_i)$: l'échéance est relative au release time de la tâche,
  - **absolue** $D(t_i)$: l'échéance est définie avant l'exécution,
- $l(t_i)$ **(Laxity)** : c'est la laxité d'une tâche t_i, qui représente le temps restant avant l'occurrence de sa date de début d'exécution ou de reprise au plus tard. Si ce paramètre vaut zéro à un instant donné, la tâche correspondante doit être impérativement exécutée à cet instant et sans interruption sinon son échéance sera inévitablement dépassée.

Principe et stratégies de l'ordonnancement des messages Il existe plusieurs stratégies d'ordonnancement de messages. Les trois suivantes sont les plus rencontrées.

- Stratégie déterministe (ou avec garantie) : tout message acceptée pour transmission est transmis en respectant toutes ses contraintes temporelles (sauf entendu en cas de défaillance du système).
- Stratégie probabilistes : les transmissions de messages sont garanties avec une probabilité connue à l'avance. Les messages peuvent donc rater leurs échéances.
- Stratégie avec meilleur effort : aucune garantie n'est assurée pour la livraison des messages. Le système de communication fait de son mieux pour garantir les contraintes de

temps des messages.

La stratégie déterministe est généralement réservée aux messages avec des contraintes de temps strictes dont le non respect peut avoir des conséquences graves, la deuxième stratégie est utilisée pour les messages avec des contraintes temporelles strictes dont le non respect n'a pas de conséquences graves. La stratégie avec meilleur effort convient pour traiter les messages avec contraintes de temps relatives ou sans contraintes de temps.

Dans un système temps réel et réparti, les trois stratégies peuvent cohabiter, pour pouvoir répondre à différents besoins de communication, selon les contraintes et la nature des tâches communicantes.

2.5.1 Les algorithmes d'ordonnancement

Avec l'émergence des systèmes temps réel et répartis, de nombreux besoins d'ordonnancement sont apparus : il faut à la fois garantir le respect des contraintes temporelles des tâches et celles des messages. Comme l'ordonnancement des messages tient compte des contraintes analogues à celles des tâches, l'ordonnancement des messages temps réel utilise des techniques analogues à celles de l'ordonnancement des tâches. Alors que les tâches peuvent, en général, accepter la préemption sans remettre en cause la cohérence des résultats, qu'elles élaborent, la transmission d'un message n'admet pas de préemption. Si la transmission d'un message commence, il faut transmettre tous les bits du message jusqu'au dernier, sinon la transmission échoue. Ainsi, certaine précautions doivent être prises pour appliquer les algorithmes d'ordonnancement de tâches aux messages :

- ne considérer que les algorithmes non préemptifs,
- utiliser les algorithmes préemptifs à condition d'avoir des messages dont la durée de transmission est inférieure ou égale à l'unité de temps de base d'allocation du médium.
- utiliser les algorithmes préemptifs à condition d'avoir des messages longs en petit paquets et de ré-assembler les paquets à la réception ; la fonction de segmentation et de ré-assemblage devant alors être effectuée par une couche de niveau supérieur à la sous-couche MAC.

Sur un réseau à commutation de paquets avec ordonnancement non préemptif, un paquet peut être gêné par un paquet moins prioritaire. En effet, si un paquet m est généré au moment où un paquet m', de priorité strictement inférieure, est en cours d'émission, alors m doit attendre la fin de traitement du paquet m'.

Il est important de noter que la non préemption peut entraîner un retard supplémentaire. En effet, si des paquets plus prioritaires que m sont générés après m mais avant son début

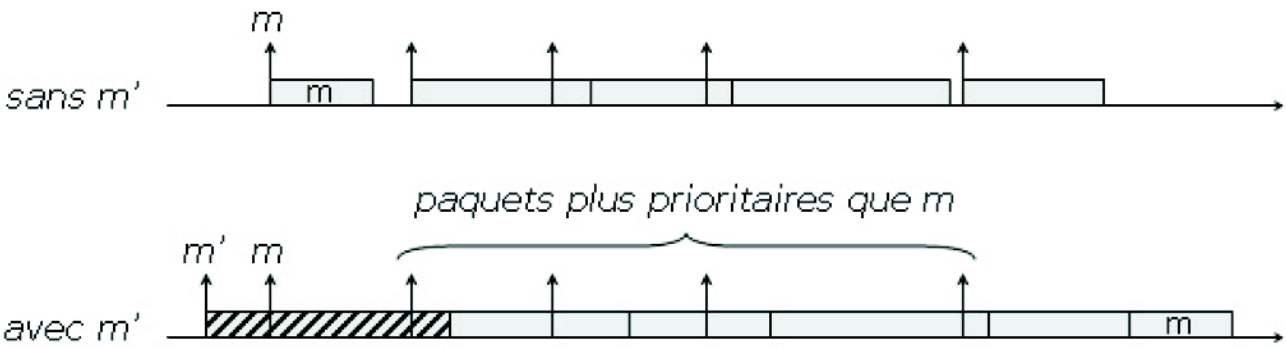


Figure I.4 – Effet non préemptif : Source [99]

d'émission, alors ils seront traités avant m lorsque le processeur se libérera, comme le montre la figure I.4.

Soit m un paquet appartenant à flux quelconque τ_i et généré à l'instant t. Si un paquet m' est généré avant t avec une priorité plus faible, alors m peut subir un délai dû à la non préemption. [99] a décomposé ce délai en deux, à savoir :

— le délai subi par m dû à l'effet non préemptif direct, correspondant au temps d'attente de fin d'exécution du paquet m' ;

— le délai subi par m dû à l'effet non préemptif indirect, correspondant au retard introduit par les paquets plus prioritaires que m qui n'auraient pas été traités avant lui si m' n'avait pas existé.

Certains protocoles de communication offrent des mécanismes puissants pour prendre en compte des contraintes de temps. C'est le cas notamment des protocoles des réseaux FDDI et des bus à jeton qui permettent de traiter facilement des messages périodiques. D'autres protocoles, plus généraux, comme CSMA/CD, nécessitent des mécanismes supplémentaires pour inclure des contraintes de temps. Par conséquent, l'ordonnancement et donc l'adaptation des tâches aux messages, est étroitement lié au type de contraintes temporelles (notamment s'il s'agit des messages périodiques ou apériodiques) et du type de protocoles (notamment si celui-ci garantit un délai d'attente borné ou non)

2.5.2 Ordonnancement à priorités fixes

Un algorithme d'ordonnancement est dit à priorité fixe si chaque tâche possède une priorité qui ne varie pas au cours de la vie de celle-ci. Les algorithmes à priorité fixe les plus connus sont «Rate Monotonic » [92] et «Deadline Monotonic »[90]

Priorité statique Les priorités des tâches sont fixées à l'avance et ne changent pas.

Rate Monotonic Avec cet algorithme, la priorité d'une tâche est fonction de sa période, de telle sorte que la tâche de plus petite période est la tâche la plus prioritaire.

Deadline Monotonic Avec cet algorithme, la priorité d'une tâche dépend de son délai critique, de telle sorte que la priorité d'une tâche est inversement proportionnelle à son délai critique. Ainsi, la tâche de plus haute priorité possède le délai critique le plus petit, et la tâche de plus basse priorité le plus grand.

2.5.3 Ordonnancement à priorité dynamiques

Ordonnancement FIFO À un instant donné, la tâche à activer est la tâche la plus vieille dans la file d'attente. Cet algorithme permet de prendre en compte naturellement toutes les tâches à activer les unes après les autres.

Ordonnancement GPS GPS (Generalized Processor Sharing) a été introduit par [106] pour résoudre le problème de partage équitable de ressources à des usagers d'un réseau à intégration de service. GPS est un ordonnancement idéal non implémentable où les flux sont vus comme des fluides. L'objet de ce modèle est de garantir une bande passante minimum à chaque classe de trafic et ceci à tout instant. Autrement dit, il garantit un débit minimum à chaque classe. Pour ce faire, les flux sont servis en parallèle avec une portion du débit de l'interface proportionnelle à des poids, prédéfinis et fixés pour chaque flux.

Considérons un serveur partagé par n flux $R_1, \ldots, R_n$. Un ordonnancement GPS se caractérise par la donnée des nombres réels positifs $\phi_1, \phi_2 \ldots, \phi_n$. Cet ordonnancement est souvent présenté comme une politique qui partage sa capacité de telle sorte que, chaque flux R_i reçoive une fraction $\rho_i = \phi_i / \sum_j \phi_j$.

La politique GPS, est présentée dans [106] de la façon suivante. Supposons qu'un serveur soit partagé par n flux $R_1, \ldots, R_n$. La politique GPS est présentée comme une politique qui partage équitablement la capacité du serveur entre les différents flux.

Définition 1 (GPS)**.** *Soit S, un serveur partagé par n flux $R_1, \ldots, R_n$. Ce serveur applique la politique GPS de paramètres (non nuls) $\phi_1, \ldots, \phi_n$ si et seulement si, pour tout intervalle de temps $[t, s[$ pendant lequel un flux R_i possède des données à transmettre, la relation suivante est vérifiée pour tout j*

$$Output_i(t, s) \geq \frac{\phi_i}{\phi_j} Output_j(t, s) \tag{I.1}$$

Où $Output_i(t,s)$ est la sortie du flux i entre les instants t et s. Cette relation est souvent écrite $\frac{Output_i(t,s)}{Output_j(t,s)} \geq \frac{\phi_i}{\phi_j}$ mais ne possède pas la même bonne propriété mathématique lorsque $Output_j(t,s) = 0$.

Plusieurs algorithmes ont été proposés pour simuler GPS, dont les plus connus sont PGPS/WFQ et DRR. Ces implantations permettent de gérer les flux transmis sous forme de paquets, contrairement à la politique GPS qui ne supporte que des flux fluides, ce qui n'est pas conforme à la réalité. Nous présentons ici les politiques PGPS et DRR.

La politique PGPS Comme il est mentionné dans [106], le problème majeur de la politique GPS est qu'elle est idéaliste dans la mesure où elle ne supporte pas la transmission des paquets. Elle suppose que le serveur peut servir plusieurs flux simultanément et que le trafic est infiniment divisible. Dans [106] la politique PGPS (aussi appelée WFQ) est définie comme une excellente approximation de la politique GPS qui supporte la transmission de messages par paquets. L'algorithme de la politique PGPS émet les paquets en respectant au mieux la politique GPS. PGPS aimerait en effet que l'ordre de sortie des paquet soit le même que si GPS avait été utilisé. Pour résoudre ce problème, PGPS associe à chaque paquet, à son arrivée, la date de départ de départ qu'il aurait eut avec GPS. Puis les paquets sont servis dans l'ordre de ces dates GPS. Mais il peut par exemple arriver qu'au moment de prendre la décision d'ordonnancement, le prochain paquet à sortir du GPS ne soit pas encore arrivé.

La politique DRR L'algorithme 1 de la page 96 [1] explique le fonctionnement de la politique DRR. Nous supposons qu'il y a n flux et que lorsqu'un paquet du $i^{ième}$ flux arrive dans le système, il est placé dans la $i^{ième}$ file d'attente. Le comportement de l'ordonnanceur est alors le suivant : une boucle globale et infinie inspecte toutes les files d'attentes, les une après les autres. Si la $i^{ième}$ file n'est pas vide, elle est sélectionnée, le compteur $\texttt{DC}[i]$ est incrémenté d'un quantum Q_i. Une boucle interne permet de s'assurer que tant-que la quantité $\texttt{DC}[i]$ est égale ou supérieure à la taille du paquet en tête de la file d'attente, ce paquet est transmis et la valeur du crédit d'émission est décrémentée d'une valeur correspondant à la taille du paquet transmis. La boucle interne s'arrête lorsque le flux considéré n'a plus assez de crédit pour transmettre le paquet se trouvant en tête de file ou alors quand celle-ci est vide. Dans ce dernier cas, $\texttt{DC}[i]$ est positionné à zéro.

1. Cet algorithme à été placé à cet endroit pour être plus proche de la preuve du théorème 11.

2.5.4 Ordonnancement hiérarchique

L'ordonnancement hiérarchique intervient en situation d'agrégation de flux. Dans un premier temps, les flux de même priorité sont agrégés pour former des classes de priorité. Une classe de priorité est vue comme étant un seul flux du point de vu priorité. À l'intérieure d'une classe priorité, on utilise une autre politique de priorité pour ordonnancer les flux. Considérons par exemple le cas de cinq flux qui se partagent un service, les deux premiers ayant une même priorité et cependant plus forte que les trois derniers qui possèdent aussi la même priorité. On peut dans ce scénario agréger les deux premiers flux en un seul flux prioritaire et les trois derniers flux en un seul flux moins prioritaire et avoir ainsi deux classes de priorité. À l'intérieur de chaque classe de priorité, une autre politique de service peut être utilisée pour l'attribution du service à chaque flux individuel. Ainsi, l'on peut par exemple imaginer que les deux flux prioritaires se partage leur service résiduel en utilisant la politique DRR.

2.6 Principales technologies réseau temps réel embarqués

Les systèmes embarqués ont des caractéristiques très spécifiques. Celles-ci nécessitent des réseaux spécifiques : les réseaux embarqués. On distingue les réseaux embarqués basés sur la technologie EVENT-TRIGGERED (ET) et les réseaux embarqués basés sur la technologie TIME-TRIGGERED (TT). En ET, il n'y a pas de déterminisme. Ce qui pose un problème pour le respect des contraintes temporelles. Avec TT, les nœuds doivent être synchronisés. Les transactions sont déclenchées à des instants prédéfinis et les destinataires doivent connaître l'état global du système. La technologie Time-Triggered est bien adaptée au trafic périodique et permet une bonne gestion de la bande passante en cas de grosse charge du réseau.

Plusieurs protocoles utilisent la technologie « Time Triggered ». On peut citer TTP [84], TTP/C et TTP/A [78, 54]. On trouve aussi TTCAN (une implémentation d'un ordonnancement Time Triggered sur CAN) [8, 120, 55, 103] et ByteFlight [21]. FlexRay [115, 110] concilie communication Time-Triggered et Event-Triggered.

Les bus de terrain Les réseaux de terrain sont des systèmes de communication qui connectent les capteurs, les actionneurs, les systèmes de commande comme des automates, des régulateurs, des micro-contrôleurs, etc. Comme tous les réseaux, ils sont constitués d'une pile de protocoles selon une architecture issue du modèle OSI de l'ISO. Les réseaux de terrain ou « fieldbus » ou (encore « fieldnetwork ») sont aussi appelés « sensor bus » ou « device bus ». Ces réseaux sont nés au début des années 1980 [124, 44].

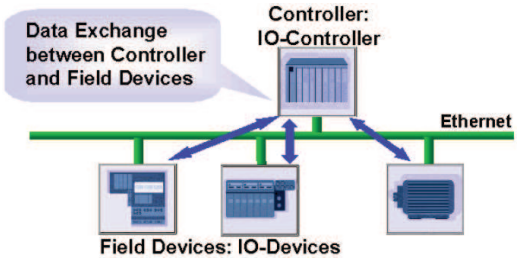


Figure I.5 – Achitecture PROFIBUS : Source [111]

Profibus Profibus est un bus de terrain destiné à couvrir les différents besoins en communication industrielle dans les automatismes, depuis le niveau capteur-actionneur jusqu'au niveau de contrôle-commande et de supervision de process [53]. Le PROFIBUS est réalisable en topologie linéaire, arborescente, étoile ou en anneau avec une vitesse de transmission allant de 9,6 kbits/s à 12 Mbits/s et pouvant même atteindre les 12 Mbits/s avec la fibre optique. La communication sur ce bus est déterministe suivant le principe maître-esclave.Une trame PROFIBUS comporte entre 1 et 144 octets de données utile. Le système peut-être "multi-maîtres", avec un maître principal et des maîtres secondaires. Le système a alors une durée de cycle. Le maître principal utilise le début du cycle pour discuter avec les esclaves, puis, s'il reste du temps, passe un jeton de parole à un maître secondaire, qui peut utiliser le reste du cycle pour discuter avec les esclaves, et, s'il reste encore du temps, passer le jeton à un autre maître secondaire, jusqu'à la fin du cycle [18].

Modbus Modbus [123] est un protocole de communication non-propriétaire, créé en 1979 par Modicon, utilisé pour des réseaux d'automates programmables, relevant du niveau 7 (applicatif) du Modèle OSI. Il fonctionne sur le mode maître/esclave. Il est constitué de trames contenant l'adresse de l'automate concerné, la fonction à traiter (écriture, lecture), la donnée et le code de vérification d'erreur appelé contrôle de redondance cyclique sur 16 bits ou CRC16 [52].

Le protocole Modbus peut être implémenté de plusieurs façons :

— En utilisant une liaison série asynchrone de type RS-232, RS-422 ou RS-485 ou TTY, avec des débits pouvant atteindre 5 Mbps sur une topologie en bus ou arborescente et selon la méthode d'accès maître/esclave ;

— Modbus peut être mis en œuvre via TCP/IP sur Ethernet ; on parle alors de Modbus TCP/IP avec un débit qui varie de 10 à 100 Mbps sur une topologie qui peut être de l'étoile ou de l'anneau selon le mode client/serveur ;

— Modbus Plus est un réseau multi-maîtres avec un accès au réseau géré par jeton et pouvant atteindre un débit de 1 Mbps.

Le bus CAN CAN [65, 66] est un bus à diffusion obéissant à la technique CSMA/CR. L'accès au médium se fait en fonction de la priorité des trames. Conforme à la classe C, dans sa version haut débit (250 Kbits à 1Mbits sur une longueur de 250 à 30 mètres), il peut alors être intégré, d'une part, dans les systèmes des domaines nécessitant des performances temps réel (contrôle moteur et châssis) et, sa conformité à la classe B, dans sa version bas débit (10 à 125 Kbits sur une longueur de 5000 à 500 mètres) le rend apte à supporter les échanges du domaine habitable [104]. Le paragraphe I.2.7 fourni plus de détail sur CAN.

FlexRay Le protocole FlexRay [FLE 04] a été présenté par un consortium constitué de BMW, Bosch, Daimler Chrysler, General Motors, Motorola, Philips et Volkswagen. Un réseau FlexRay peut être bâti sur une topologie en bus, en étoile simple ou étoile multiple ou alors être issue d'une combinaison de ces solutions. La stratégie d'accès au médium de communication repose sur un cycle de communication qui se répète périodiquement et qui enchaîne une fenêtre statique à accès strictement TDMA et une fenêtre dynamique à accès TDMA flexible.

La fenêtre statique est divisée en un nombre fixe de slots de taille constante (le nombre de slots est limité à 2047). Chaque nœud se voit allouer statiquement un ou plusieurs slots pendant lesquels il peut transmettre ses messages. La fenêtre dynamique permet de réaliser un comportement guidé par les événements. La base de temps est le minislot. La trame FlexRay est formée comportement 254 octets de données utiles.

2.6.1 TTP (Time Triggered Protocol)

Le bus TTP (Time-Triggered Protocol) est un bus de terrain développé par TTTech, pour l'automobile et l'aéronautique.

Le protocole TTP/C est en fait le coeur d'un concept plus large, appelé TTA (Time-Triggered Architecture) [77, 76] qui sont des architectures de systèmes dont le comportement (tâches et émission de trames) est strictement guidé par le temps. TTP/C fournit, entre autres, une base de temps stable à toutes les activités s'exécutant sur chacun des nœuds connectés.

La politique d'accès au médium de communication proposée par TTP est de type TDMA (Time Division Multiple Access).

La charge utile de la trame est au maximum de 240 octets.

2.6.2 Réseau Ethernet

Ethernet est un protocole de réseau local à commutation de paquets. Il propose des spécifications pour la couche physique, la sous-couche MAC et décrit le format de trame. Les topologies

qu'on rencontre avec Ethernet sont le bus et l'étoile. En fonction du support de transmission utilisé, un réseau Ethernet peut émettre à un débit allant de 10 Mbps quelques dizaines de Gbps. Il a été standardisé sous le nom IEEE 802.3.

Dans un réseau Ethernet, les informations sont diffusées à tous les équipements connectés et l'accès au médium est assuré par le protocole CSMA/CD.

2.6.3 Réseau Ethernet commuté

Dans un réseau Ethernet commuté, la configuration qui est adoptée ressemble à celle d'une topologie en étoile étendue. Ici, il n'y a pas vraiment d'équipement central puisque tous les commutateurs de la topologie sont susceptibles d'être reliés directement avec un équipement terminal. Cette topologie permet de réduire par segmentation les domaines de collision au seul lien point à point entre un équipement et son commutateur, ou bien entre deux commutateurs. Dans un réseau Ethernet commuté, l'équipement le plus important est bien sûr le commutateur. La norme 802.1D définit ces équipements [59].

Plusieurs aspects font de l'Ethernet commuté un bon candidat pour supporter les besoins de communication temps réel dans les couches basses du processus de Contrôle-Commande. Ethernet est le standard réseau le plus répandu se prêtant à des transmissions haut débit (de 10Mbps à plusieurs dizaines de Gbps). L'absence de collision le prédestine aux communications nécessitant un temps d'accès déterministe. La notion de priorité est supportée par Ethernet, ce qui le rend apte à la différenciation de classe de service et par conséquent de fournir une meilleur qualité de service aux applications temps réels [79].

2.6.4 PROFInet

PROFInet est un standard de communication créé par PROFIBUS International pour mettre en œuvre des solutions d'automatisation intégrées et cohérentes, sur Ethernet industriel. ll s'agit avant tout d'intégrer les bus de terrain en place, tels PROFIBUS, sans toucher à l'existant [67]. Il peut être mis en œuvre avec une topologie en étoile, en anneau ou hiérarchique [68, 56]. PROFInet sait fédérer, sur Ethernet avec la méthode d'accès CSMA/CD, [56] aussi bien des appareils de terrain simples et des applications à temps critique, que des automatismes répartis à base de composants.

PROFInet mixe communications temps réel et non temps réel sur une base cyclique : des slots temporels sont dédiés aux communications temps-réel, avec un protocole spécifique, SRT (Soft Real Time), directement sur Ethernet, puis une partie dédiée au non-temps réel, sur TCP/IP [108].

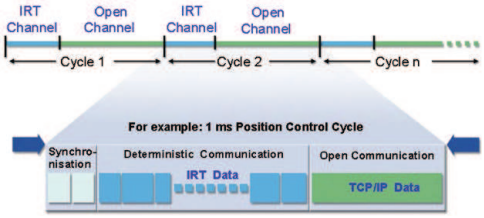


Figure I.6 – Découpage communications temps réel/non temps réel de PROFInet : Source [108]

2.6.5 AFDX

Le réseau AFDX (Avionics Full DupleX Switched Ethernet) qui se défini comme étant la version avionique de l'Ethernet commuté Full-Duplex, est standardisé par la norme ARINC 664 [1, 2, 3]. Il s'agit d'un réseau avec routage statique, qui offre l'avantage de fournir un accès déterminisme au support de communication et donc de ne pas avoir de pertes de trames par collision. L'un des aspects singuliers de l'AFDX est la maîtrise du trafic à l'entrée du réseau. Ce contrôle de flux permet une gestion garantie des délais et l'absence de perte de trame par débordement des mémoires. L'AFDX sera présenté en détail au paragraphe I.2.8

2.6.6 TTEthernet

La technologie TTEthernet supporte 3 types de flux : TT, RC et BE. Le flux TT est le plus prioritaire, a un accès au médium TDMA. Le flux RC, de priorité intermédiaire, a un accès suivant le modèle AFDX présenté au paragraphe I.2.6.7. Le flux BE, de plus faible priorité est un flux ethernet classique. TTEthernet s'appuie sur l'Ethernet commuté et supporte toute topologie formé de commutateurs [122, 75]. Les transmissions vont de 100Mbps à 1Gbps. Le format d'un message TT Ethernet est basée sur le format du message normalisé Ethernet (norme IEEE 802.3).

2.6.7 Le standard MIL STD 1553B

Le standard MIL STD 1553B [64] est un bus utilisé dans plusieurs applications militaires comme l'avionique, les sous marins, les missiles et les satellites [102]. La figure I.7 montre un exemple de l'architecture du bus avionique MIL STD 1553B. L'accès au bus est géré par le mécanisme commande/réponse : le contrôleur du bus (BC) envoie des commandes aux différents terminaux (RT) pour leur autoriser l'accès au bus et l'envoi de leurs données. Un terminal spécial appelé Monitor (M) reçoit et stocke tout message circulant sur le bus et son rôle principal est la surveillance et la vérification de l'état du bus.

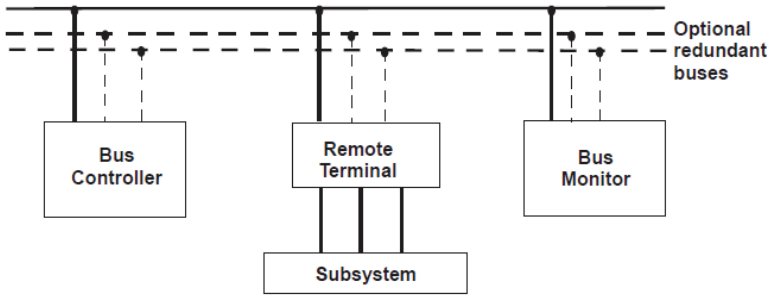


Figure I.7 – Architecture d'un bus MIL STD 1553B : Source [102]

Le standard MIL STD 1553B supporte les messages périodiques et apériodiques. Il définit aussi deux formats de messages : broadcast et non broadcast, et chaque format comporte des messages de données et des messages de gestion de communication. Tous les messages sont transmis sous la forme d'un ensemble de mots de 20bits (3 bits de synchronisation, 16 bits de données et 1 bit de parité).

2.7 CAN : Control Area Network

Cette partie a pour but de présenter les caractéristiques essentielles du bus/réseau de terrain. Le bus CAN (Controller Area Network) est un moyen de communication série qui supporte des systèmes embarqués temps réel avec un haut niveau de fiabilité.

2.7.1 Description

Le protocole « Controller Area Network » (CAN) [58] a été développé pour l'industrie automobile pour assurer des communications entre plusieurs équipements, rapides, sûres et prédictibles. Il est maintenant utilisé dans un grand nombre d'autres domaines. La topologie logique associée à CAN est un bus (à diffusion). Les messages sont encapsulés dans une trame. Une trame peut transporter des données dont la taille peut atteindre 8 octets. La trame reçoit un identifiant unique codé sur 11 bits (pour la version 2.0A) qui permet de filtrer les trames à la réception et de lui allouer une priorité. L'acceptation d'une trame par un nœud est autorisée au regard de l'identifiant de cette trame : si l'application qui s'exécute sur un nœud a besoin des données transportées par une trame alors son identifiant aura été déclaré au préalable à son interface de communication. Ainsi, seules les trames reçues qui ont les identifiants désirés, sont acceptées. La trame ne transporte donc pas explicitement l'adresse de ses destinataires [63]. L'identifiant d'une trame est aussi utilisé pour contrôler les accès au bus CAN et mettre ainsi en œuvre un mécanisme d'arbitrage selon des priorités entre les trames. Un détail sur le

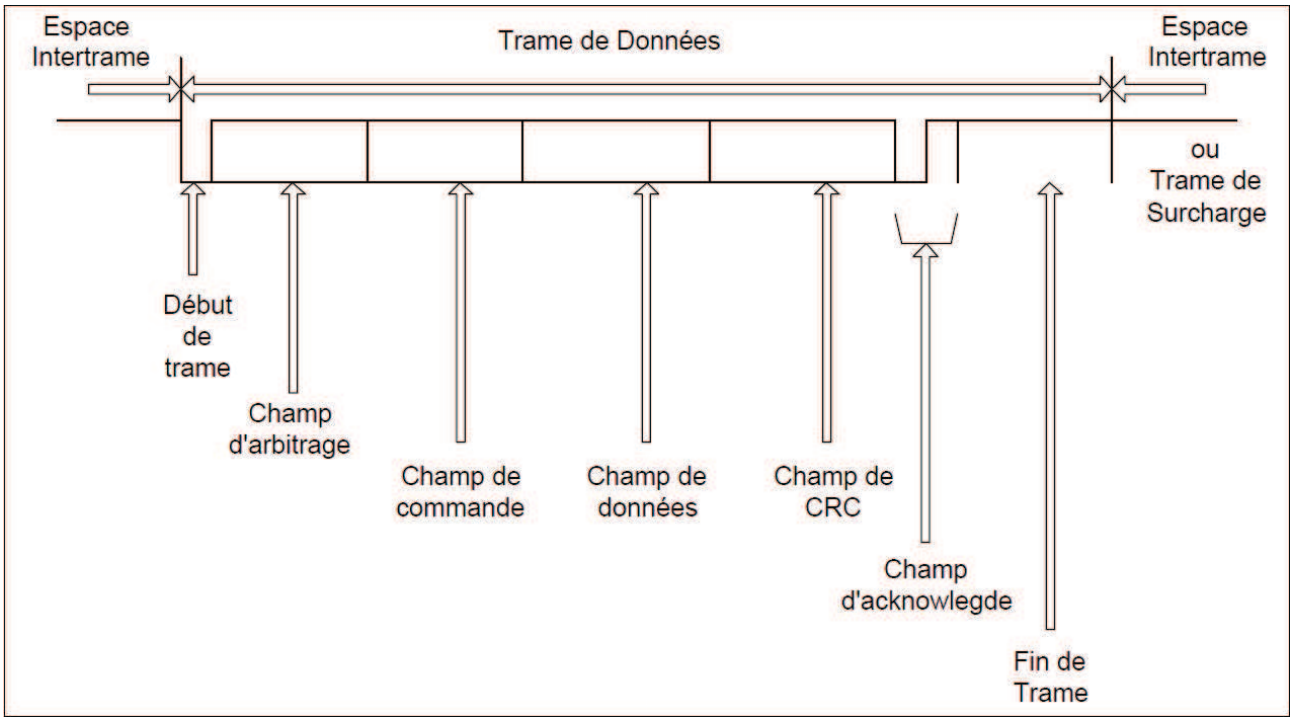


Figure I.8 – Format de la trame CAN : Source [69]

mécanisme utilisé pour gérer ces priorités se trouve dans [126].

2.7.2 Principes de Fonctionnement

Chaque nœud recevant un message regarde si celui-ci est intéressant pour lui grâce à l'ID. Si c'est le cas, il le traite, sinon, il l'ignore. Cet unique ID indique aussi la priorité des messages. Plus la valeur est faible, plus le message sera prioritaire. Si deux nœuds ou plus cherchent à avoir accès au bus en même temps, c'est celui de plus haute priorité qui gagne. Les messages de priorité inférieure seront automatiquement retransmis lorsque le bus sera libre [107].

La couche physique ne fait l'objet d'aucune spécification dans la norme CAN. Par conséquent, plusieurs implémentations sont possibles. La seule contrainte que doit respecter toute implémentation est la règle des bits dominants / récessifs. Il est nécessaire pour chaque nœud de pouvoir présenter un bit dominant (0 logique) et un bit récessif (1 logique) sur le bus : le bit dominant est prioritaire sur le bit récessif.

L'accès au bus CAN est fonction de la priorité. Les nœuds ne possèdent pas d'adresse. Aucun nœud ne joue un rôle prépondérant dans le protocole. Chaque message possède un identificateur, unique à tout le système. Cet identificateur permet filtrage et l'attribution d'un niveau de priorité pour la transmission. En fonction de la taille de l'identificateur, on distingue

2 versions de CAN : CAN 2.0A (identificateur sur 11 bits) et CAN 2.0B (identificateur sur 29 bits). Les données sont segmentées en trames transmises périodiquement ou sporadiquement.

2.7.3 Analogie avec l'ordonnancement à priorités fixes de tâches non préemptives

Pour le calcul du pire temps de réponse des messages, il est important de savoir que :
- les trames ont des priorités uniques basées sur les identifiants ;
- parmi l'ensemble des trames transmises à un instant, celle qui a la plus haute priorité gagne l'arbitrage et est effectivement transmise ;
- le mécanisme qui permet d'arbitrer la transmission des trames se fait en temps nul ;
- une trame est transmise sans interruption (hors dysfonctionnement).

Un tel comportement peut être comparé à celui d'un ordonnancement à priorités fixes de tâches non préemptives en monoprocesseur. Une analyse détaillée consacrée au temps de traversée de bout en bout d'un réseau CAN se trouve dans [51] et sera présentée au paragraphe III.4.

2.8 Le réseau AFDX

Le réseau AFDX s'appuie sur la norme ARINC 664 pour proposer une architecture avionique modulaire intégrée. Il fournit des moyens de partage des ressources, de ségrégation des flux ainsi que le déterminisme et la disponibilité requise pour les certifications aéronautiques.

2.8.1 Réseau Ethernet commuté Full-Duplex

L'Ethernet Commuté Full Duplex consiste en l'utilisation de liaisons bidirectionnelles exerçant en mode Full Duplex. Avec le mode Full-duplex, il n'est pas nécessaire de retarder l'émission d'un message, ni même d'écouter ou de réagir à l'activité sur le médium physique, car l'existence de collision a été supprimée. De ce fait, le besoin d'utilisation du CSMA/CD ne se pose plus. Cependant, la suppression de CSMA/CD ne rend pas toute l'architecture déterministe. Ce problème de déterminisme a juste été déplacé dans les commutateurs, où les différents flux vont entrer en compétition pour l'utilisation des ressources du commutateur. Ce qui peut aboutir à la congestion des commutateurs, avec comme conséquence une perte de trames. Un des buts de notre étude est de comprendre si de telles congestions peuvent ou non intervenir dans le réseau.

Le réseau Ethernet n'est pas conçu à la base pour des applications temps réels. Si le réseau Ethernet commuté Full-Duplex se prête bien à ce type d'application, son exploitation dans un

domaine aussi contraignant que l'aéronautique nécessite quelques adaptations. C'est le rôle de la norme ARINC 664 qui a été développée dans ce sens.

2.8.2 La notion de VL

Un point très importante dans un réseau AFDX est la notion de VL (virtual link ou lien virtuel). Le VL permet une représentation du trafic. L'idée principale derrière cette notion est la volonté de fournir une ségrégation des flux, ce qui est imposé par la norme pour raison de sécurité : le mauvais comportement d'un flux ne doit pas nuire aux autres flux. La ségrégation robuste des flux de données s'appuie sur la réservation de bande passante au niveau d'un VL. Cette ségrégation est assurée au niveau des commutateurs par un mécanisme de listes de contrôle d'accès (ACL) filtrant le trafic en fonction des adresses. En résumé, le VL permet de virtualiser un bus avionique classique pour chaque flux, où il serait le seul flux à émettre [59].

Ce qui facilite la garantie du respect des contraintes temps réel est l'obligation d'un VL à respecter des contraintes liés à la bande passante. Ainsi, il est nécessaire pour un VL de fixer des paramètres comme la taille maximale des trames transmises et le temps minimum qui sépare l'arrivée de deux trames consécutives. Le commutateur utilise ces deux paramètres pour contrôler qu'aucun VL ne consomme une bande passante supérieur à sa bande passante maximale.

Plus précisément, un VL est défini par :

— un identifiant unique,
— une adresse de destination (*unique ou multicast*),
— la route empruntée pour rallier ces destinations,
— la taille maximale et minimale d'un paquet (en bits, notées s_{max} et s_{min}),
— un temps minimum d'émission entre deux trames consécutives, appelé BAG (Bandwidth Allocation Gap).

On voit immédiatement que ces dernières données permettent de définir la bande passante maximale du lien : il ne peut émettre au maximum qu'un paquet de taille maximale toutes les BAG. Son débit maximal en bits par seconde, noté ρ, est donc : $\rho = \frac{s_{max}}{BAG}$

Chapitre II

Méthodes d'analyse

Nous présentons ici un état de l'art des méthodes d'analyse de performance temps réel. Nous commençons par présenter quelques éléments de classification. Celle-ci peut être lié au système analysée où à la méthode d'analyse elle-même. Ensuite, nous traitons de l'ordonnancement temps réel. Nous donnons ici les résultats d'une méthode d'évaluation de délai pire cas à l'aide d'ordonnancement. Une illustration est faite avec le réseau CAN. Nous continuons avec une présentation du model-checking, de l'approche par trajectoire et d'une méthode basée sur les Event Stream. Nous terminons avec le calcul réseau. Un rappel des principaux résultats de calcul réseau est donné avec un exemple d'application et les défis du calcul réseau.

1 - Éléments de classification

1.1 Périodique vs Non périodique

Une tâche peut s'exécuter à des intervalles réguliers (tâches périodiques) ou de manière aléatoire (tâches non périodiques).

1.1.1 Tâches périodique

On dit que la tâche t_i est périodique de période $T(t_i)$ si l'événement qui conditionne l'activation de cette tâche se produit à des intervalles de temps réguliers $T(t_i)$. Donc le service fourni par cette tâche périodique est rendu indéfiniment. Pour cette raison, chacune de ces exécutions est appelée instance.

Les figures II.1 et II.2 représentent deux types de tâches périodiques : i) le premier type est ce qu'on peut appeler la période classique puisque c'est le plus utilisé dans les applications temps réel. Chaque instance de la tâche t_i doit s'exécuter entièrement à l'intérieur de l'intervalle de longueur $T(t_i)$ qui démarre à la date d'activation de cette instance (voir la figure II.3) ; ii) le deuxième type est appelé période stricte. À la différence de la période classique, les instances

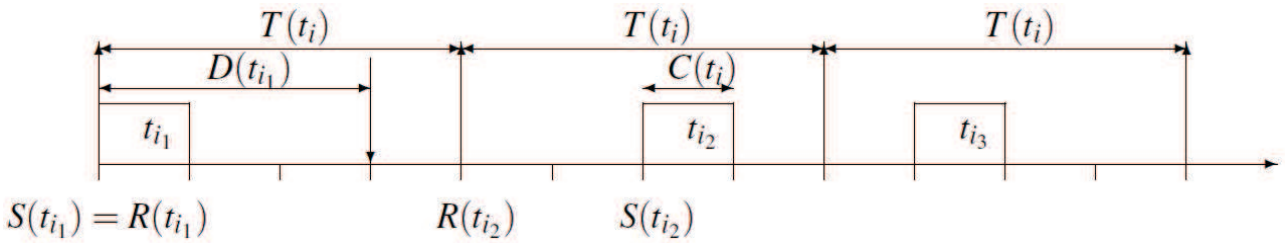


Figure II.1 – Représentation graphique du modèle temps réel [74]

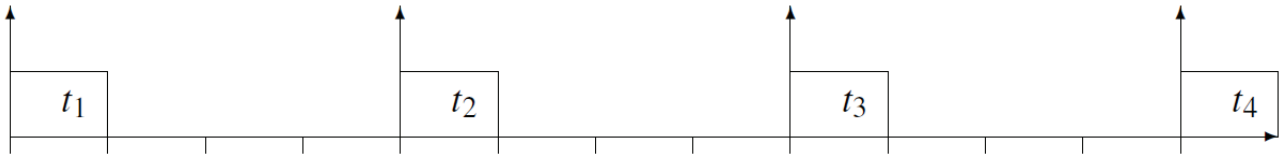


Figure II.2 – Périodicité stricte [74]

d'une tâche périodique stricte t_i doivent démarrer exactement au début de chaque intervalle de longueur $T(t_i)$ (voir la figure II.2). On peut remarquer que la période stricte est inclue dans la période classique et que dans ce cas, $R(t_i)$, la date à laquelle la tâche t_i peut commencer son exécution coïncide avec $S(t_i)$, la date de début effectif de son exécution ($R(t_i) = S(t_i)$).

1.1.2 Tâches non périodique

Il existe deux types de tâches non périodiques :

— **apériodique** : les dates d'activations sont aléatoires et ne peuvent être anticipées. Son exécution est le produit d'événements internes ou externes qui peuvent se déclencher à tout instant.

— **sporadique** : c'est un cas particulier des tâches apériodiques où une durée de temps minimale sépare deux activations.

Cette durée minimale s'apparente à une période. En réalité, tâches périodiques et tâches sporadiques se ressemblent beaucoup, d'ailleurs, elles sont strictement identiques en calcul réseau et sont représentées de la même façon.

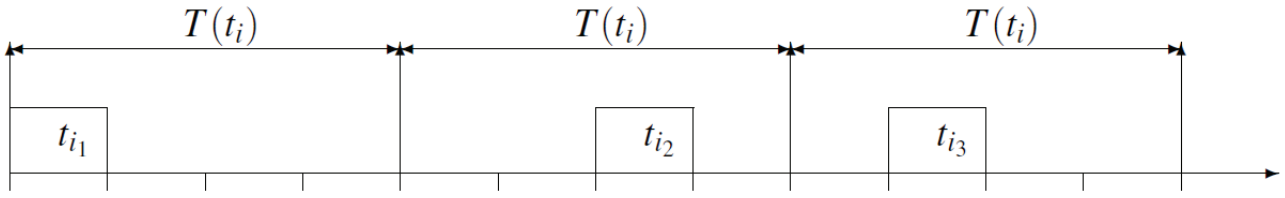


Figure II.3 – Périodicité classique [74]

1.2 Méthodes globales Vs méthodes locales

Si on s'intéresse à la façon dont les méthodes d'analyse de performance travaillent, on peut les classer en deux grandes familles : les méthodes dites globales et le méthodes dites locales.

Les méthodes locales commencent par considérer les différents nœuds du réseau, puis elles mènent une analyse au niveau de chacun de ces nœuds. Les approches locales proposent généralement un modèle de représentation des des flux d'entrées et de sortie d'un nœud. Ensuite, elles proposent des règles de combinaison servant à propager les résultats de la sortie d'un nœud vers l'entrée du nœud suivant dans la topologie.

À l'inverse, les approches globales considèrent le réseau comme un tout et proposent directement une analyse de celle-ci. Ces approches ont le mérite d'être plus précises mais ne s'appliquent qu'aux réseaux de petite taille contrairement aux méthodes globales qui savent gérer les réseaux de grande taille mais qui doivent aussi payer le prix de l'imprécision induite par leurs règles de combinaison.

2 - Analyse d'un algorithme d'ordonnancement temps réel : exemple du réseau CAN

2.1 Résultats de l'analyse du réseau CAN

L'analyse du temps de réponse sur le bus CAN, qui a fait l'objet d'une présentation au paragraphe I.2.7, page 27, a pour but de fournir une méthode d'évaluation du pire temps de réponse de chaque message. Ce sujet à fait l'objet de plusieurs études [128, 127, 129, 126, 51].

On considère un ensemble de flux k, possédant les propriétés suivantes : J_k la gigue, T_k la période, C_k la durée de transmission.

Pour évaluer le délai pire cas d'un flux R_m, la méthode commence par déterminer la valeur de t_m, qui représente la longueur de la période d'activité de ce flux. t_m est donné par la relation suivante.

$$t_m^{n+1} = B_m + \sum_{\forall k \in hp(m) \cup m} \left\lceil \frac{t_m^n + J_k}{T_k} \right\rceil C_k \tag{II.1}$$

Cette relation de récurrence commence avec $t_m^0 = C_m$ et s'achève avec lorsque $t_m^{n+1} = t_m^n$. $B_m = \max_{k \in lp(m)} (C_k)$ est la durée maximale de temps dû à l'attente d'un paquet de faible priorité. $lp(m)$ est l'ensemble des flux de priorité plus faible que celle du flux R_m.

Une fois le paramètre t_m calculé, la méthode continue avec l'évaluation de Q_m, le nombre

de messages du flux R_m devant être examiné.

$$Q_m = \left\lceil \frac{t_m + J_m}{T_m} \right\rceil \tag{II.2}$$

La phase suivante est l'évaluation de w_m, le délai dû à l'attente dans la file d'attente. Sa valeur est donnée par la relation suivante.

$$w_m^{n+1}(q) = B_m + qC_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k \tag{II.3}$$

Cette relation de récurrence commence avec la valeur $w_m^0(q) = B_m + qC_m$ et se termine lorsque $w_m^{n+1}(q) = w_m^n(q)$. Enfin, le délai est calculé en utilisant l'équation II.4

$$R_m(q) = J_m + w_m(q) - qT_m + C_m \tag{II.4}$$

Le temps de traversée pire cas s'obtient alors par avec l'équation II.5 suivante.

$$R_m = \max_{q=0...Q_m-1} (R_m(q)) \tag{II.5}$$

2.2 Exemple

Pour illustrer, nous reprenons l'exemple décrit dans [51] et représenté par le tableau III.4 de la page 90. Le délai pire cas pour le flux R_3 se calcule de la façon suivante. Puisqu'il n'existe pas de flux moins prioritaire que R_3, $B_3 = 0$. Partant de la valeur $t_3^0 = C_3 = 1$, la relation de récurrence donnée par l'équation II.1 itère comme suit : $t_3^1 = 3$, $t_3^2 = 4$, $t_3^3 = 6$, $t_3^4 = 7$, pour converger lorsque $t_3^5 = t_3^4 = 7$. La taille de la période d'activité est donc de 7ms et le nombre de messages du flux R_3 devant être examiné est donné par l'équation II.2

$$Q_3 = \left\lceil \frac{7,0}{3,5} \right\rceil = 2$$

L'évaluation du délai pour le premier message en utilisant l'équation II.3 donne : $w_3^0(0) = 0$, $w_3^1(0) = 2$, convergeant avec $w_3^2(0) = w_3^1(0) = 2$. En utilisant l'équation II.4, on obtient : $R_3(0) = 3$. Pour le deuxième message, on obtient : $w_3^0(1) = w_3(0) + C_m = 3$, $w_3^1(1) = 4$, $w_3^2(1) = 5$, $w_3^3(1) = 6$, et s'arrêtant avec $w_3^4(1) = w_3^3(1) = 6$. Soit finalement $R_3(1) = 3,5ms$. D'après l'équation II.5, c'est le délai pire cas subit par un message du flux R_3.

2.2.1 Points forts et points faibles

Le principal avantage de cette approche est qu'elle donne le vrai pire délai. Par contre, elle se limite à un service de débit fixe et ne dit rien sur la capacité de service non utilisée (le service résiduel). On ne peut donc pas l'utiliser dans le cadre d'ordonnancement hiérarchique (cf. paragraphe III.3, page 108).

3 - Le modèle checking

3.1 Généralités

Le model-checking est une technique de vérification qui s'applique à une large classe de systèmes : ceux qui sont modélisables par un automate fini [10] (ou une variante de cette représentation générale). Elle comporte trois étapes [117].

— Représentation d'un programme ou d'un système par un automate.

Sommairement, un automate est un modèle de machine évoluant d'état en état sous l'effet de transitions. Dans un premier temps, le système est divisé en sous-système et chaque sous-système est décrit à partir des notions d'états et de transitions. Le système complet est ensuite modélisé par un nouvel automate, obtenu par composition et synchronisation des différents composants. Enfin, les comportements possibles du système sont modélisés par des exécutions de l'automate global, l'ensemble de ces exécutions formant un arbre.

— Représentation d'une propriété par une formule logique.

Une fois le modèle construit, on énonce de façon formelle les propriétés qu'il doit vérifier, usuellement dans une logique temporelle. Seulement, il faut une certaine habitude pour décrire des énoncés en logique temporelle, et une habitude encore plus grande pour lire les énoncés écrits par d'autres. C'est un des obstacles qui s'opposent à une plus grande diffusion des techniques du model-checking. De plus, le model-checking dépend de la logique temporelle sous-jacente. Par exemple, PLTL, aussi appelée logique du temps linéaire voit le système comme un ensemble d'exécution, tandis que CTL, aussi appelée logique temporelle arborescente voit le système comme un arbre de comportements possibles.

— Algorithme de model-checking.

Un modèle $_{\mathbf{A}}$ et une propriété φ étant donnés, un algorithme de model-checking répond à la question : « le modèle $_{\mathbf{A}}$ satisfait-il la propriété φ » ? Les techniques mises en œuvre dans ces algorithmes utilisent des méthodes explicites ou des méthodes symboliques. Le principal obstacle que rencontrent les algorithmes de model-checking, à base de méthodes explicites, est le problème dit « de l'explosion du nombre d'états ». Cette explosion se produit chaque fois

que l'on décide d'énumérer explicitement en mémoire tous les états de l'automate examiné. Une tentative de résolution de ce problème se trouve dans le model-checking symbolique. Le terme model-checking symbolique s'applique à toute méthode de model-checking qui chercherait à représenter de façon symbolique (par opposition à « de façon explicite ») les états et les transitions d'un automate à vérifier. Par ailleurs, on utilise souvent ce terme pour désigner une méthode symbolique particulière oú l'on utilise des diagrammes de décision binaires pour représenter les ensembles d'états. L'idée de fond des méthodes symboliques est de pouvoir représenter de façon concise des ensembles très grands d'états et de les manipuler en quelque sorte par paquets. Une telle approche devient moins sensible à la finitude du nombre total d'états et peut s'appliquer également à des systèmes à infinité d'états. En ce qui concerne les systèmes temps-réels, un autre type de model-checking est nécessaire : les automates temporisés. Pour de tels modèles, oú la notion de temps est manipulée de manière explicite, les propriétés sont exprimées à l'aide d'une extension de la logique temporelle (TCTL). Dans ce cadre, les algorithmes du model-checking sont compliqués. Le principal problème réside dans la complexité des procédures de décision et en pratique, le nombre d'horloges est déterminant pour la faisabilité du model-checking. Le nombre d'états du modèle déplié est infini, ce qui explique là encore, une méthode symbolique pour le model-checking. En somme, le model-checking constitue une technique puissante mais limitée : sa puissance provient du fait qu'elle est exhaustive et automatique. Une réponse positive garantit la propriété pour tous les comportements du modèle. Une réponse négative est généralement accompagnée d'un contre-exemple exhibant une exécution particulière du système violant la propriété souhaitée ; sa limitation est essentiellement due aux barrières de complexité : la vérification exhaustive, même avec des méthodes symboliques peut s'avérer d'un coût excessif, en temps et/ou en espace. En pratique, la taille des systèmes est bien le principal obstacle qui reste à franchir. Un système très simple à manipuler pour un simulateur, un compilateur ou un interpréteur peut être hors de portée d'un model-checker. Pour augmenter les performances des méthodes de model-checking, plusieurs approches se penchent sur la réduction de l'espace d'états à analyser [60, 4, 5, 6].

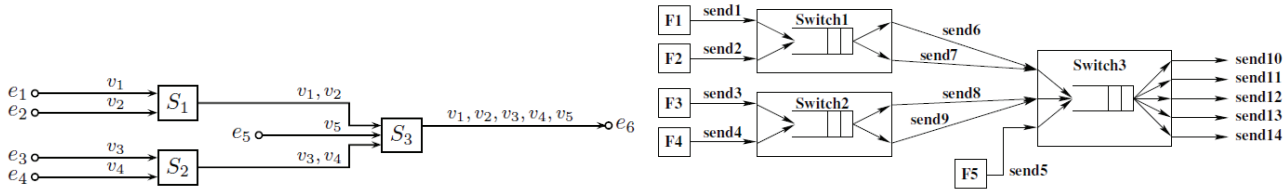
3.2 Le model-checking et les systèmes temps-réels

L'approche par model checking a déjà été mis en œuvre pour caractériser des délais sur plusieurs architectures réseau, notamment sur des configurations simplifiées d'un réseau AFDX [40, 11, 4, 5, 6]. L'obtention des délais pire cas de bout en bout avec le model-checking se fait à partir des modèles formels d'automates temporels. Le model checking est utilisé pour déterminer la latence de transmission globale de chaque trame dans le système tout en vérifiant que la trame

est reçue avant le délai de transmission global. En d'autre terme, on vérifie la propriété que pour un message donné (mi), le délai de transmission global d'une trame ($trame_i$), noté $d(trame_i)$ doit être inférieur à un délai borné d_i : $d(trame_i) \leq d_i$ [39]. Cette méthode permet de construire la vérification. Un automate de test codant la propriété est alors construit. Un model-checker permet alors de vérifier si un rejet de nœud est accessible ou non. Cette approche permet non seulement d'atteindre le vrai délai pire cas pour des petites configurations, mais aussi d'exhiber le scénario correspondant. La première approche à base de model-checking a été proposée dans [40]. Elle est basée sur les automates temporisés et calcule le vrai délai pire cas de bout en bout de chaque VL. Une telle approche ne peut cependant pas être appliquée sur une configuration AFDX avec plus de 8 VLs pour les raisons bien connues du problème d'explosion combinatoire du nombre d'états qui est reconnu au model-checking. Pour contourner ce problème, [4] propose une approche diviser-pour-régner. Celle-ci consiste à calculer le vrai délai pire cas au niveau de chaque sortie rencontrée sur un chemin d'un VL donné. Le délai de bout en bout d'un chemin s'obtient par propagation consécutive du délai d'une sortie sur la suite du chemin. Bien-que cette approche permette de gérer une configuration plus grande qu'avec les précédentes (15 VLs strictement périodiques dans une configuration AFDX à 2 commutateurs), elle ne conduit pas toujours au vrai délai pire cas. Dans certaines configurations en effet, les résultats produits souffrent d'une sous-estimation du réel délai [5]. Ce problème est pris en compte dans [5, 6] qui propose une technique permettant de caractériser les scénarios susceptibles de conduire au vrai pire cas. Ce qui permet une réduction considérable de l'ensemble de scénarios à analyser. [6] propose un model-checking basé sur cet ensemble réduit de scénario, applicable pour évaluer le vrai délai d'une configuration AFDX allant jusqu'à 50 Vls strictement périodiques. Au delà de 50 Vls, [6] propose un algorithme qui donne des valeurs de délai atteignables, « proches » du vrai délai pire cas.

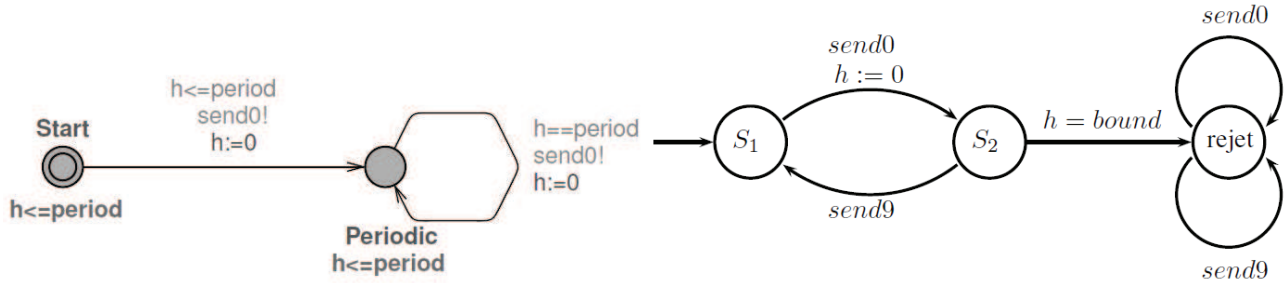
3.3 Les outils du Model-checking

Il existent plusieurs outils logiciels qui implémentent les concepts du model-cheking. Ces model-checkers ont permis, non seulement, de découvrir des erreurs dans des systèmes, mais aussi de valider d'autres. Chacun de ces outils vise un domaine d'utilisation particulier. Ainsi, on peut citer SVM [101] qui est un model-checker symbolique pour le hardware et les systèmes embarqués, SPIN pour les systèmes distribués, Maude pour vérifier des propriétés temporelles sur des systèmes distribués concurrents, CADP pour l'ingénierie des protocoles et UPPAAL [81] pour les systèmes temps-réels. Le succès et les limitations de ces outils sont à l'origine d'une importante activité de recherche autour du domaine. [133] propose une classification des model-



(a) Architecture du réseau **(b)** Modèle correspondant

Figure II.4 – Modélisation d'une architecture AFDX par Model Checking [13]



(a) Automate associé à un émetteur **(b)** Automate de test du VL v_1

Figure II.5 – Modèles d'automates temporisés utilisé pour la vérification de modèle [13]

checkers axée sur la comparaison de leurs architectures techniques. Les model-checkers dédiés aux systèmes temps-réels sont basés sur la logique temporelle temporisée. Dans ce registre, si le plus répandu est UPPAAL [40, 82, 11, 4, 5], on trouve aussi KRONOS [40] et HyTech [40]. Une comparaison de ces trois model-checkers se trouve dans [20].

3.4 Exemple d'application : Model checking avec le réseau AFDX

Nous reprenons ici un exemple simple de [13] qui montre la modélisation d'une configuration AFDX complète. La configuration présentée dans cet exemple est illustrée à la figure II.4a. Le système global est obtenu en composant les automates des VL et des ports de sortie des commutateurs. Les cinq VL et quatre ports de sortie se traduisent en neuf automates, comme l'illustre la figure II.4b. À titre d'exemple, le VL v_2 est modélisé par l'automate v_2, qui génère le signal $send_1$. Celui-ci est reçu par l'automate qui modélise l'unique port de sortie du commutateur S_1. Le cheminement de v_2 se poursuit via les signaux $send_5$ et $send_10$. Partant de ce modèle, le délai pire cas pour chaque VL est obtenu en utilisant la méthode de l'automate de test [19, 12]. L'automate de test correspondant au VL v_1 est présenté sur la figure II.5b. Il modélise la propriété « délai de v_1 inférieur à bound ». Lorsque le signal $send_0$ est reçu, on passe dans l'état s_2. On attend alors le signal $send_9$ (transmission du VL v_1 en sortie du commutateur S_3). S'il

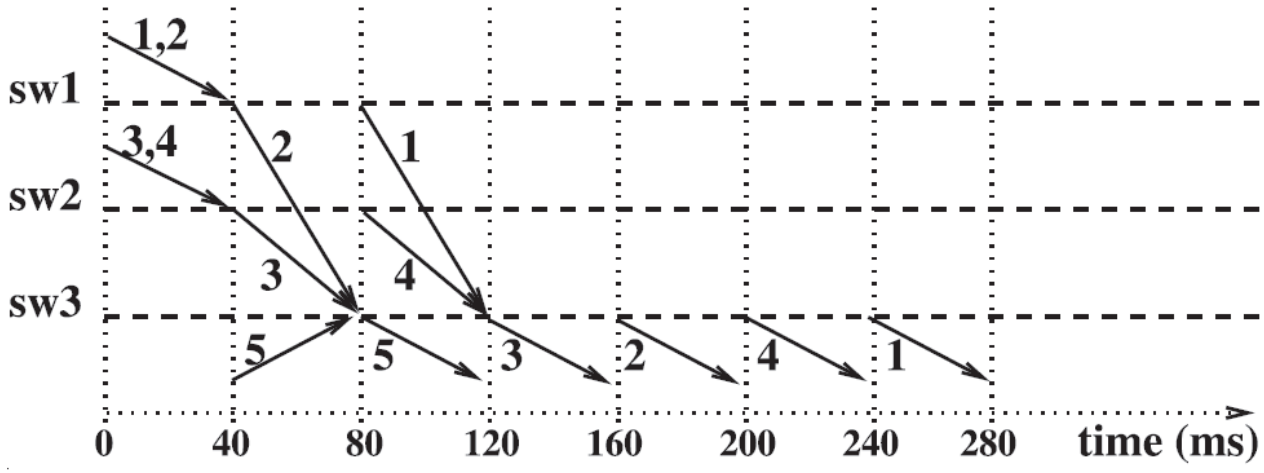


Figure II.6 – Scénario pire cas pour le VL v_1 [13]

n'est pas arrivé à l'issue d'une durée bound, on passe dans l'état $send_5$. Cela signifie qu'il existe au moins un scénario où v_1 dépasse la borne bound. Le principe consiste donc à déterminer la plus petite valeur de bound pour laquelle l'état rejet est atteignable. On obtient en outre un scénario correspondant à cette valeur maximale de délai de bout en bout. Le scénario pire-cas pour le VL v_1 est présenté à la figure II.6

L'outil Uppaal permet d'effectuer la vérification sur cet exemple en moins d'une seconde sur une station Linux munie d'un processeur Pentium 4 et de 2 Go mémoire vive [13]. Pour ce type de configuration réduite, l'approche par vérification de modèle offre donc la possibilité de connaître le scénario pire cas et le délai de bout en bout correspondant. Cependant, l'augmentation de la taille du réseau et du nombre de flux à considérer entraîne une explosion combinatoire pour l'outil de vérification qui se traduit par une explosion du temps de calcul et de la mémoire nécessaire.

4 - La méthode des trajectoires

L'approche par trajectoires est une théorie basée sur l'ordonnancement visant à borner les délais de traversée d'un système composé de différents éléments informatiques (e.g. réseau de commutateurs) appelés nœuds. Elle est constituée d'un ensemble de résultats qui ont été présentés dans [99, 100], avant d'être considérablement enrichis par d'autres auteurs [14, 15, 16, 17, 13].

Très récemment, des erreurs ont été trouvée dans cette méthode, qui montrent que les bornes calculées peuvent être plus petites que le véritable pire cas [73], [72]. À ce jour, nous ne

savons pas si ces erreurs seront corrigées sans changer beaucoup la théorie où si leur correction impliquera des modification majeures de l'approche.

Nous avons décidé de présenter tout de même l'approche dans cette thèse, une fois cet avertissement fait.

4.1 Principe de l'approche

L'approche par trajectoires considère un système distribué composé de nœuds et de flux qui parcourent ces différents nœuds. Chaque paquet transitant à travers ce réseau est une tâche qui s'exécute sur les différents nœuds parcourus. Les nœuds exécutent les tâches en attente en respectant une stratégie d'ordonnancement. L'approche par trajectoires prend en considération le scénario pire cas pour un paquet tout au long de sa trajectoire [13].

4.2 Contexte

Les travaux originels de [99] considèrent que tout paquet entrant dans le réseau se voit assigner une priorité généralisée, c'est-à-dire une priorité fixe (celle du flux auquel il appartient) et une priorité dynamique. Ainsi, dans chacun des nœuds du réseau, les paquets sont ordonnancés selon leurs priorités généralisées. Plus précisément, les paquets sont traités dans l'ordre de leurs priorités fixes (ordonnancement FP) puis ceux de même priorité fixe dans l'ordre de leurs priorités dynamiques (ordonnancement DP*). Les flux sont traités dans chacun des nœuds visités suivant un ordonnancement de type FP/DP*. Comme il a été montré dans [99], et plus tard dans [13], ces résultats avec la politique FP/DP* s'adaptent bien avec les politiques FP (les flux ont une priorité fixe différente : le traitement DP* n'intervient pas dans l'ordonnancement FP/DP*, puisque DP* permet de départager des paquets ayant la même priorité fixe) et DP* (tous les flux partagent la même priorité fixe : le traitement FP n'intervient pas dans l'ordonnancement FP/DP*).

L'approche par trajectoires s'intéresse au temps de départ au plus tard d'un paquet m appartenant au flux τ_i et suivant le chemin $Pi = \{1, ..., q\}$ dans son dernier nœud. Son temps de traversée se déduit de ce temps de départ au plus tard par soustraction avec son instant de génération, c'est-à-dire son instant d'arrivée sur son premier nœud. Le paquet m traverse q nœuds. La figure II.7 montre un exemple de configuration avec trois nœuds ($q = 3$) et six flux ($n = 6$). L'approche présentée dépend de la topologie du réseau. On distingue le cas d'une ligne de diffusion et le cas distribué.

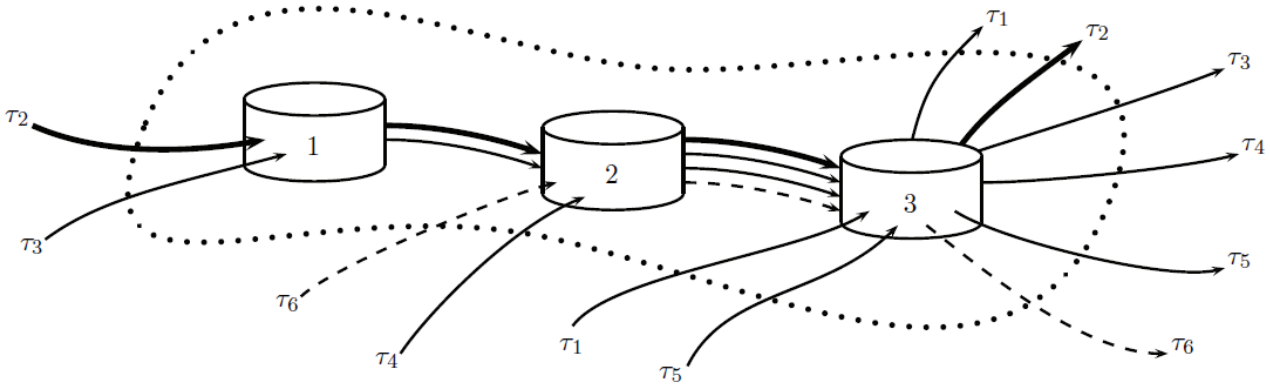


Figure II.7 – Exemple de configuration pour le calcul par trajectoires [13]

4.3 Cas d'une ligne de diffusion

C'est le cas où les flux suivent tous une même ligne de diffusion, c'est-à-dire une même séquence de nœuds.

Voici une description de l'approche par trajectoire pour déterminer le temps de réponse pire cas de bout en bout d'un flux quelconque τ_i, $i \in [1, n]$, lorsque les flux suivent tous une même ligne de diffusion et que les paquets sont ordonnancés suivant un algorithme du type FP/DP* dans chacun des nœuds visités.

L'approche consiste à examiner l'ordonnancement produit sur l'ensemble des nœuds visités par un flux. Plus précisément, étant donné un paquet quelconque m de τ_i généré à l'instant t, la première étape consiste à identifier sur chacun de ses nœuds visités (en commençant par le dernier puis en remontant de nœud en nœud) la période active et les paquets affectant son temps de réponse de bout en bout. Le procédé est répété jusqu'à la détermination, sur le nœud source, du premier paquet ayant affecté le temps de réponse de bout en bout du paquet m. Le temps passé par m dans le réseau est alors égal à l'instant d'activation du premier paquet gênant m dans le nœud source, plus la somme des temps de traitement des paquets identifiés dans les périodes actives considérées, moins l'instant de génération du paquet m [99].

Pour cela, considérer la période active de niveau $PG_i(t)$ dans laquelle m est traité sur le nœud q. Soient bp^q cette période active et $f(q)$ le premier paquet traité dans bp^q de priorité supérieure ou égale à $PG_i(t)$. Le paquet $f(q)$ a été traité sur le nœud $q-1$ dans une période active de niveau au moins égal à $PG_i(t)$. Soient bp^{q-1} cette période active et $f(q-1)$ le premier paquet traité dans bp^{q-1} avec une priorité supérieure ou égale à $PG_i(t)$. On procède ainsi, de proche en proche jusqu'à la détermination sur le nœud 1 de la période active bp^1 dans laquelle $f(2)$ est traité et où $f(1)$ est le premier paquet de priorité supérieure ou égale à $PG_i(t)$. La figure II.8 illustre cette décomposition.

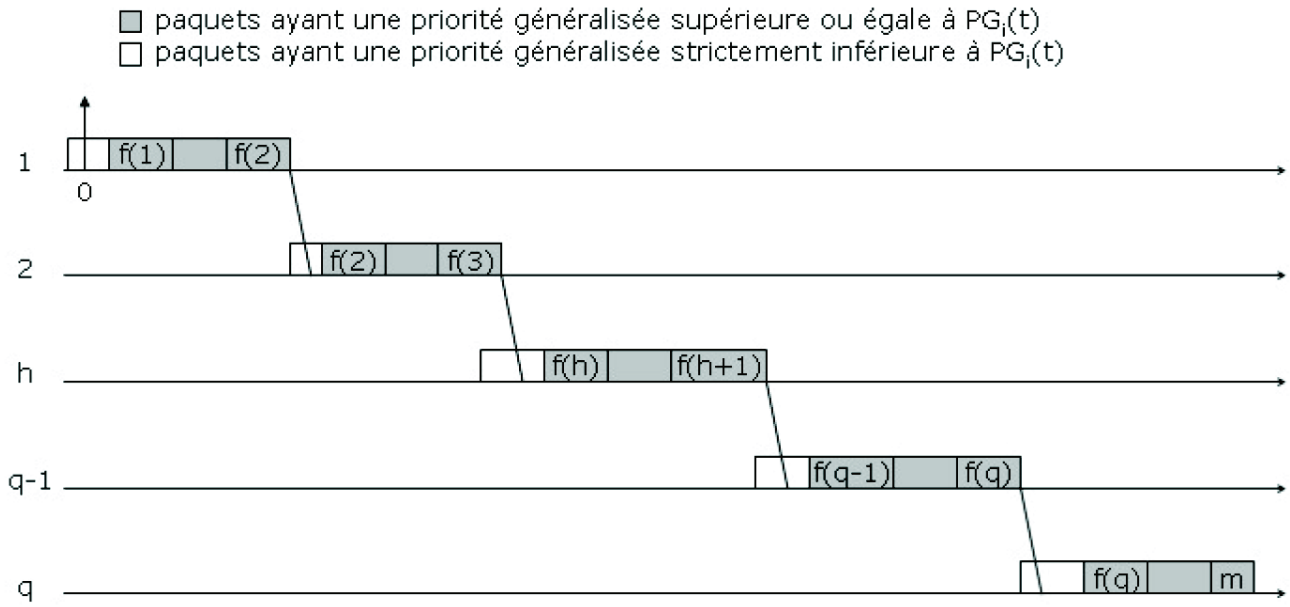


Figure II.8 – Détermination du temps de réponse de bout en bout avec l'approche par trajectoire [99]

A partir de cette décomposition, W_i^q l'instant de démarrage au plus tard du paquet m sur le nœud q s'obtient en sommant les durées suivantes :

- le temps de traitement sur le nœud 1 des paquets $f(1)$ à $f(2) + L_{\max}$
- le temps de traitement sur le nœud 2 des paquets $f(2)$ à $f(3) + L_{\max}$
- le temps de traitement sur le nœud q des paquets $f(q)$ à $(m-1)$
- ...
- le temps de traitement sur le nœud q des paquets $f(q)$ à $(m-1)$
- $\delta_i^{1,q}(t)$, le délai maximum subi par m dans les nœuds 1 à q à cause de l'effet non-préemptif direct.

Soit finalement :

$$W_i^q(t) = \sum_{h=1}^{q} \left(\sum_{g=f(h)}^{f(h+1)} C_{\tau(g)}^h \right) - C_i^q + \delta_i^{1,q}(t) + (q-1) \cdot L_{\max} \tag{II.6}$$

4.4 Cas distribué

Dans un contexte de topologie réseau plus générale où les flux suivent des lignes de diffusion quelconques et sont ordonnancés FP/DP* dans chacun des nœuds visités, pour déterminer le temps de réponse pire cas de bout en bout d'un flux quelconque τ_i, $i \in [1, n]$, l'approche par trajectoire précédemment décrite pour une ligne de diffusion été étendue. Plus précisément, [99]

s'est intéressé à un paquet quelconque m du flux τ_i généré à l'instant t et a identifié sur chacun des nœuds visités par ce paquet (en commençant par le dernier puis en remontant de nœud en nœud) la période active et les paquets affectant son temps de réponse de bout en bout. Cette décomposition permet d'établir l'instant de démarrage au plus tard du paquet m sur son nœud de sortie. Après une analyse de cet instant dont l'expression est une formule récursive, le temps de réponse pire cas de bout en bout du flux τ_i est obtenu.

[99] s'intéresse à un flux quelconque i, $\in [1, n]$, suivant une ligne de diffusion L_i composée de q nœuds numérotés de 1 à q. Avant de déterminer le temps de réponse pire cas de bout en bout du flux i, une attention est accordée à l'un de ses paquets, m, généré à un instant t. Son instant de démarrage au plus tard sur le dernier nœud visité, le nœud q, est calculé. Pour ce faire, il convient d'identifier les périodes actives qui retardent l'instant de démarrage au plus tard de m sur le nœud q.

Soient bp^q, la période active de niveau $PG_i(t)$ dans laquelle m est traité sur le nœud q et $f(q)$ le premier paquet traité dans bp^q avec une priorité supérieure ou égale à $PG_i(t)$. Puisque les flux ne suivent pas tous la même ligne de diffusion, le paquet $f(q)$ ne vient pas nécessairement du nœud $q-1$. Par conséquent, afin de remonter sur les nœuds visités par m, il convient de considérer un paquet supplémentaire, à savoir $p(q-1)$: le premier paquet de priorité supérieure ou égale à $PG_i(t)$ traité entre $f(q)$ et m sur le nœud q et venant du nœud $q-1$. Le paquet $p(q-1)$ a été traité sur le nœud $q-1$ dans une période active de niveau au moins égal à $PG_i(t)$. Soient bp^{q-1} cette période active et $f(q-1)$ le premier paquet traité dans bp^{q-1} avec une priorité supérieure ou égale à $PG_i(t)$. On défini alors $p(q-2)$: le premier paquet de priorité supérieure ou égale à $PG_i(t)$ traité entre $f(q-1)$ et m sur le nœud $q-1$ et venant du nœud $q-2$. Ce procédé est ainsi répété jusqu'à la détermination sur le nœud 1 de la période active bp^1 dans laquelle $p(1)$ est traité et où $f(1)$ est le premier paquet de priorité supérieure ou égale à $PG_i(t)$. Par conséquent :

- $\forall h \in [1, q[$, $f(h)$ est le premier paquet traité dans bp^h avec une priorité supérieure ou égale à $PG_i(t)$;
- $\forall h \in]1, q]$, $p(h-1)$ est le premier paquet de priorité supérieure ou égale à $PG_i(t)$ traité entre $f(h)$ et m sur le nœud h et venant du nœud $h-1$.

La figure II.9 illustre cette décomposition.

A partir de cette décomposition, W_i^q l'instant de démarrage au plus tard du paquet m sur le nœud q s'obtient en sommant les durées suivantes :

- le temps de traitement sur le nœud 1 des paquets $f(1)$ à $p(1) + L_{\max}$
- le temps de traitement sur le nœud 2 des paquets $f(2)$ à $p(2) - \left(a^2_{p(1)} - a^2_{f(2)}\right) + L_{\max}$
- ...

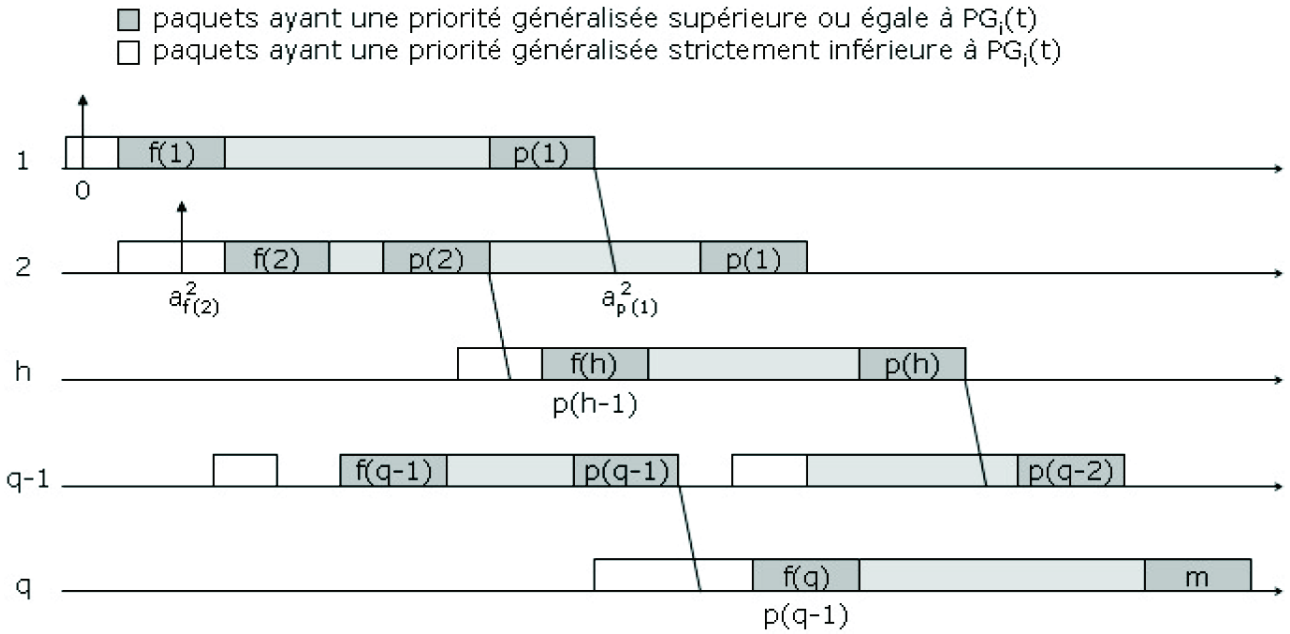


Figure II.9 – Périodes actives de niveau $PG_i(t)$ considérées [99]

— le temps de traitement sur le nœud q des paquets $f(q)$ à $(m-1) - \left(a^q_{p(q-1)} - a^q_{f(q)}\right)$

— $\delta_i^{1,q}(t)$, le délai maximum subi par m dans les nœuds 1 à q à cause de l'effet non-préemptif direct.

Soit finalement :

$$W_i^q(t) = \sum_{h=1}^{q} \left(\sum_{g=f(h)}^{p(h)} C^h_{\tau(g)} \right) - C_i^q + \delta_i^{1,q}(t) + (q-1) \cdot L_{\max} - \sum_{h=2}^{q} \left(a^h_{p(h-1)} - a^h_{f(h)} \right) \qquad \text{(II.7)}$$

4.5 Exemple

Cet exemple, extrait de [13] est basée sur la figure II.7. Tous les nœunds travaillent à la même vitesse :

$$C_i^h = C_i, \forall i \in [1,6], \forall h \in [1,3]$$

Tous les flux émettent des paquets de même longueur et ont le même niveau de priorité, à l'exception du flux τ_6 :

— $C_1 = \cdots = C_5 = 2 \times C_6$;

— $P_1 = \cdots = P_5 > P_6$.

Chaque flux τ_i suit un chemin Pi avec :

— $P1 = \{3\}$;

— $P2 = \{1,2,3\}$;

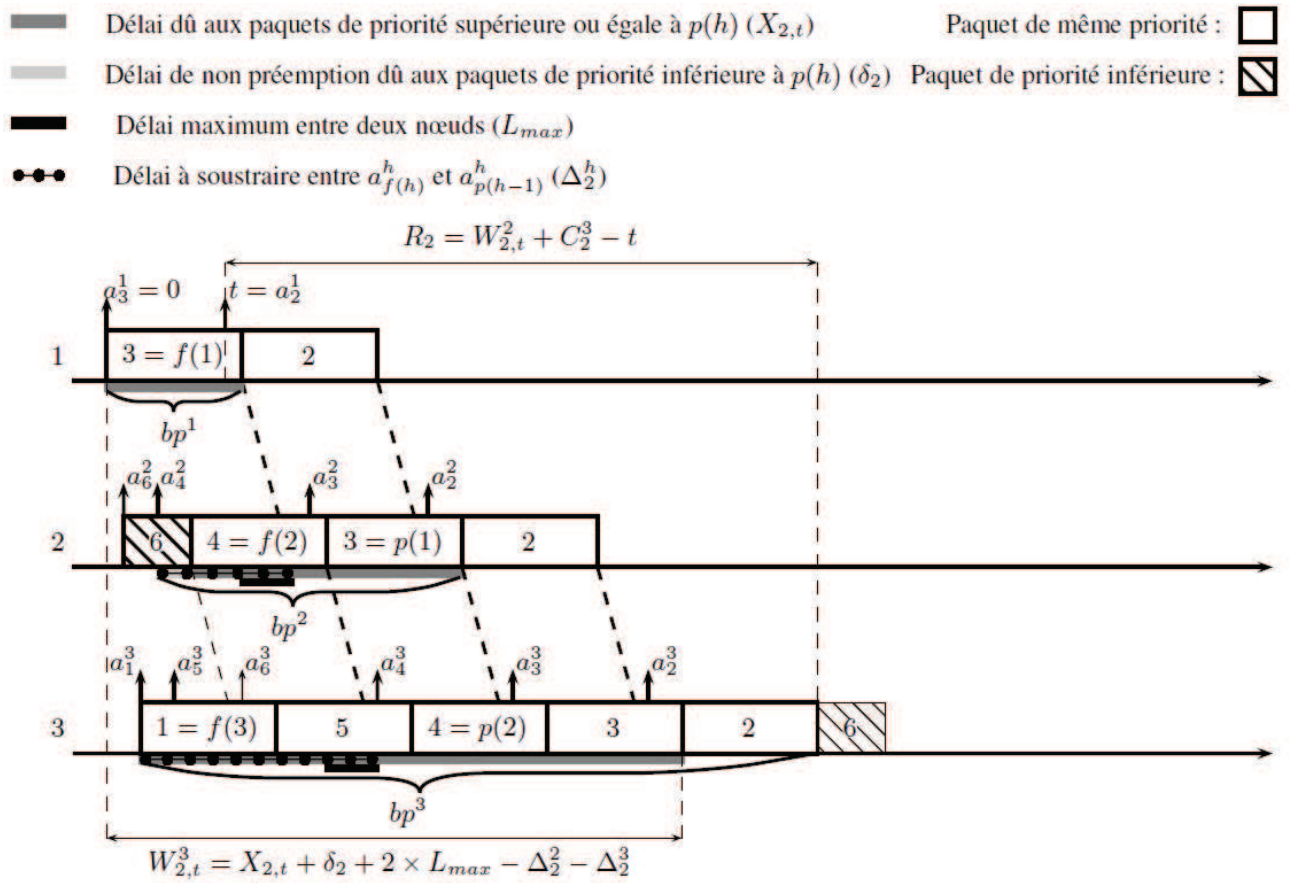


Figure II.10 – Un ordonnancement possible sur la trajectoire du paquet 2 [13]

— $P3 = \{1, 2, 3\}$;
— $P4 = \{2, 3\}$;
— $P5 = \{3\}$;
— $P6 = \{2, 3\}$.

On s'intéresse au délai de bout en bout du flux τ_2 qui traverse successivement les nœuds $1, 2$ et 3. Soit t, l'instant de génération du paquet 2 du flux τ_2 dans son premier nœud : $t = a^1_2$, en notant $a^h_{m'}$ l'instant d'arrivée du paquet m' sur le nœud h. La figure II.10 donne un exemple d'ordonnancement des paquets sur la trajectoire du paquet 2. Soit bp^3 la période active dans laquelle le paquet 2 est traité dans son dernier nœud. Dans cette période active bp^3, le premier paquet dont la priorité est supérieure ou égale à P_2 est appelé $f(3)$, tandis que le premier paquet provenant du même nœud précédent que le paquet 3, appelé $p(2)$. Pour le nœud 3 nous avons donc : $f(3) = 1$ et $p(2) = 4$. La démarche est poursuivie en considérant la période active bp^2 dans laquelle est traitée le paquet $p(2)$. De la même manière, $f(2) = 4$ est le premier paquet de priorité supérieure ou égale à celle de $p(2)$ traité dans la période active bp^2, et $p(1) = 3$ est le

premier paquet provenant du nœud 1. Le calcul par trajectoires se poursuit ainsi jusqu'à arriver au nœud 1 dans lequel est identifié le premier paquet de la période active bp^1 dans laquelle est traité le paquet $p(1)$: dans notre exemple $f(1) = 3$.

Dans la suite : $\sum_{g=f(h)}^{p(h)} C^h_{\tau_g}$ représente la somme des temps de transmission des paquets placés entre $f(h)$ et $p(h)$. Le temps de départ sur son dernier nœud du paquet 2 s'obtient en sommant :

- le temps de traitement des paquets de $f(1)$ à $p(1)$ sur le nœud 1, ie. le temps de traitement du paquet 3 ;
- le délai subi par $p(1)$ entre les nœuds 1 et 2, $\cdots$, $L_{\max}$;
- le délai qui s'écoule entre l'arrivée de $p(1)$ sur le nœud 2 et la fin du traitement de $p(2)$. Ce délai est obtenu en prenant le temps écoulé entre le début de la période active bp^2 (a^2_4 sur l'exemple) et la fin du traitement de $p(2)$ et en lui retranchant le temps écoulé entre a^2_4 et l'arrivée de $p(1)$ sur le nœud 2 (a^2_3). Sur l'exemple, le temps écoulé entre a^2_4 et la fin du traitement de $p(2)$ comprend la fin du traitement du paquet 6 (effet de non préemption car le paquet 6 est moins prioritaire) et le traitement du paquet 4 ;
- le délai subi par $p(2)$ entre les nœuds 2 et 3 ;
- le délai qui s'écoule entre l'arrivée de $p(2)$ sur le nœud 3 et la fin du traitement du paquet 3. Le calcul est effectué de la même manière que sur le nœud 2.

Ce temps de départ $W^3_{2,t}$ du paquet 2, généré à l'instant t, du nœud 3 s'écrit :

$$W^3_{2,t} = X_{2,t} + \sum_{h=1}^{3} \delta^h_2 + (3-1) \times L_{\max} - \sum_{h=2}^{3} \Delta^h_2 \tag{II.8}$$

où :

- $X_{2,t} = \sum_{h=1}^{3} \left(\sum_{g=f(h)}^{p(h)} C^h_{\tau_g} \right) - C^3_2$, correspond à l'ensemble des paquets (de priorité fixe supérieure à celle de m) des périodes actives à comptabiliser (le paquet 3 sur le nœud 1, le paquet 4 sur le nœud 2, les paquets 1, 5, 4 et 3 sur le nœud 3) ;
- $\Delta^h_2 = a^h_{p(h-1)} - a^h_{f(h)}$, correspond à l'avance dans le traitement prise par les paquets qui rejoignent le paquet 2 dans le nœud h, ie. aux portions en début de périodes actives à retrancher ;
- δ^h_2, correspond au délai dû à l'effet de non préemption sur le nœud h ;
- $(3-1) \times L_{\max}$, correspond au délai maximum entre les différents nœuds.

Chacune de ces contributions est représentée par un soulignage différent dans la figure II.10.

Les résultats existants sur les trajectoires font l'hypothèse que dans tous les nœuds, les paquets qui rejoignent le paquet 2 pour la première fois peuvent toujours être retardés de manière à avoir $f(h) = p(h-1)$ pour tout nœud h du chemin P_2. L'ordonnancement correspondant

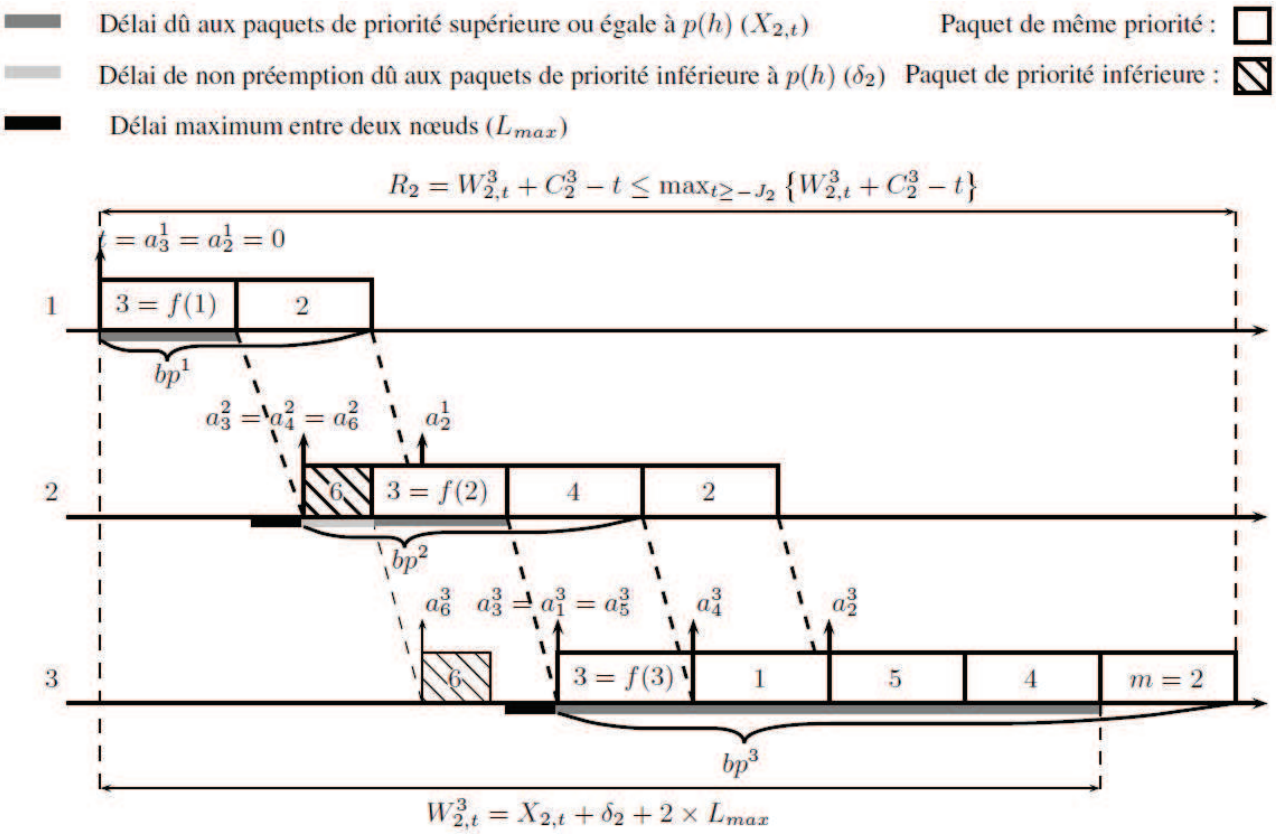


Figure II.11 – Maximisation du retard sur la trajectoire du paquet 2 [13]

à cette hypothèse est donné à la figure II.11. Dans ce cas, pour tout nœud h du chemin P_2, $\Delta_2^h = 0$. Il est montré dans [99] que cet ordonnancement maximise l'instant de départ du paquet étudié dans son dernier nœud.

Les résultats du Lemme 1 de [99] démontrent que dans la quantité $X_{i,t}$ pour un paquet m parcourant q nœuds, seul un paquet est commun à deux périodes actives consécutives. Le paquet commun à la période active bp^h et la période active bp^{h+1} est $f(h+1)$. Dans le pire cas, le paquet compté deux fois est le plus grand de la période active : $\max_{j \in hp_i \cup sp_i \cup \{i\}} C_j^h$.

$X_{i,t}$ comprend donc un premier terme qui comptabilise une fois chaque paquet pouvant retarder m un second terme qui prend en compte qu'un (et un seul) paquet apparaît systématiquement dans deux périodes actives consécutives. Un paquet est toujours compté dans le nœud

sur lequel son temps de traitement est le plus grand. Ainsi :

$$X_{i,t} \leq \underbrace{\sum_{g=f(1)}^{m} C_{\tau(g)}^{slow_{\tau(g),i}} - C_i^q}_{\substack{Paquets\ comptabilisés\ dans \\ une\ seule\ période\ active}} + \underbrace{\sum_{\substack{h=1 \\ h \neq slow_i}}^{q} \max_{j \in hp_i \cup sp_i \cup \{i\}} C_j^h}_{\substack{Paquets\ comptabilisés\ dans \\ deux\ périodes\ actives}}$$

Dans le cas de cet exemple, le terme $X_{2,t}$ est donc majoré par :

$$X_{i,t} \leq \underbrace{C_3^1 + C_1^3 + C_5^3 + C_4^3 + C_2^3 - C_2^3}_{\substack{Paquets\ comptabilisés\ dans \\ une\ seule\ période\ active}} + \underbrace{C_3^3 + C_3^3}_{\substack{Paquets\ comptabilisés\ dans \\ deux\ périodes\ actives}}$$

4.6 Bilan de l'approche

L'approche par trajectoire a fait l'objet de plusieurs publications depuis les travaux fondateurs de S. Martin [99, 100]. Elle a été adaptée à l'AFDX par H. Bauer [15, 16, 17, 13]. Partant en effet des travaux de S. Martin, H. Bauer a utilisé cette approche pour borner le délai de traversée de bout en bout dans une configuration industrielle du réseau AFDX [14]. L'approche a ensuite été enrichie par la technique de groupage introduite dans [59] pour obtenir de meilleurs performances [15, 16, 17, 13].

Un apport important de l'approche par trajectoire est qu'elle a inspiré une approche permettant de construire des scénarios défavorables sur lesquels s'appuyer pour borner le pessimisme des méthodes d'évaluation de borne sur le délai pire cas. D'après cette étude le pessimisme est estimé à environ 10% [32].

L'approche par trajectoire est loin d'être une approche simple.

En effet, une mauvaise compréhension de celle-ci a conduit [93] à affirmer, sur un exemple, que les résultats de cette approche [14, 15] présentent une anomalie pouvant conduire dans certains cas à une sous-approximation des bornes obtenues. [93] introduit alors le calcul récursif, sans prise en charge du groupage, pour corriger ce problème. Dans une correspondance privée, S. Martin et H. Bauer ont souligné des erreurs dans [93], mais aucun document public faisant la confrontation n'est disponible à ce jour à notre connaissance.

Mais surtout, comme nous l'avons dit en introduction, il a été montré dans [73] et [72] que l'approche était dans certains cas, fausse, puisqu'elle donne des bornes supérieures plus petites que le véritable pire cas. Et cette erreur a été trouvée plus de dix ans après les premiers travaux, alors que deux équipes distinctes ont travaillé sur le sujet, et deux thèses soutenues [13, 91].

La méthode est sûrement juste dans certains sous-cas, mais ces sous-cas n'ont pas encore été identifiés.

La méthode des trajectoires a fait l'objet de comparaison avec le calcul réseau [14, 15, 16, 17, 13, 105], surtout dans le cadre du réseau AFDX.

Malgré le fait qu'il est désormais su que la méthode est trop optimiste, nous reportons ici quelque unes de ces comparaisons, qui montre que les approches sont comparables.

Notons que l'approche par trajectoire est une méthode paquetisée dans le sens où elle sait tenir compte des flux arrivant sous forme de paquets. Par contre les résultats de calcul réseau utilisés dans cet exercice de comparaison sont ceux ne s'appliquant que pour un modèle fluide, avec une approximation fluide pour la transmission des paquets. Cette comparaison montre un gain d'environ 10% pour l'approche par trajectoire par rapport au Calcul réseau [14]. Une amélioration de l'approche par trajectoire présentée dans [15] permet d'obtenir en général de bien meilleurs bornes qu'avec le calcul réseau avec modèle fluide. Signalons cependant que cette version non fluide du calcul réseau réussi qu'en même, dans certains cas, à obtenir des bornes 5% meilleures que l'approche par trajectoire améliorée [15].

5 - Event Stream

5.1 Description

La technique repose sur une abstraction simple du mécanisme d'activation et de communication de tâches. Au lieu de considérer chaque événement du système de façon individuelle, cette méthode formelle d'analyse d'ordonnancement fait une abstraction consistant à remplacer les événements individuels par des flots d'événements. L'analyse ne représente que quelques caractéristiques simples des flots d'événements comme la période de l'événement ou sa gigue maximale.

Partant de ces paramètres, le modèle réutilise des analyses d'ordonnancement déjà connues (*"[Event Stream]" don't necessarily need to develop new local analysis techniques if we can benefit from the host of work in real-time scheduling analysis"* [62, § 2]).

5.2 SymTA

SymTA/S Est un outil basé sur la méthode des Event Stream [62]. SymTA/S permet d'obtenir des indicateurs d'évaluation de performances des systèmes comme le délai de bout en bout, le scénario conduisant à l'ordonnancement pire cas, le niveau d'utilisation des bus et

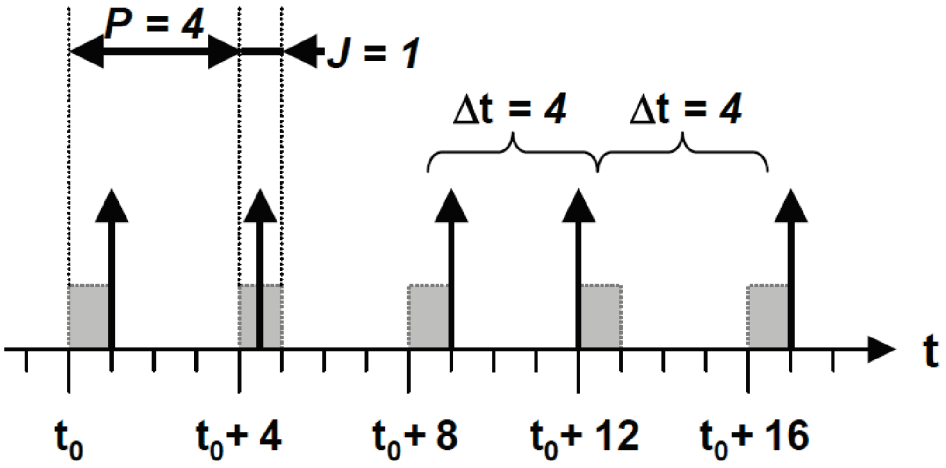


Figure II.12 – Exemple d'un flot d'événements qui respecte le modèle $P = 4, J = 1$ [62]

processeurs.

5.2.1 Modèle applicatif des Event Stream

Le modèle des Event Stream est un modèle de tâches. Chaque tâches doit être associée à une ressource pour que son exécution ait lieu. Lorsque plusieurs tâches partagent la même ressource, il peut arriver que deux d'entre elles, ou même plus, aient besoin de la ressource en même temps. Dans le but d'arbitrer les conflits liés à l'utilisation de ressources partagées, chaque ressource est associée à un ordonnanceur. L'ordonnanceur sélectionne la tâche active à laquelle la ressource est affectée, conformément à une politique d'ordonnancement donnée. Les autres tâches actives doivent alors attendre. L'analyse d'ordonnancement évalue, pour toutes les tâches qui partagent la ressource, le pire temps de réponse, qui est le temps qui s'écoule entre l'activation et la fin d'exécution de la tâche. L'analyse d'ordonnancement garanti alors que tout temps de réponse observé sera borné par le temps de réponse pire cas.

5.2.2 Modèle standard d'événements de SymTA/S

Des modèles d'événements peuvent être décrits par des ensembles de paramètres. Par exemple, un modèle d'événements périodiques avec gigue peut être décrit par deux paramètres (P, J). Ce qui signifie que chaque événement survient périodiquement avec une période P et peut subir une gigue autour de sa position avec un intervalle de gigue de longueur J. Par exemple, pour $(P, J) = (4, 1)$, la figure II.12 permet de visualiser ce modèle d'événements. Chaque zone grise indique un intervalle de gigue $J = 1$ qui se répète avec une période $P = 4$. La figure II.12 montre aussi une séquence d'événements qui satisfont le modèle d'événements.

— Un modèle d'événements peut aussi s'exprimer en utilisant deux fonctions. Une fonction qui spécifie le nombre maximum d'événements pouvant survenir durant un intervalle de temps donné. C'est une fonction borne supérieure du nombre d'événements.

$$\eta^{u}_{P+J}(\triangle t) = \left\lceil \frac{\triangle t + J}{P} \right\rceil$$

— Une fonction qui spécifie le nombre minimum d'événements pouvant survenir durant un intervalle de temps donné. C'est une fonction borne inférieure du nombre d'événements.

$$\eta^{l}_{P+J}(\triangle t) = \max\left(0, \left\lceil \frac{\triangle t - J}{P} \right\rceil\right)$$

Ces modèles permettent de gérer des systèmes relativement simples, précisément ceux qu'on peut abstraire avec la période et la gigue. Pour prendre en compte d'autres réalités comme le burst, le modèle est étendue avec les notion de distance maximale et de distance minimale.

— La fonction de distance minimale $\delta^{min}(N \geq 2)$ spécifie la distance minimale entre $N \geq 2$ événements consécutifs.

— La fonction de distance maximale $\delta^{max}(N \geq 2)$ spécifie la distance minimale entre $N \geq 2$ événements consécutifs.

Pour un modèle périodique à événements avec gigue ces fonctions s'expriment comme suit.

$$\delta^{min}(N \geq 2) = max\{0, (N-1) \times P - J\}$$

$$\delta^{max}(N \geq 2) = (N-1) \times P + J$$

Avec ces définitions, la distance minimale entre deux événements dans un modèle à événements périodique avec gigue ($P = 4, J = 1$) est de 3 unités de temps et la distance maximale est de 5 unités de temps [113]. Le modèle peut aussi être étendu pour gérer les événements sporadiques [112].

Ce modèle est dans sa version basique très semblable, comme nous allons le voir au paragraphe II.6, au calcul réseau dan sa description des flux d'entrée : les courbes η et δ^{min} permettent de borner le nombre d'événements sur un intervalle de temps.

5.2.3 Composition de serveurs

L'analyse de performance d'un système global alterne entre analyse d'ordonnancement local et propagation du modèle à événements. Cette propagation nécessite une modélisation des

événements sortant des composants ordonnançables. Ainsi, partant du modèle d'entrée d'un tel composant, l'obtention du modèle de sortie est une base qui servira pour obtenir le modèle d'entrée du prochain composant ordonnançable. C'est de cette façon que le modèle à événements est propagé de proche en proche.

Par exemple, pour le modèle à événements standard, des règles simples permettent d'obtenir le modèle de sortie. Pour un tel modèle, la période de sortie ne change pas. Elle est la même que celle du modèle d'entrée. Par contre, pour ce qui est de la gigue, la gigue du modèle d'entrée est augmentée par la différence entre le temps de réponse maximal et le temps de réponse minimal pour obtenir la gigue du modèle de sortie.

$$J_{out} = J_{act} + (t_{resp,max} - t_{resp,min})$$

Cette combinaison entre entre analyse d'ordonnancement local et propagation du modèle à événements a besoin de mécanismes plus poussés pour tenir compte de certaines contraintes qui peuvent exister entre les tâches. Des connecteurs appelés activateurs sont prévus pour des situations de conjonction ou de disjonction de tâches. Une extension existe pour gérer les contraintes de dépendances cycliques entre les tâches et les cas où la gigue résultante serait plus grande que la période [70].

5.3 Comparaison avec le calcul réseau

Le modèle d'évènement présenté au paragraphe II.5.2.2 est assez proche, comme nous allons le voir, du modèle de courbe d'arrivée du calcul réseau. Il s'agit, à l'aide des fonctions η^u et δ^{min} de borner le nombre maximal d'événement que peut produire un flux sur un intervalle de largeur Δ. Un événement sera, ensuite, l'activation d'une tâche. Mais ces courbes ayant des paramètres fixes (rafale, période, gigue)

Le calcul réseau, lui, prend une approche un peu plus générale, puisque les courbes d'arrivée, qui permettent de borner la quantité de donnée maximale dans un intervalle Δ, peuvent avoir n'importe quelle forme. Le calcul réseau, lui, prend en compte des quantités de donnée, et non de simples événements. Cela est plus adapté pour le réseau où le temps de traitement associé à un message sera généralement dépendant de sa taille (alors que les événements n'ayant pas de taille, les temps de traitement ne pourront pas être différenciés). Inversement, le calcul réseau a du mal a faire la différence entre deux paquets de taille l et un unique paquet de taille $2l$.

En ce qui concerne le temps de traitement sur un nœud, la méthode "Event Stream" se veut très proche de la théorique classique de l'ordonnancement, et permettre de réutiliser des

méthodes existantes, alors que le calcul réseau définit un modèle spécifique.

Les temps de réponse calculés par les autres méthodes sont alors utilisés comme gigue ou augmentation de la rafale dans le flux d'évènement en sortie.

En conclusion, on pourrait dire que le modèle Event-Stream est une généralisation des modèles classiques de l'ordonnancement, qui permet une réutilisation de résultats existant, alors que le calcul réseau part d'un modèle mathématique beaucoup plus général.

D'ailleurs, comparant ses travaux avec d'autres plus généraux, ils écrivent *"This generality, however, has its price. Because they introduce new stream models, both Thiele and Gresser had to develop new scheduling analysis algorithms for the local components that utilize these models ; the host of existing work in real-time system can not be re-used. Furthermore, the new models are far less intuitive than the ones known from the classical real-time systems research, e. g. the model of rate-monotonic scheduling with its periodic tasks and worst-case execution times. A system-level analysis should be simple and comprehensible, otherwise its acceptance is extremely doubtful."* [62]

Mais cette simplicité se perd un peu dès que de nouveaux comportements doivent être pris en compte. Ainsi, lorsqu'il s'agit d'introduire des notions non présentes dans le modèle initial (comme le "groupage" qui signifie que le passage par un lien de communication sérialise les paquets, ce qui n'existe pas dans un modèle à mémoire partagée), cela peut demander des extensions du modèle dans un cas [114], alors que dans un cadre général, cela s'introduit beaucoup plus facilement [59].

6 - Le calcul réseau

6.1 Introduction

Le calcul réseau est une théorie, de systèmes de files d'attente déterministes, rencontrée dans les réseaux de communications. Il est basé sur l'algèbre $(min, +)$. Plus qu'un simple formalisme, il permet d'analyser les systèmes complexes et de prouver des bornes déterministes sur les delays et backlogs. L'analyse s'intéresse aux performances pire cas. Les informations sur les fonctionnalités du système sont stockées dans les fonctions, telle que les courbes d'arrivée qui caractérisent le trafic entrant dans un système, ou les courbes de service qui permettent de quantifier le service garanti au niveau d'un nœud. Ces fonctions peuvent aussi être combinées ensemble grâce aux opérateurs du calcul réseau pour obtenir des bornes sur la taille des files d'attente.

6.1.1 Historique

Le calcul réseau est né des travaux fondamentaux de Cruz [49, 50]. Dans ces travaux, le contexte est celui des réseaux de communication, avec commutation des paquets selon un routage statique. Toute donnée qui est émise sur le réseau par un utilisateur doit être sous forme de paquet. Ces paquets parcourent alors le réseau suivant un itinéraire prédéfini, jusqu'à atteindre leur destination. La grande particularité de ces travaux réside dans le fait que flux en entrée du réseau doivent satisfaire certaines contraintes de régularité.

À l'heure actuelle, la théorie s'est développée et a conduit à des résultats aboutis dont l'essentiel se trouve dans deux ouvrages de référence : livre de Chang [41] et celui de Le Boudec et Thiran [23].

6.1.2 Équivalence NC RTC

Le Real Time Calculus (RTC) est présentée comme une alternative au calcul réseau. Le calcul réseau et le RTC partagent les mêmes fondements à savoir l'utilisation d'enveloppes de mise en forme des courbes cumulatives de trafic et de services. Comparativement au calcul réseau, le RTC affirme présenter des modèles plus expressifs. Une bonne comparaison entre NC et RTC se trouve dans [26].

Cette comparaison relève les différences entre les deux approches. L'espace des fonctions manipulées parait plus large avec le RTC (définition des fonctions sur R, au lieu de R_+). Le RTC manipule des fonctions à deux variables, contrairement au NC qui utilise les fonctions à une variable ; cependant, de l'utilisation qui en est faite de ces fonctions à deux variables, on constate une équivalence entre les deux notations. La différence notoire entre le RTC et le NC est que le RTC utilise deux fonctions, une minimale et une maximale, pour chacun des concepts de courbe d'arrivée et courbe de service. En calcul réseau, la notion de service maximal et de trafic minimal ont aussi été définis, mais elles ne sont pas systématiquement étudiées. Cela est sûrement du à des domaines d'application différents. Par exemple, dans ce mémoire, nous traitons des systèmes embarqués. Il peut alors arriver qu'une fonction soit inactive, ou en panne et n'émette rien. Elle n'a alors pas de débit minimal. Une autre différence très subtile entre le NC et le RTC est la nature du service. Celle-ci peut être simple, strict ou faiblement stricte en NC. Le RTC ne défini que le service strict.

En somme, comme il est montré dans [26] à la lumière de [132, 131], NC et RTC sont formellement équivalents.

6.2 Fondements mathématiques

6.2.1 L'algèbre min-plus

L'algèbre min-plus se défini souvent par analogie avec l'algèbre classique dans laquelle la structure algébrique $(R, +, \times)$ est remplacée par $(R \cup +\infty, \wedge, +)$. L'opération d'addition est remplacée par le minimum, noté $\wedge$ tandis que l'opération de multiplication est remplacée par l'addition classique.

L'algèbre min-plus vérifie les propriétés suivantes :

— (Clôture de $\wedge$) $\forall a, b \in R \cup +\infty, a \wedge b \in R \cup +\infty$.

— (Associativité de $\wedge$) $\forall a, b, c \in R \cup +\infty, (a \wedge b) \wedge c = a \wedge (b \wedge c)$.

— (Existence de l'élément neutre pour $\wedge$) Il existe $e = +\infty \in R \cup +\infty$ tel que $\forall a \in R \cup +\infty, a \wedge e = a$.

— (Idempotence $\wedge$) $\forall a \in R \cup +\infty, a \wedge a = a$.

— (Commutativité de $\wedge$) $\forall a, b \in R \cup +\infty, a \wedge b = b \wedge a$.

— (Clôture de $+$) $\forall a, b \in R \cup +\infty, a + b \in R \cup +\infty$.

— (Associativité de $+$) $\forall a, b, c \in R \cup +\infty, (a + b) + c = a + (b + c)$.

— (L'élément neutre de $\wedge$ est absorbant pour $+$) $\forall a \in R \cup +\infty, a + e = e = e + a$.

6.2.2 Les principaux opérateurs

Nous présentons ici quelques opérateurs du calcul réseau utilisés dans la suite. Au-delà des opérations habituelles comme le minimum ou l'addition des fonctions, le calcul réseau utilise d'autres opérations classiques qui sont une translation des opérations usuelles $(+, \times)$ dans le dioïde $(\min, +)$. Le calcul réseau travaille généralement sur des fonctions non-décroissantes. Les fonctions suivantes sont utilisées dans cette thèse.

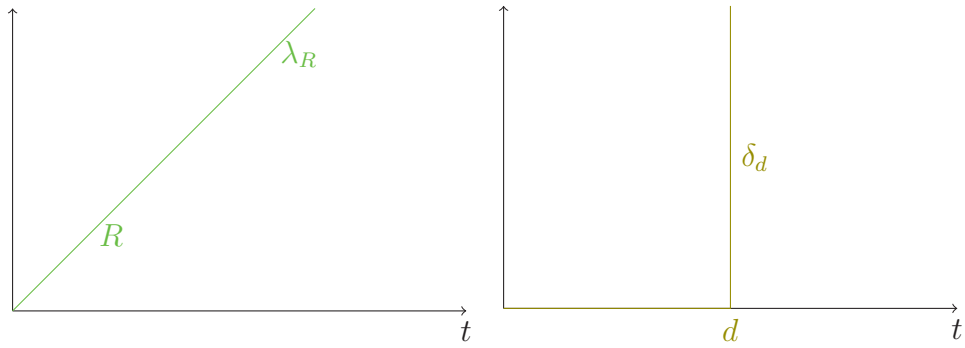
Ensemble $\mathcal{F}$ $\mathcal{F}$ désigne l'ensemble des fonctions non décroissantes $f : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ tel que $f(t) = 0$ for $t < 0$.

Fonction $[\ ]^+ : x \mapsto \max(x, 0)$.

Déviation verticale Elle se définie pour deux fonctions f et g par
$v(f, g) = \sup_{t \geq 0} \{f(t) - g(t)\}$

Déviation horizontale Elle est définie pour deux fonctions f et g par
$h(f, g) = \sup_{t \geq 0} \{\inf \{d \geq 0 \mid f(t) \leq g(t + d)\}\}$

Min-plus convolution Elle se définie pour deux fonctions f et g par
$(f \otimes g)(t) = \inf_{0 \leq s \leq t} \{f(t - s) + g(s)\}$



(a) La fonction Peak Rate **(b)** La fonction Burst Delay

Figure II.13 – Fonctions usuelles du NC.

Min-plus déconvolution Elle se définie pour deux fonctions f et g par
$(f \oslash g)(t) = \sup_{u \geq 0} \{f(t+u) - g(u)\}$

Clôture positive non décroissante Elle se définie comme suit pour une fonction f :
$f \uparrow (t) = [\sup_{0 \leq s \leq t} f(s)]^+$

Pseudo inverse L'inverse, f^{-1}, d'une fonction $f \in \mathcal{F}$ n'est pas toujours définie, cependant, deux pseudo-inverses peuvent être définies [109] :

$$f^{-1}_{\inf}(u) \stackrel{\text{def}}{=} \inf\{t | f(t) \geq u\} \qquad \stackrel{\text{ppty}}{=} \sup\{t \mid f(t) < u\}$$
$$f^{-1}_{sup}(u) \stackrel{\text{def}}{=} \sup\{t | f(t) \leq u\} \qquad \stackrel{\text{ppty}}{=} \inf\{t \mid f(t) > u\}$$

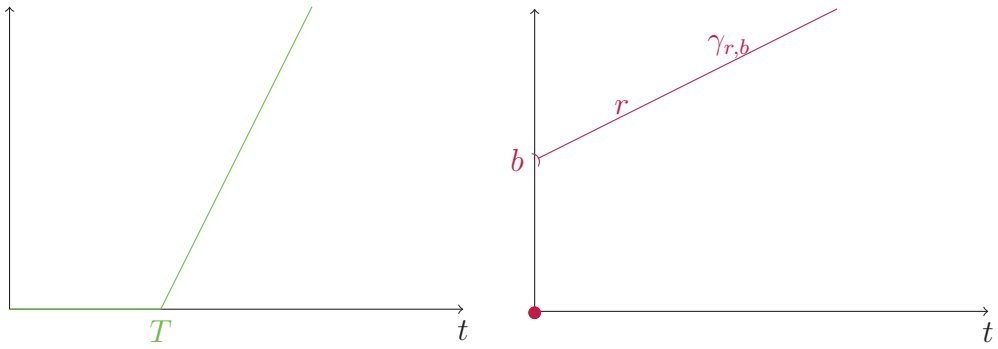
6.2.3 Les principales fonctions

Nous présentons ici un ensemble de fonctions qui permettent de décrire des courbes d'arrivée et des courbes de service. On peut aussi retrouver ces fonctions sous forme de combinaisons de plusieurs d'entre elles à l'aide des opérateurs du calcul réseau.

La fonction Peak Rate λ_R Elle est définie par :

$$\lambda_R(t) = \begin{cases} Rt \ si \ t > 0 \\ 0 \ sinon \end{cases}$$

Elle est utilisée pour représenter le service offert par un serveur émettant à une vitesse constante R (figure II.13a).



(a) La fonction Rate-Latency **(b)** La fonction Affine

Figure II.14 – Fonctions usuelles du NC.

La fonction Burst Delay δ_R Elle est définie par :

$$\delta_R(t) = \begin{cases} +\infty \ si \ t > R \\ 0 \ sinon \end{cases}$$

Elle est utilisée pour représenter le service offert par un serveur fournissant un délai borné (figure II.13b).

La fonction Rate-Latency $\beta_{R,T}$

$$\beta_{R,T}(t) = R\,[t-T]^+ = \begin{cases} R\,(t-T) \ \ si \ t > T \\ 0 \ sinon \end{cases}$$

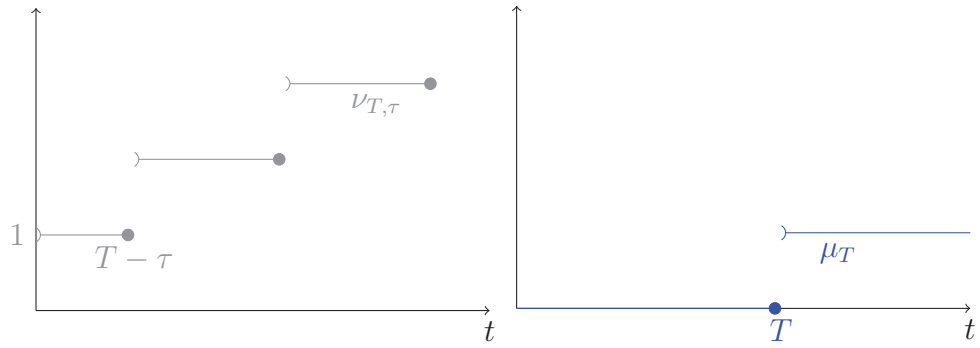
La fonction Rate-Latency convient pour décrire le comportement d'un serveur émettant à une vitesse constante R et pouvant observer une période d'inactivité d'une durée maximale de T (figure II.14a).

La fonction Affine $\gamma_{r,b}$

$$\gamma_{r,b}(t) = \begin{cases} rt + b \ si \ t > 0 \\ 0 \ sinon \end{cases}$$

Cette fonction convient pour décrire le comportement d'un serveur émettant à une vitesse constante r et pouvant, de façon sporadique émettre une grande quantité de données (burst b) comme le montre la figure II.14b.

La fonction Staircase ν_T Elle se définie par la relation suivante.



(a) La fonction Staircase **(b)** La fonction Step

Figure II.15 – Fonctions usuelles du NC

$$\nu_{T,\tau}(t) = \begin{cases} \left\lceil \frac{t+\tau}{T} \right\rceil \ si\ t > 0 \\ 0\ sinon \end{cases}$$

La fonction Staircase convient pour un serveur émettant des paquets de message en fixant le temps minimum d'émission entre deux paquets consécutifs (T), avec une gigue τ comme le montre la figure II.15a.

La fonction Step μ_T

$$\mu_T(t) = 1_{\{t>T\}} = \begin{cases} 1\ si\ t > T \\ 0\ sinon \end{cases}$$

La fonction Step permet de décrie le comportement d'un serveur émettant un unique message (figure II.15b).

6.2.4 Quelques exemples

La convolution de deux fonctions Rate-Latency donne le résultat suivant.

$$\beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{min(R_1,R_2),T_1+T_2}$$

La déconvolution d'une fonction affine par une fonction Rate-Latency donne la fonction suivante (figure II.16).

$$\gamma_{r,b} \oslash \beta_{R,T}(t) = \begin{cases} [b + R(t+T)]^+ \ si\ t \leq -T \\ b + r(t+T)\ sinon \end{cases}$$

La convolution d'une fonction affine par une fonction Rate-Latency donne la fonction sui-

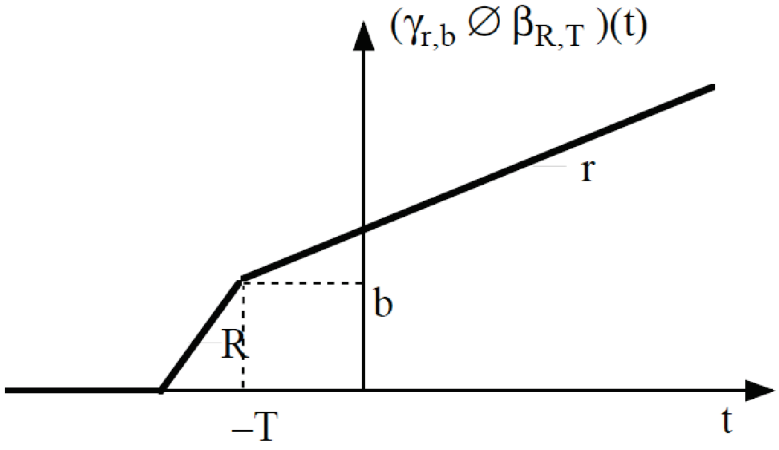


Figure II.16 – Fonction $\gamma_{r,b} \oslash \beta_{R,T}$ pour $(0 < r < R)$

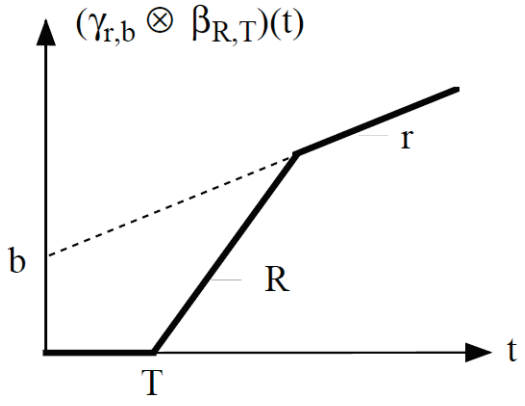


Figure II.17 – La fonction $\gamma_{r,b} \otimes \beta_{R,T}$ $(0 < r < R)$

vante (figure II.17).

$$\gamma_{r,b} \otimes \beta_{R,T}(t) = \begin{cases} 0 \ si \ 0 \leq t \leq T \\ \{b + r(t-T)\} \wedge \{R(t-T)\} \ \ si \ t > T \end{cases}$$

6.3 Modélisation d'un réseau de communication

Un modèle de calcul réseau pour un réseau de communication se compose des trois composants suivants :

1. Une partition du réseau en sous-systèmes (souvent appelés nœuds) qui peuvent avoir différentes échelles (du matériel élémentaire comme un processeur à un sous-réseaux de grande taille).

2. Une description des flux de données, où chaque flux suit un trajet à travers une séquence spécifiée de sous-systèmes et où chaque flux est mis en forme par une courbe d'arrivée juste avant d'entrer dans le réseau.

3. Une description du comportement de chaque sous-système, Précisément, il s'agit des courbes de services permettant de décrire les performances de chaque sous-système, ainsi que des politiques de service en cas de multiplexage (plusieurs flux entrant dans un même sous-système et par conséquent partageant son service).

Les flux qui circulent dans le réseau sont modélisés en calcul réseau par des fonctions cumulatives. $R \in \mathcal{F} : R(t)$ comptabilise la quantité totale de données produite par le flux jusqu'à la date t. Les serveurs ne sont que des relations entre flux entrant et flux sortant ($S \in \mathcal{F} \times \mathcal{F}$). Alors $(R, R') \in S$, notée R $R \xrightarrow{S} R'$, signifie qu'un serveur S reçoit un flux d'entrée, $R(t)$, et fournit en sortie des données après un délai variable. Nous avons relation $R' \leq R$, ce qui signifie que les données sortent du serveur après y être entrées. Le système S pourrait être, par exemple, un seul tampon acheminant les données à une fréquence constante, un nœud de communication complexe, ou même d'un réseau complet. La Figure II.20 montre, à gauche, un serveur doté d'une seule file d'attente, et à droite, une agrégation de deux flux.

Pour fournir des garanties aux flux de données, des contrats de trafic sur les flux et les services du réseau sont nécessaires. A cet effet, le calcul réseau fournit les concepts de courbe d'arrivée et de courbe de service.

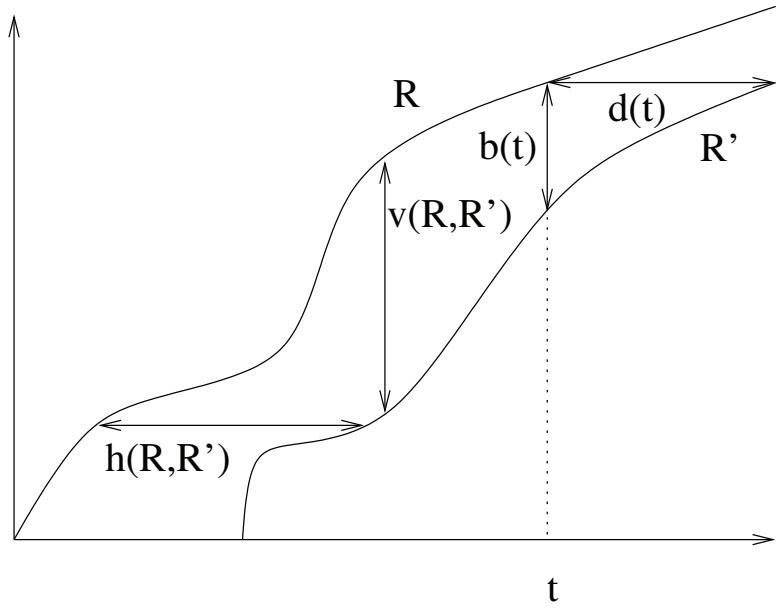


Figure II.18 – Flux d'entrée et de sortie d'un serveur, et indicateur de performance : backlog et délai

6.4 Indicateurs de performances d'un réseau : Backlog et délai

Les principaux indicateurs de performance mesurés par le calcul réseau sont le backlog (taille de la file d'attente) et le délai (subi relatif à la traversée d'un nœud) (Figure II.18).

Définition 2 (Délai virtuel)**.** *Pour un système dont R est l'entrée et R' la sortie,* le délai virtuel *à l'instant t est*

$$d(t) = \inf \{d \geq 0 \mid f(t) \leq g(t+d)\} \tag{II.9}$$

Cette définition du délai signifie que $d(t)$ est le temps qu'il faudra attendre pour que le la quantité de flux donnée en sortie rejoigne la quantité en entrée.

Définition 3 (Backlog)**.** *Pour un système dont R est l'entrée et R' la sortie,* le backlog *à l'instant t est*

$$b(t) = R(t) - R'(t) \tag{II.10}$$

Remarquons que $b(t) \leq v(R, R')$. Le *backlog* est la quantité de données se trouvant dans le système ; si le système est un simple buffer, alors le backlog se réduit à la taille de sa file d'attente. Par contre, si le système est plus complexe, le backlog est le nombre de bits en transit dans le système. On suppose alors ici que l'entrée et la sortie du système peuvent être observées simultanément [23].

Définition 4 (Période de backlog)**.** *Une* période de backlog *est une période durant laquelle le backlog n'est pas nul ($R'(t) < R(t)$).*

Définition 5 (Début de période de backlog)**.** *Soit t, un moment d'une période de backlog, cette période de backlog a débuté à un instant*

$$StBl(t) = \sup\{u \leq t \mid R'(u) = R(u)\} \tag{II.11}$$

Dans la suite, nous nous limitons aux fonctions continues à gauche pour nous assurer que :

$$R'(StBl(t)) = R(StBl(t)) \tag{II.12}$$

Afin de fournie une garantie aux flux de données, il est nécessaire de connaître certaines informations sur les contrats auxquels sont soumis les les flux et les politiques de service fourni pas le réseau. Pour ce besoin, le calcul réseau propose les concepts de courbe d'arrivée et de courbe de service

6.5 Courbe d'arrivée

6.5.1 Définition

Définition 6 (Courbe d'arrivée)**.** *Un flux $R \in \mathcal{F}$* est contraint par *$\alpha \in \mathcal{F}$ si et seulement si pour tout $s \leq t$:*

$$R(t) - R(s) \leq \alpha(t - s) \tag{II.13}$$

On dit aussi que R possède *α* comme courbe d'arrivée*, ou que R* est *α*-contraint*. Cette condition est l'équivalent de $R \leq R \otimes \alpha$.*

Pour un flux R, il existe une infinité de courbes d'arrivée possibles. Il existe des critères de choix de « bonnes » courbes d'arrivée.

6.5.2 Sous-additivité et courbe d'arrivée

Il existe une relation particulière entre les courbes d'arrivée et les fonctions sous-additives de l'algèbre min-plus.

Définition 7 (Fonction sous-additive)**.** *Soit $f \in F$ une fonction, f est sous-additive si et seulement si pour tout $s, t \geq 0$,*

$$f(t + s) \leq f(t) + f(s)$$

Remarquons que cette définition revient à imposer $f \leq f \otimes f$, ou tout simplement $f = f \otimes f$ si $f(0) = 0$. À titre d'exemple, les fonctions concaves passant par l'origine sont des fonctions

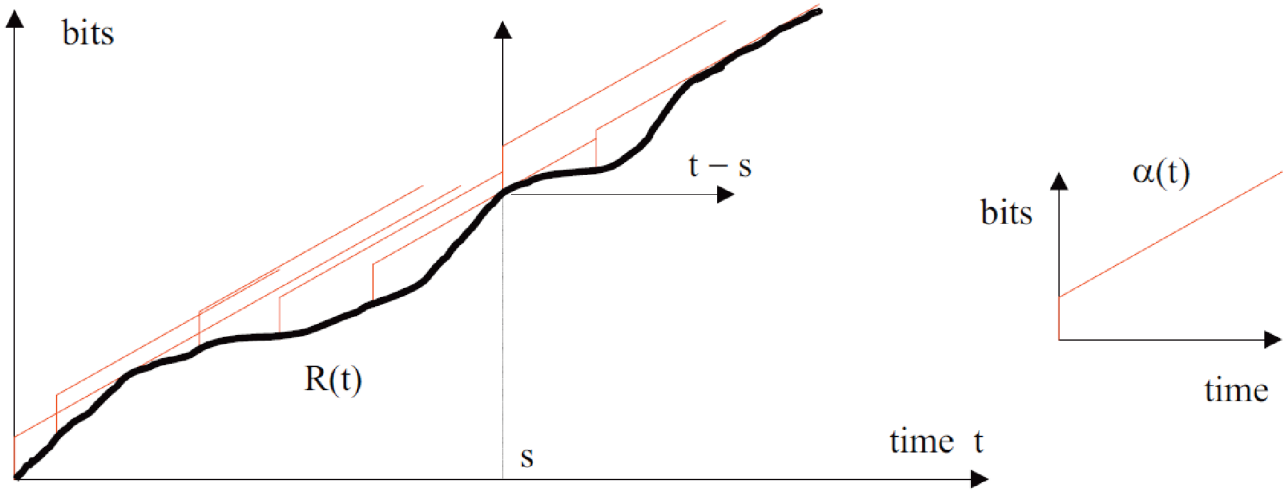


Figure II.19 – La fonction R(t) est contrainte par la courbe d'arrivée $\alpha(t)$: Source [23]

sous-additives. Ainsi, les fonctions affine, peak rate, staircase sont des fonctions sous-additives. Les fonctions sous-additives ne sont pas nécessairement des fonctions concaves.

Définition 8 (Bonne fonction d'arrivée)**.** *Étant donnée α, une fonction dans F, α est une bonne fonction si une des propriétés équivalentes suivantes est vérifiée*

- *α est sous-additive et $\alpha(0) = 0$*
- *$\alpha = \alpha \otimes \alpha$*
- *$\alpha = \alpha \oslash \alpha$*
- *$\alpha = \overline{\alpha}$ (clôture sous-additive de α)*

Définition 9 (La clôture sous-additive)**.** *Soit f, une fonction de F, notons $f^{(n)}$, la fonction obtenue en répétant $(n-1)$ convolutions de f par elle même. Par convention, $f^{(0)} = \delta_0$, de sorte que $f^{(1)} = f$, $f^{(2)} = f \otimes f$, etc. La clôture sous-additive de f, notée $\overline{f}$, est définie par*

$$\overline{f} = \delta_0 \wedge f \wedge (f \otimes f) \wedge (f \otimes f \otimes f) \wedge \cdots = \inf_{n \geq 0} \left\{ f^{(n)} \right\}$$

Un résultat important concernant les courbes d'arrivée se trouve dans le théorème suivant

Théorème 1. *Si α est une courbe d'arrivée pour un flux R, alors sa clôture sous-additive $\alpha = \overline{\alpha}$ l'est aussi.*

6.6 Courbes de service

Le comportement d'un serveur est modélisé par le concept de courbe de service. La courbe de service modélise la garantie de service offert à un flux.

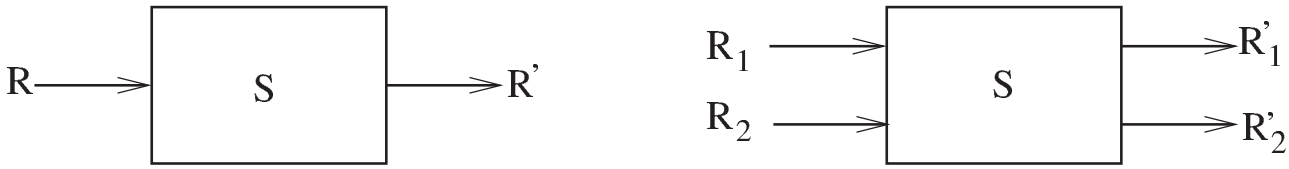


Figure II.20 – Agrégation.

La littérature propose différentes définitions de courbe de service en fonction de la qualité du service offert. [26] propose une étude comparative.

6.6.1 Définitions des différentes notions de courbe de service

Considérons un système $R \xrightarrow{S} R'$, c.-à-d. un serveur S qui reçoit en entrée le flux R et délivre en sortie le flux R' (Figure II.20).

Définition 10 (Courbe de service simple)**.** *Alors S offre au flux R un* service simple de courbe *β si et seulement si.*

$$R' \geq R \otimes \beta \tag{II.14}$$

Définition 11 (Courbe de service strict)**.** *On dit que S offre un* service strict de courbe *β au flux R si et seulement si, durant une période de backlog $[t, s[$, on a :*

$$R'(s) - R'(t) \geq \beta(s - t) \tag{II.15}$$

Définition 12 (Courbe de service faiblement strict)**.** *On dit qu'un système S offre un* service faiblement strict de courbe *β au flux R si et seulement si, $\forall t \geq 0$,*

$$R'(t) - R'(StBl(t)) \geq \beta(t - StBl(t)) \tag{II.16}$$

Notons que si t n'est pas un moment d'une période de backlog alors $StBl(t) = t$. Il existe une hiérarchie entre ces différentes notions de service. En effet, un serveur offrant un service strict offre aussi un service faiblement strict et un serveur qui offre un service faiblement strict offre aussi un service simple. L'intérêt de ces différents types de services réside dans la décomposition du service résiduel.

6.7 Les résultats fondamentaux du calcul réseau : Backlog et borne sur le délai

Le résultat suivant constitue le résultat principal du calcul réseau :

Théorème 2. *Considérons un flux, contraint par une courbe d'arrivée α, traversant un système qui offre un service β, alors le backlog $b(t)$ pour tout t satisfait : $b(t) \leq v(\alpha, \beta)$;*
le délai virtuel $d(t)$ pour tout t satisfait : $d(t) \leq h(\alpha, \beta)$.

6.8 Courbe de service résiduel

En général, les serveurs ne sont pas utilisés pour transférer un flux unique de données, mais un ensemble de flux. La définition du serveur doit être généralisée à celle de serveur à entrées/sorties multiples : $S \in \mathcal{F}^n \times \mathcal{F}^n$, $(R_1, \ldots, R_n) \xrightarrow{S} (R'_1, \ldots, R'_n)$. La capacité du serveur est alors partagée entre plusieurs flux.

La modélisation de l'agrégation et du service résiduel est une question importante en calcul réseau. Agrégation signifie que le service est partagé par différents flux : par exemple, si un serveur S offre un service simple β à deux flux R_1 et R_2, cela signifie qu'il offre ce service au flux $R = R_1 + R_2$ (*i.e.* $R'_1 + R'_2 \geq (R_1 + R_2) \otimes \beta$) mais la répartition du service entre les flux dépend de la priorité des flux et de la politique de service du serveur (comme politique généralement rencontrée, il y a la politique FIFO [88], la politique à base de priorité statique [61, 26], la politique P-GPS/WFQ/DRR [106]) Il est important de bien modéliser l'agrégation en calcul réseau. L'agrégation signifie que le service est partagé par différents flux : par exemple, si un serveur S offre un service simple de courbe β à deux flux R_1 et R_2 ($(R_1, R_2) \xrightarrow{S} (R'_1, R'_2)$, Figure II.20). Lorsqu'on traite de la décomposition d'un service en services résiduels, le type de service est très important : certains résultats supposent un service agrégé strict, d'autres un service agrégé simple. De même, le service résiduel peut aussi être simple ou strict. Avec plus de deux flux agrégés, la nature du service résiduel est cruciale dans la mesure où on peut avoir un service résiduel qui à son tour est partagé par d'autres flux.

Les résultats, qui sont présentés dans la suite, dépendent de la politique de service (blind, FIFO, Priorité statique) et de la nature même du service partagé (simple ou strict).

6.8.1 Politique Blind

Théorème 3 (Politique Blind)**.** *Considérons un nœud servant deux flux, R_1 et R_2. On suppose qu'aucune information n'est disponible sur la politique de gestion de priorité entre les deux flux. Supposons que le nœud garantisse à l'ensemble des deux flux agrégés un service strict de courbe β. Supposons aussi que le flux R_2 est α_2-contraint. Alors la courbe β_1 définie par*

$$\beta_1(t) = [\beta(t) - \alpha_2(t)] \uparrow [26] \tag{II.17}$$

est une courbe de service pour le flux R_1.

6.8.2 Politique FIFO

Théorème 4 (Politique FIFO)**.** *Considérons un nœud servant deux flux, R_1 et R_2 selon la politique de priorité FIFO. Supposons que le nœud garantisse à l'ensemble des deux flux agrégés un service simple de courbe β. Supposons aussi que le flux R_2 est α_2-contraint. Étant donné la définition de la famille de fonctions β_θ^1 définie par*

$$\beta_\theta^1(t) = [\beta(t) - \alpha_2(t-\theta)]^+ 1_{\{t>\theta\}} \tag{II.18}$$

β_θ^1 est une courbe de service pour le flux R_1, [23].

6.8.3 Politique à priorité statique

Des résultats pour cette politiques existent pour un service simple et pour un service strict.

Service résiduel simple Il est montré dans les résultats de [23, Cor. 6.2.1] le résultat suivant.

Théorème 5. *Si un serveur offre un service strict de courbe β à deux flux R_1 et R_2, et si R_1 est contraint par une courbe d'arrivé α_1, alors le flux le moins prioritaire R_2 reçoit un service résiduel simple de courbe $[\beta - \alpha_1]^+$.*

Ce résultat comporte deux limitations : le service résiduel obtenu n'est pas strict (un service résiduel strict peut être décomposé à son tour) et est limité aux fonctions non décroissantes (cette restriction a pour conséquence l'exclusion de certains cas réels en pratique).

Service résiduel strict Les deux problèmes soulignés plus haut ont été étudiés indépendamment par [26, 61] avec le même résultat [1].

Théorème 6. *Considérons que n flux $R_1, \ldots, R_n$, contraints chacun par une courbe d'arrivée α_i et possédant chacun des paquets de taille maximale l_i^{max}, traversent un serveur qui leurs offre un service strict de courbe β. Alors chaque flux R_i bénéficie d'un service simple de courbe β_i et un service stricte de courbe β_i', toutes les deux définies respectivement par :*

1. A priori, on pourrait penser que ces deux résultats sont différents dans la mesure où l'un utilise le calcul réseau et l'autre le RTC (real-time calculus). Cependant, comme il est montré dans [26], la notion de courbe de service minimal du RTC est équivalente à la notion de courbe de service strict du calcul réseau

Flux i	Période	Taille de paquet (l_i)
R_1	3	1
R_2	9	3
R_3	4	1

Tableau II.1 – Paramètre de l'exemple II.6.9

$$\beta_i = (\beta - \sum_{j=1}^{i-1} \alpha_j - \max_{i<j<n} l_j^{max}) \uparrow \tag{II.19}$$

$$\beta'_i = (\beta - \sum_{j=1}^{i-1} \alpha_j - \max_{i\leq j<n} l_j^{max}) \uparrow \tag{II.20}$$

Il est intéressant de noter que la seule différence entre les deux est que dans le cas du service résiduel strict, le flux R_i est en compétition avec sa propre taille de paquet (le terme $-l_i^{\max}$), ce qui n'est pas le cas avec le service simple.

Il est montré dans [26] que ce terme que nous avons nommé "self competitive term" ne peut pas être ignoré dans le cas de service résiduel strict [2].

Les résultats ne sont pas exactement présentés comme ici : [26, Cor. 2] présente ses résultats sans utiliser la fermeture supérieure non décroissante, ce qui peut être ajouté dans le cas du service strict en tenant compte du résultat de [26, Prop. 2].

L'équation (II.19), c.-à-d. sans le "self competitive term", est présentée dans [61]. Mais il ne précise pas si ce service est simple ou strict. Le RTC utilise implicitement le service strict. Avec cette considération, soit le résultat de [61] est faux soit il y a juste une erreur dans la formule.

6.9 Exemple d'application

Description Pour illustrer, considérons un serveur offrant un service strict de courbe $\beta(t) = t$ à trois flux R_1, R_2 et R_3. chaque flux R_i est α_i contraint, et possède des paquets de taille fixe l_i, comme on peut le voir avec le Tableau III.2.

2. Cette notion de "compétition contre soi-même" est contre-intuitive. Sinon, comment comprendre que plus un flux a des paquets de grande taille moins il aura du service avec la non préemption ? Par exemple, avec deux niveaux de priorité, $\beta_2 = (\beta - \alpha_1) \uparrow$ et $\beta'_2 = (\beta - \alpha_1 - l_2^{max}) \uparrow$. Considérons une période de backlog du flux le moins prioritaire R_2. Avant d'être servi, le flux R_2 attend la fin du service du flux le plus prioritaire R_1 (le terme α_1). Cependant, à cause de la non préemption, le flux le plus prioritaire peut avoir été bloqué par un paquet en cours de transmission du flux le moins prioritaire. Si cette situation s'est produite, elle n'a pu se réaliser que dans une période de backlog *précédente*, et non dans la période de backlog considérée comme on peut le voir sur la Figure III.2 . Ce point constitue la différence fondamentale entre le service strict (qui considère une période de backlog *quelconque*) et le service simple basé sur la convolution (ce qui revient à considérer une période de backlog *particulière*)

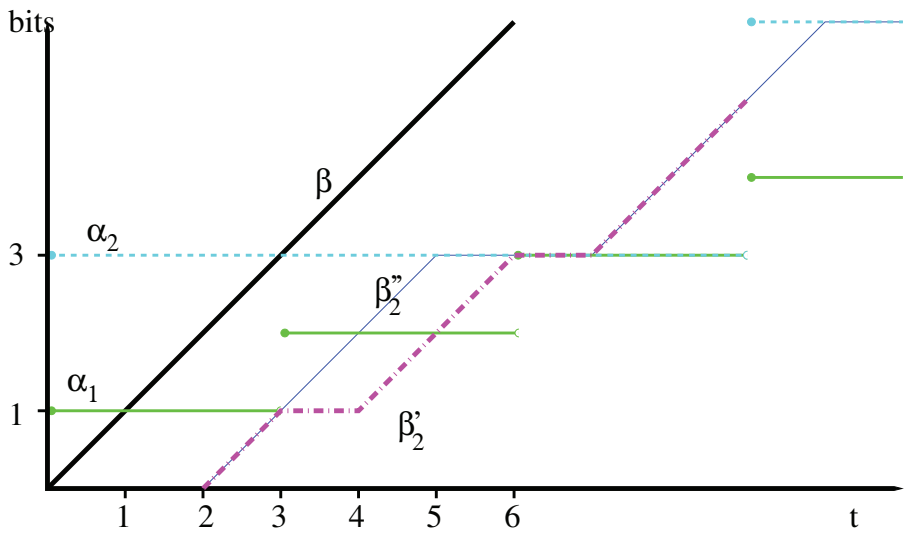


Figure II.21 – Comparaison entre [26, 61], de courbe β'_2 (eq. II.20) et [42] de courbe β"$_2$ (eq. II.21)

La Figure II.21 présente le service résiduel du flux R_2 avec les résultats de [26, 61], eq. II.20 (β'_2 sur la figure).

6.9.1 Modélisation

De la description du tableau III.7 une modélisation respectant le formalisme du calcul réseau s'impose. Pour chaque flux, il convient de déterminer la meilleure courbe d'arrivée qu'elle respecte. Cas du premier flux : R_1 Ce flux génère au plus un paquet de taille une unité toutes les trois unités de temps.

Par conséquent sa courbe d'arrivée, α_1 est décrite par l'expression suivante.

$$\alpha_1(t) = \left\lceil \frac{t}{3} \right\rceil$$

Cas du deuxième flux : R_2 Ce flux génère au plus un paquet de taille trois unité toutes les neufs unités de temps. Par conséquent sa courbe d'arrivée, α_2 est décrite par l'expression suivante.

$$\alpha_2(t) = 3 \left\lceil \frac{t}{9} \right\rceil$$

Cas du flux le moins prioritaire : R_3 Ce flux génère au plus un paquet de taille une unité toutes les quatre unités de temps. Par conséquent sa courbe d'arrivée, α_3 est décrite par l'expression suivante.

$$\alpha_3(t) = \left\lceil \frac{t}{4} \right\rceil$$

Délai virtuel (ms)	R_2
$h(\alpha_2, \beta'_2)$	6
$h(\alpha_2, \beta"_2)$	5

Tableau II.2 – Délai avec β'_2 [26, 61] et $\beta"_2$ [42].

6.9.2 Service résiduel

Intéressons-nous au flux R_2. D'après l'équations eq. II.19, ce flux bénéficie d'un service simple β'_2 dont l'expression est la suivante (Figure II.21).

$$\beta'_2(t) = \left[t - \left\lceil \frac{t}{3} \right\rceil - 1\right]^+$$

6.9.3 Les bornes

Avec ces résultats, on peut calculer une borne le délai sur le flux R_2 de 6 ms : $d(t) \leq h(\alpha_2, \beta'_2) = 6$, comme le montre le Tableau II.2. Sur la courbe β'_2 on peut constater que le service résiduel croit d'une unité puis, de deux unités ensuite avec une interruption entre ces deux périodes de croissance. Ce constat est gênant dans la mesure où non seulement R_2 devrait bénéficier de toute la puissance du serveur au moment de son service, mais en plus, à cause de la non préemption, ne peut être interrompu pendant la transmission d'un paquet. Puisque R_2 n'a que des paquets de taille 3 unités, on se serait attendu à ce que toute croissance de la fonction de la courbe résiduelle soit un multiple de 3 unités.

Cet aspect a été pris en compte dans [42]. Mais le résultat a quelques limites :

— il ne donne pas une expression analytique, mais uniquement un algorithme pour transformer une courbe calculée avec eq. II.19,
— il ne donne pas de preuve,
— le service résiduel calculé est simple.

Nous pouvons illustrer ce résultat sur les données de tu tableau III.2. Avec ce résultat, le flux R_2 bénéficie d'un service simple $\beta"_2$ dont l'expression est la suivante (Figure II.21).

$$\beta"_2(t) = \left[t - \left(\left\lfloor \frac{t}{5} \right\rfloor \times 5 + 2\right)\right]^+ + \left\lfloor \frac{t}{5} \right\rfloor \times 3 \tag{II.21}$$

Cette courbe de service nous conduit à une amélioration de la borne sur le délai qui passe de 6ms à 5 ms : $d(t) \leq h(\alpha_2, \beta"_2) = 5$.

6.10 Les défis du Calcul réseau

6.10.1 Pessimisme de l'approche

Plusieurs approches de l'analyse de performance des réseaux embarqués, comme l'approche par trajectoires ou le modèle Checking, se sont comparées au calcul réseaux. Chaque fois, les reproches qui sont formulées concernent le pessimisme du calcul réseau. Il convient aussi de souligner que les résultats du calcul réseau qui sont utilisées dans les comparaisons sont généralement ceux des travaux fondateurs des années 1990-2000 [50, 41, 23], et qu'ils prennent souvent un modèle fluide pour le calcul réseau. Lorsque l'on prend en compte les paquets en calcul réseau, les différences se réduisent voir s'annulent [32, 34, 33].

Néanmoins, il a été montré que le problème général du calcul de bornes est NP-difficile [27, 29]. Lorsque l'on se situe dans le cas général, il faudra de toute façon utiliser des approximations pour traiter des cas de grande taille.

6.10.2 Calculs effectifs

Le calcul réseau utilise des opérations comme la convolution et la déconvolution qui sont définies par des expressions mais dont il n'existe pas d'algorithmes généraux. Des algorithmes de calcul ont été proposés pour une classe particulière de fonctions. Il s'agit de la classe UPP (ultimement pseudo périodique) [31], mais l'implantation reste complexe (COINC [45], interpréteur min-plus [32])

6.10.3 Calcul locaux

Des résultats intéressants ont été obtenus en local (étude d'un serveur partagé). Ceux-ci sont exacts pour certaines politique de service (FIFO [30], Static prioriy [28]).

Il reste des améliorations à faire comme la priorité statique non préemptive dont il est question dans cette thèse. Il y a aussi d'autres politiques à traiter comme GPS et DRR qui sont abordées dans ce travail, mais aussi d'autres types de réseaux utilisés dans les systèmes embarqués, comme FlexRay [43], ou ARINC 825 [7], qui ne sont pas traités dans cette thèse.

6.10.4 Calculs globaux

L'analyse des bornes du temps de traversée d'un réseau à l'aide du calcul réseau se fait en général selon les deux principes suivants :

— approche globale (PBOO : Pay Burst Only Once). Le réseau est considéré comme un tout et les calculs sont effectués sur l'ensemble du réseau en respectant le principe du « pay burst only once » qui veux qu'un flux ne subisse l'influence du burst rien qu'une seule fois au plus dans un réseau.
— approche locale (service résiduel). Le réseau est subdivisé en plusieurs nœuds dont chacun fait l'objet d'une étude locale. Les résultats locaux sont propagés de proche en proche sur le réseau en utilisant des opérateurs dédiés à cet effet comme la convolution et la déconvolution.

Dans la pratique, on assiste très souvent à un mixage des deux approches à cause des politiques d'ordonnancement.

Des résultats d'exactitude locaux sont disponibles. Par contre, il n'existe pas de résultat d'exactitude globaux. À cet effet, il est même montré dans [116] qu'on peut obtenir des bornes aussi éloignées qu'on veut. Des travaux d' analyse globale sont en cours ([30], [85])

Chapitre III

Contributions

Cette partie présente nos principales contributions. Nous commençons dans un premier temps par exposer nos résultats obtenus sur les priorités statiques non préemptives (*Non-Preemprive Static Priority* – NP-SP), notamment sur l'amélioration de la précision des calculs locaux avec cette politique. Ensuite nous faisons une généralisation de la politique GPS et proposons une expression d'un service résiduel strict pour la politique DRR. Nous montrons aussi comment l'intégration des politiques NP-SP et DRR est possible.

1 - Priorité statique non préemptive

Cette partie est consacrée à l'étude de la politique à priorité statique. L'objectif de cette étude est l'amélioration des résultats présentés au paragraphe II.6.8.3 Nous allons détailler notre modélisation, nos résultats et illustrer par un exemple. Nous terminons cette partie par une évaluation du pessimisme de notre approche.

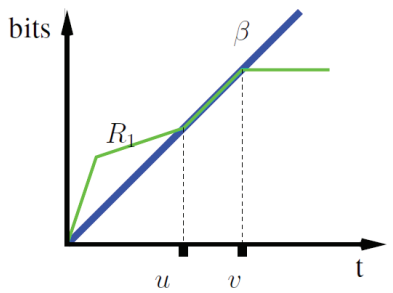


Figure III.1 – Modélisation de la priorité statique

1.1 Nouveau service résiduel

1.1.1 Modélisation de l'ordonnancement à priorité statique non préemptive

Nous considérons un serveur partagé par un ensemble de flux $R_1, \ldots, R_n$. Notons R'_i le flux sortant associé à chaque flux entrant R_i, $R = \sum_i R_i$ et $R' = \sum_i R'_i$. Nous présentons ici une modélisation de l'ordonnancement à priorité fixe non préemptive.

Notre démarche s'inscrit dans le cadre de démarche de preuves mathématiques et par conséquent présente des propriétés inhérentes à ce contexte. Les preuves sur un système ne sont jamais faites sur le composant réel, mais sur un modèle mathématique de ce composant. La validité d'une preuve dépend donc non seulement de l'exactitude de son exactitude mathématique mais également des choix de modélisation faits. Dans ce manuscrit, nous essayons de dissocier les deux.

Les paragraphes "Exclusion mutuelle" et "Agrégation et service résiduel" présentent notre modélisation. Le paragraphe "Propriétés formelles relatives à l'ordonnancement" donne les équations mathématiques dérivées de ce modèle et utilisées dans les différentes preuves.

Exclusion mutuelle En absence de backlog (c.-à-d. $R'(t) = R(t)$), le serveur a les capacités suffisantes pour servir tous les flux simultanément et la politique de service importe très peu. Pendant les périodes de backlog, nous considérons que les flux se partagent les services du serveur en *exclusion mutuelle*. Tout se passe comme s'il y avait un ordonnanceur chargé d'allouer le serveur aux flux.

Nous supposons qu'il existe, dans chaque période de backlog, $[StBl(t), t[$, une séquence ordonnée d'instants sc_i telle que à chacun de ces instants, le serveur est alloué à un flux. De plus, il existe une fonction définie dans N ou dans l'ensemble des intervalles de R^+ et à valeurs dans N. Lorsqu'elle est définie de N dans N, c'est une fonction $Sched : sc_i \mapsto [1, n]$, qui à tout instant fait correspondre le numéro du flux qui est servi par le serveur à ce moment. Précisément, $Sched(x_i) = j$ signifie que le serveur est alloué au flux j sur un intervalle $[x_i, x_{i+1}[$. Par extension sur les intervalles pour une période de backlog, $[t, s[$, $Sched([t, s[) = j$ signifie que le serveur est alloué au flux i sur l'intervalle de temps $[t, s[$. Remarquons qu'il n'y a aucune condition stipulant que $Sched(x_i) \neq Sched(x_{i+1})$. Nous appellerons ***periode de service*** P pour un flux R, une période de temps pendant lequel ce flux est servi par le serveur c.-à-d. $Sched(P) = i$. Une période de service commence à un moment précis que nous appellerons ***date de début de service***. Nous définissons $StSc(t)$ la date de début de service pour un instant t quelconque. Si t est dans une période de service i, c.-à-d. $Sched(t) = i$ alors $StSc(t)$ désigne la date de début

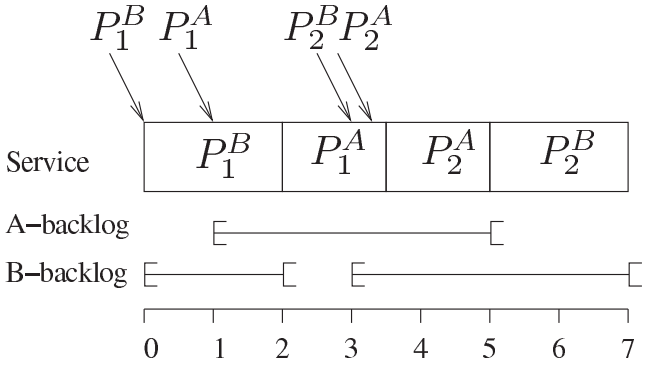


Figure III.2 – Exemple d'ordonnancement

de cette période de service, sinon $StSc(t)$ n'est pas définie.

$$StSc(t) \leq t \tag{III.1}$$

Signalons que le fait d'être dans une période de service n'implique pas forcément la présence de données en sortie. Considérerons par exemple deux flux R_H et R_L, qui se partagent les services d'un serveur fournissant un service de courbe $\beta_{R,T}$. Supposons que R_H soit plus prioritaire par rapport à R_L. Supposons aussi que R_H soit le premier flux à envoyer des données et que son premier paquet arrive au serveur à la date t. Entre l'instant t et l'instant $t+T$, la sortie du serveur peut être nulle, $(R'_H(t+D) = R'_H(t))$. Cependant, $[t, t+T]$ est une période de service pour R_H.

Nous allons considérer qu'une période de service est un intervalle ouvert à droite et fermé à gauche $[t, s[$. Pendant la période de service d'un flux, il ne peut exister de données sortant du serveur qui appartienne à un autre flux.

Pour illustrer ces notions, considérerons deux flux A et B, A étant plus prioritaire que B, A envoie les paquets P_1^A, P_2^A, et B envoie les paquets P_1^B, P_2^B (cf. Figure III.2). Les moments de service sont représentés sur la figure. L'intervalle $[1, 5[$ est une période de backlog du flux A, de même que tout intervalle inclus dans $[1, 5[$ comme $[2, 4]$ ou $]2, 5[$. Pour le flux B, $[0, 2[$ et $[2, 7[$ sont ses périodes de backlog, $[0, 7[$ est une période de backlog du flux agrégé $A + B$. On peut aussi noter les dates de début de service : $StSc_A(1) = StSc_A(3) = StSc_A(3.2) = 1$, $StSc_B(1) = 0$, $StSc_B(6) = 5$. Notons par exemple que $StSc_B(3)$ n'est pas définit, puisqu'à l'instant 3, c'est le flux A qui est servi.

Cet exemple permet aussi d'illustrer la notion de « self competing term » qui exprime le fait qu'un flux soit en compétition avec lui même. Cette notion est présente dans eq. II.20 : la période de backlog $[3, 7[$, de B doit attendre le service parce qu'il doit attendre la fin de

transmission du flux A qui lui même à son tour attend la fin de transmission du paquet en cours, P_1^B, du flux B pour des raisons de non préemption.

Agrégation et service résiduel Le comportement lié à la priorité stricte non préemptive n'est pas facile à modéliser. Considérerons à nouveau la figure III.1. Le flux le plus prioritaire R_1 est dans une période de backlog entre les instants 0 et u, et émet des données *à la cadence du serveur* jusqu'à l'instant v avant d'arrêter. Quelle peut-être la décision de l'ordonnanceur à l'instant u ? Il n'y a plus de backlog du flux R_1 et donc, l'ordonnanceur pourrait attribuer le serveur au flux moins prioritaire R_2. Cependant, R_1 continue toujours d'envoyer des données. Puisque nous nous intéressons au flux le moins prioritaire et que nous faisons une analyse pire cas, nous allons retenir l'hypothèse selon laquelle au moment u, le serveur est à mesure de se rendre compte que R_1 envoie toujours les données et donc continue de le servir jusqu'à l'instant v. Nous modélisons la priorité statique de façon suivante : soit t le début d'une période de backlog. S'il y a du flux entrant du flux le plus prioritaire, alors le serveur lui est alloué *jusqu'à ce qu'il cesse d'émettre ou alors que son émission ne soit pas suffisamment conséquente pour occuper pleinement le serveur*. Dans le cas contraire, le serveur est alloué à un autre flux moins prioritaire, *jusqu'à la fin de transmission d'un de ses paquets* (c.-à-d. jusqu'à l'instant s tel que $R'_2(s) - R'_2(t) = l_2$). À la fin d'une période de service, l'ordonnanceur alloue de nouveau le serveur comme s'il s'agissait d'un début d'une période de backlog.

Propriétés formelles relatives à l'ordonnancement Notre modélisation de la priorité fixe implique certaines propriétés qui seront utilisées dans les preuves. Considérons donc un serveur S, offrant un service strict β et partagé par un ensemble de flux $R_1, \ldots, R_n$, suivant une politique de priorité statique non préemptive telle que modélisée aux paragraphes III.1.1.1 et III.1.1.1. Par définition d'une période de backlog, $[t, s[$ du flux agrégé $R = \sum_i R_i$, on a :

$$R'(s) - R'(t) \geq \beta(s - t) \tag{III.2}$$

Soit $[t, s[$ une période de backlog du flux i alors, le début de service intervient toujours après le début de backlog (eq. III.3). La modélisation de l'ordonnancement dans un contexte d'exclusion mutuelle implique que lorsqu'un flux n'est pas servi, il ne bénéficie d'aucune donnée en sortie. (eq. III.4). De plus, si le serveur S offre un service strict de courbe β, alors un flux en période de service entre les instants t et s bénéficie d'une sortie de données de quantité au moins égale

à $\beta(s-t)$ (eq. III.5)

$$\forall u \in [t,s[:\ StSc_i(u) \geq StBl_i(u) \tag{III.3}$$

$$Sched([t,s[) = i \implies \forall j \neq i, \forall u,v \in [t,s[:\ R'_j(u) = R'_j(v) \tag{III.4}$$

$$Sched([t,s[) = i, \text{strict service } \beta \implies R'_i(s) - R'_i(t) \geq \beta(s-t) \tag{III.5}$$

Par définition du début de service et de la période de backlog :

$$R'(StSc(t)) = R'(StBl(t)) \tag{III.6}$$

$$\text{et donc, } R'_i(StSc_i(t)) = R'_i(StBl_i(t)) \tag{III.7}$$

Cette relation permet juste de préciser que entre le moment où un flux commence à accumuler du backlog et celui où celui-ci commence à être servi, il ne bénéficie d'aucun service.

1.1.2 Théorème : service strict

Théorème 7 (NP-SP service résiduel)**.** *Considérons un serveur offrant à trois flux, R_1, R_2 et R_3, un service strict dont la courbe est représentée par une fonction $\beta \in \mathcal{F}$. Supposons que le flux R_2 (resp. R_3) émette ses données sous forme de paquets de taille fixe l_2 (resp. de taille maximale l_3). Si le flux R_1 (resp. R_2) est contraint par une courbe d'arrivée α_1 (resp. α_2) et R_1 possède une priorité non préemptive sur R_2 et R_2 à son tour possède une priorité non préemptive sur le flux R_3, alors le serveur garanti au flux R_2 un service strict de courbe β_2^{np} défini dans l ?équation eq (III.8).*

$$\beta_2^{np}(t) = \min \begin{cases} i \times l_2 \\ \beta(t) + (i-1) \times l_2 - \beta(\chi'_i) \\ \beta(t) + (i-1) \times l_2 \\ \quad -\beta(\chi"_i + \psi_2) + \beta(\Delta + \psi_2) \end{cases} \tag{III.8}$$

Avec $i = \max\{j : \chi_j \leq t\}$ et les définitions :

$$\psi_1 = (\beta - \alpha) \uparrow^{-1}_{\sup} (0) \qquad \psi_i = \beta^{-1}_{\sup}(l_i) \text{ pour } i \in \{1,2\}$$

$$\chi'_i = ((\beta - \alpha_1) \uparrow)^{-1}_{\inf}(l_3 + (i-1) \times l_2) \qquad \chi"_i = ((\beta - \alpha_1) \oslash \delta_{\psi_2})^{-1}_{\inf}(i \times l_2)$$

$$\chi_i = \max\{\chi'_i, \chi"_i\} \qquad \Delta = (\alpha_2)^{-1}_{\inf}(2 \times l_2) - \psi_2$$

Nous donnons ici une définition intuitive des termes utilisés : ψ_i, est une borne supérieure

Notations liées au comportement réel des flux	
R_i	fonction cumulée croissante du flux d'entrée i
R'_i	fonction cumulée croissante du flux de sortie i
x_i	$i^{ième}$ début de service d'un paquet du flux R_i dans un intervalle $[t, s[$
w_i	durée de la $i^{ième}$ période de service dans un intervalle $[t, s[$
$StBl_i$	date de début période de backlog du flux R_i
$StSc_i$	date de début de service du flux R_i
t_i^c	$StSc_i(StBl_1(t))$
Notations liées au modèle	
β	service fourni par le serveur
α_i	courbe d'arrivée exprimant la contrainte à respecter par le flux R_i
χ_i	borne supérieure de la durée de temps précédant le début de la $i^{ième}$ période de d'un paquet du flux R_i
ψ_i	borne supérieure du temps nécessaire au serveur pour servir un paquet de longueur l_i
Δ	borne supérieure de la période de temps pendant lequel il n'y a pas de backlog entre l'émission de deux paquets consécutifs, compte tenu de la courbe d'arrivé à respecter

Tableau III.1 – Notations utilisées dans la preuve

du temps nécessaire au serveur pour servir un paquet de taille l_i, et χ_i est une borne supérieure du temps d'attente du $i^{ième}$ avant qu'il ne soit servi.

1.1.3 Preuve

Considérons une période de backlog $[t, s[$, du flux R_2. Cet intervalle servira dans toute la démonstration.

Quelques définitions supplémentaires Soit x_i le $i^{ième}$ début de service de paquet du flux R_2 après l'instant t, c.-à-d. $x_1 = \min\{sc_i \mid sc_i \geq t, Sched(sc_i) = 2\}$. Cet $i^{ième}$ *période de service de paquet* de R_2 intervenant dans une période de backlog $[t, s[$ est un intervalle $[x_i, x_i + w_i[$, avec $w_i = \inf\{t \mid R'_2(x_i + t) - R'_2(x_i) \geq l_2\}$. Ce qui signifie que

$$R'_2(x_i) - R'_2(x_1) = (i-1) \times l_2 \tag{III.9}$$

$$R'_2(x_i + w_i) - R'_2(x_1) = i \times l_2 \tag{III.10}$$

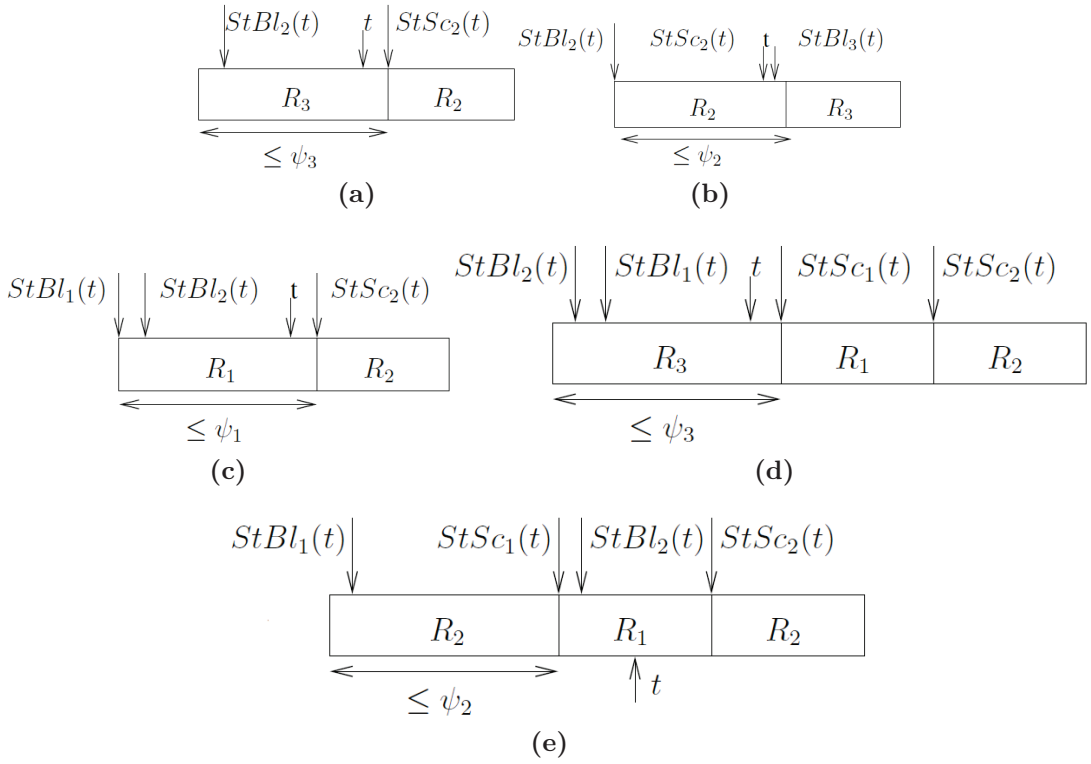


Figure III.3 – Différents cas pour $StSc_2(t)$

Entre t et x_1, R_2 peut avoir reçu du service. Cependant, puisque t est arbitraire, t n'est pas nécessairement un début de service. Donc,

$$0 \leq R'_2(x_1) - R'_2(t) < l_2 \tag{III.11}$$

Ce qui suit se déduit des équations (III.9), (III.10) et (III.11).

$$R'_2(x_i) - R'_2(t) \geq (i-1) \times l_2 \tag{III.12}$$

$$R'_2(x_i + w_i) - R'_2(t) \geq i \times l_2 \tag{III.13}$$

Preuve du théorème Étant donnée une période de backlog $[t, s[$, de R_2, plusieurs situations peuvent se présenter.

t n'appartient pas à une période de backlog de R_1. Dans ce cas, R_2 est le flux de plus haute priorité dans la file d'attente. Sa transmission peut cependant être retardée par au plus un paquet du flux moins prioritaire R_3. On distingue alors deux sous-cas à l'instant $StBl(t)$:

a. un paquet du flux R_3 est en train d'être servi. Comme le montre la Figure III.3a, le flux R_2 peut attendre la fin de transmission d'un paquet du flux R_3 avant de transmettre.

b. aucun paquet du flux R_3 n'est en train d'être servi. Dans ce cas, comme le montre la Figure III.3b, le flux R_2 est directement servi. Il n'y a pas de temps d'attente.

t appartient à une période de backlog du flux R_1. Dans ce cas, comme R_1 est plus prioritaire, il est servi. Trois cas peuvent alors exister à l'instant $StBl(t)$.

c. le flux R_1 est directement servi comme le montre la figure III.3c. Le flux R_2 doit alors attendre la fin de transmission du flux R_1 (jusqu'à ce que sa file d'attente soit vide) avant d'être servi.

d. le flux R_3 est en train d'être servi. La figure III.3d présente cette situation. Dans ce cas, le flux R_2 attend la fin de transmission du paquet en cours du flux R_3, suivie de la transmission complète du flux R_1, jusqu'à ce que ce dernier vide sa file d'attente. Il est important de remarquer que durant la transmission du paquet en cours du flux R_3, le flux R_1 continue d'accumuler des données dans sa file d'attente, entraînant ainsi un backlog plus important.

e. un paquet du flux R_2 préalablement placé en file d'attente est en train d'être servi. C'est une situation qui permet d'illustré ce que nous avons appelé « self competitive term » . Comme on peut le voir sur la figure III.3e, le flux R_2 attend que le flux R_1, plus prioritaire vide sa file d'attente, pendant que le flux R_1 à son tour attend la fin de transmission du paquet en cours du flux R_2

Les trois premiers cas décrits ci-dessus (figures III.3a-III.3b-III.3c) sont des cas particuliers des deux cas plus généraux (figures III.3d-III.3e). Nous allons nous concentrer que sur ces deux derniers cas.

Nous utiliserons l'expression « cas 3-1-2 » la situation dans laquelle le flux R_2 attend la fin de transmission du flux R_1 qui à son tour est en train d'attendre la fin de transmission du paquet en cours du flux R_3 (figure III.3d). De même, nous utiliserons l'expression « cas 2-1-2 » la situation dans laquelle le flux R_2 attend la fin de transmission du flux R_1 qui à son tour est en train d'attendre la fin de transmission du paquet en cours du flux R_2 (figure III.3e).

Dans les cas 3-1-2 et 2-1-2, (le flux R_1 commence à accumuler du backlog dans sa file d'attente à la date $StBl_1(t)$ pendant que le flux R_i ($i \in \{2,3\}$) est en train d'être servi. $StBl_1(t)$ appartient à une période de service de R_i. Cette période de service débute à l'instant $StSc_i(StBl_1(t))$ ($i \in \{2,3\}$). Pour la suite, notons $t_i^c = StSc_i(StBl_1(t))$, et remarquons que

$$t_i^c \leq StBl_1(t) \tag{III.14}$$

Avant de démontrer le théorème, voici quelques résultats importants.

Lemme 1. *cas 2-1-2*
$StSc_1(t) - t_2^c \leq \psi_2$

Démonstration. Un paquet du flux R_2 est servi entre t_2^c et $StSc_1(t)$.

$$\begin{aligned} l_2 &= R_2'(StSc_1(t)) - R_2'(t_2^c) && \text{(Seul le flux } R_2 \text{ est servi)} \\ &\geq \beta(StSc_1(t) - t_2^c) && \text{(Par définition du service strict)} \end{aligned}$$

Puisque β est une fonction non décroissante, $StSc_1(t) - t_2^c \leq \beta_{sup}^{-1}(l_2) = \psi_2$. □

Lemme 2. *cas 3-1-2*
$StSc_1(t) - t_3^c \leq \psi_3$

Démonstration. Seul un paquet du flux R_3 est servi entre t_3^c et $StSc_1(t)$.
$R'(t_3^c + u) - R'(t_3^c) \geq \beta(u)$ (Par définition du service strict). D'où,
$\inf\{u | R'(t_3^c + u) - R'(t_3^c) \geq l_3\} \leq \inf\{u | \beta(u) \geq l_3\}$
$\inf\{u | R'(t_3^c + u) - R'(t_3^c) \geq l_3\} \leq \psi_3$ (Par définition de ψ_3)
$\inf\{u | R_3'(t_3^c + u) - R_3'(t_3^c) \geq l_3\} \leq \psi_3$ (Seul le flux R_3 est servi)
$StSc_1(t) - t_3^c \leq \psi_3$. □

Les deux résultats suivants donnent une borne supérieure du temps qui s'écoule avant le début de service du $i^{ième}$ paquet du flux R_2

Lemme 3. *cas 2-1-2 :* $x_i - t_2^c \leq \chi"_i + \psi_2$.

Démonstration. Comme pour le lemme 1, nous allons utiliser une borne sur l'expression $R_2'(x_i) - R_2'(t_2^c)$ pour déduire une borne sur $x_i - t_2^c$. Du fait que la définition de $\chi"_i$ s'appuie sur la pseudo-inverse de $\beta - \alpha$, qui n'est pas nécessairement non décroissant, une attention particulière se doit d'être accordée pendant la manipulation des expressions à base de l'opérateur inf. C'est la raison pour laquelle une variable $u \in [0, s - t_2^c]$ est introduite dans la preuve.

Par définition de ce cas, le flux R_3 n'est pas servi dans une période de backlog du flux R_2

$$R_3'(t_2^c + u) - R_3'(t_2^c) = 0 \tag{III.15}$$

Pour le flot R_1,

$$
\begin{aligned}
&R'_1(t_2^c+u) - R'_1(t_2^c)\\
&= R'_1(t_2^c+u) - R'_1(StSc_1(t)) + R'_1(StSc_1(t)) - R'_1(t_2^c)\\
&= R'_1(t_2^c+u) - R'_1(StSc_1(t))\\
&\quad \text{(puisque } R'_1(StSc_1(t)) = R'_1(t_2^c)\text{, d'après l'équation eq. III.4)}\\
&= R'_1(t_2^c+u) - R'_1(StBl_1(t)) \qquad \text{(d'après l'équation eq. III.7)}\\
&\leq R_1(t_2^c+u) - R_1(StBl_1(t)) \quad (\text{car } R' \leq R)\\
&\leq R_1(t_2^c+u) - R_1(t_2^c) \quad \text{(d'après l'équation eq. III.14)}\\
&\leq \alpha_1(u) \; c.-\text{à}-d.
\end{aligned}
$$

$$R'_1(t_2^c+u) - R'_1(t_2^c) \leq \alpha_1(u) \tag{III.16}$$

D'après la définition du service strict,

$$R'(t_2^c+u) - R'(t_2^c) \geq \beta(u) \tag{III.17}$$

Une combinaison des équations eq III.17, eq. III.16 et eq III.15, donne

$$
\begin{aligned}
&R'_2(t_2^c+u) - R'_2(t_2^c) \geq \beta(u) - \alpha(u)\\
\implies &\inf\{u | R'_2(t_2^c+u) - R'_2(t_2^c) > i \times l_2\} \leq \inf\{u | \beta(u) - \alpha(u) > i \times l_2\} = \chi"_i + \psi_2
\end{aligned}
$$

Puisque $u \mapsto R'_2(t_2^c+u) - R'_2(t_2^c)$ est non decroissant : $x_i - t_2^c \leq \chi"_i + \psi_2$. D'après le lemme 1 $x_i - StSc_1(t) \leq \chi"_i$, d'où $x_i - t_2^c \leq \chi"_i + \psi_2$. (Puisque $StSc_1(t) - t_2^c \leq \psi_2$) □

Lemme 4. *cas 3-1-2 : $x_i - t_3^c \leq \chi'_i$*

Démonstration.

$$R'_3(t_3^c+u) - R'_3(t_3^c) \leq l_3 \tag{III.18}$$

Pour le flux R_1,

$$R'_1(t_3^c+u) - R'_1(t_3^c) \leq \alpha_1(u) \tag{III.19}$$

De part la définition d'une courbe de service strict,

$$R'(t_3^c+u) - R'(t_3^c) \geq \beta(u) \tag{III.20}$$

$$
\begin{aligned}
&R'_2(t_3^c+u) - R'_2(t_3^c) = [R'(t_3^c+u) - R'(t_3^c)] - [R'_1(t_3^c+u) - R'_1(t_3^c)] - [R'_3(t_3^c+u) - R'_3(t_3^c)]\\
&\geq \beta(u) - \alpha(u) - l_3 \quad \text{(D'après les équations eq. III.19 et eq. III.20)} \; c.-\text{à}-d.
\end{aligned}
$$

$$R'_2(t_3^c + u) - R'_2(t_3^c) \geq \beta(u) - \alpha(u) - l_3 \tag{III.21}$$

Donc,

$$\begin{aligned}
\inf\{u | R'_2(t_3^c + u) - R'_2(t_3^c) > (i-1) \times l_2\} &\leq \inf\{u | \beta(u) - \alpha(u) - l_3 > (i-1) \times l_2\} \\
&\iff x_i - t_3^c \leq \chi'_i \\
&\implies x_i - StSc_1(t) \leq \chi'_i \quad \text{D'après le lemme 2}
\end{aligned}$$

Puisque le flux R_3 est le moins prioritaire de tous et que ce flux ne peut tout au plus servir qu'un seul paquet dans une période de backlog d'un autre flux, plus prioritaire que lui,

$$R'_3(x_i) - R'_3(t_3^c) = l_3 \tag{III.22}$$

Par définition de x_i et en prenant en compte le contexte du cas 3-1-2,

$$R'_2(x_i) - R'_2(t_3^c) = (i-1)l_2 \tag{III.23}$$

Pour le flux R_1,

$$\begin{aligned}
&R'_1(x_i) - R'_1(t_3^c) \\
&= R'_1(x_i) - R'_1(StSc_1(t)) + R'_1(StSc_1(t)) - R'_1(t_3^c) \\
&= R'_1(x_i) - R'_1(StSc_1(t)) && (\text{car } R'_1(StSc_1(t)) - R'_1(t_3^c) = 0) \\
&= R'_1(x_i) - R'_1(StBl(t)) && (\text{car } R'_1(StSc_1(t)) - R'_1(StBl(t)) = 0)
\end{aligned}$$

$$\begin{aligned}
R'_1(x_i) - R'_1(t_3^c) &\leq R_1(x_i) - R_1(StBl(t)) && (\text{car } R' \leq R) \\
&\leq R_1(x_i) - R_1(StSc_1(t) - \psi_3) && (\text{cas 3-1-2 cf. figure III.3-e}) \\
&\leq R_1(x_i) - R_1(t_3^c) && (\text{D'après le lemme 2}) \\
&\leq \alpha_1(x_i - t_3^c)
\end{aligned}$$

Donc,

$$R'_1(x_i) - R'_1(t_3^c) \leq \alpha_1(x_i - t_3^c) \tag{III.24}$$

De part la définition d'une courbe de service strict,

$$\begin{aligned}\beta(x_i - t_3^c) &\leq R'(x_i) - R'(t_3^c)\\ &= R'_1(x_i) - R'_1(t_3^c) + R'_2(x_i) - R'_2(t_3^c) + R'_3(x_i) - R'_3(t_3^c)\\ &\leq \alpha_1(x_i - t_3^c) + (i-1)l_2 + l_3\\ &\quad \text{(D'après les équations eq. III.24, eq. III.23 et eq. III.22)}\\ \iff \beta(x_i - t_3^c) - \alpha_1(x_i - t_3^c) &\leq l_3 + (i-1) \times l_2\end{aligned}$$

Puisque $\chi'_i = \inf\{t | \beta(t) - \alpha_1(t) > l_3 + (i-1) \times l_2\}$ on obtient donc, $x_i - t_3^c \leq \chi'_i$ □

Démonstration. (du théorème 7)

Étant donné $[t, s[$, une période de backlog du flux R_2. Posons $i = \max\{j \in \mathbb{N} | x_j \leq s\}$.

Si $s \geq x_i + w_i$: s n'est pas dans une période de service du flux R_2

$R'_2(s) - R'_2(t) \geq i \times l_2$ (par définition de x_i et w_i, & eq. III.13)

If $x_i \leq s < x_i + w_i$: s est dans une période de service du flux R_2

c.-à-d. ($R'_1(s) = R'_1(x_i)$ et $R'_3(s) = R'_3(x_i)$: eq. III.4). Et donc,

$$\begin{cases} R'_1(s) - R'_1(t) = R'_1(x_i) - R'_1(t) \\ R'_3(s) - R'_3(t) = R'_3(x_i) - R'_3(t) \end{cases}$$

soit, $R'_2(s) - R'_2(t) \geq \beta(s-t) - (R'_1(x_i) - R'_1(t) + R'_3(x_i) - R'_3(t))$

La suite nécessite d'avoir une borne sur les expressions $R'_1(x_i) - R'_1(t)$ et $R'_3(x_i) - R'_3(t)$.

— case 2-1-2 (Figure III.3e) Ce cas est un peu particulier, car il y a deux périodes de backlog distinctes pour R_2 à considérer dans la preuve : R_2 est servi une première fois, jusqu'à l'instant $StSc_1(t)$, puis une seconde, qui commence à $StBl_2(t)$, qui est la période de backlog générale à laquelle apparient $[t, s[$. C'est la première période qui retarde le service du flux R_1, et cette attente va augmenter son backlog, ce qui crée le "self competing term".

Définissons t_0 l'instant de démarrage du service du paquet de R_2 qui bloque le flux R_1 :

$$t_0 \leq StBl_1(t) \leq StSc_1(t) \tag{III.25}$$

$$R'_2(StSc_1(t)) - R'_2(t_0) = l2 \tag{III.26}$$

Il est clair dans ce cas que : $R'_3(x_i) - R'_3(t) = 0$, ce qui donne $R'_2(s) - R'_2(t) \geq \beta(s-t) - (R'_1(x_i) - R'_1(t))$.

Dans ce cas, où un paquet de R_1 arrive (à $StBl_1(t)$) pendant le service d'un paquet de R_2, (cf Figure III.3e), définissons t_0 l'instant de démarrage du service du paquet de R_2

qui bloque le flux R_1. Puisque l'on a des paquets de taille fixe, on a III.28. De plus, du fait de l'exclusion mutuelle, aucun des flux R_1 ou R_3 n'est pas servi pendant une période de service du flux R_2, doù

$$t_0 \leq StBl_1(t) \leq StSc_1(t) \tag{III.27}$$
$$R'_2(StSc_1(t)) - R'_2(t_0) = l_2 \tag{III.28}$$
$$R'_1(StSc_1(t)) - R'_1(t_0) = 0 \tag{III.29}$$
$$R'_3(StSc_1(t)) - R'_3(t_0) = 0 \tag{III.30}$$

Considérons $R'_1(x_i) - R'_1(t)$:

$$\begin{aligned} R'_1(x_i) - R'_1(t) &\leq R'_1(x_i) - R'_1(StBl_2(t)) && \text{Il est trivial que } StBl_2(t) \leq t \\ &= (R'_1(x_i) - R'_1(t_0)) - (R'_1(StBl_2(t)) - R'_1(t_0)) && \text{d'après III.29 :} R'_1(t_0) = R'_1(StSc_1(t) \end{aligned}$$

Considérons tout d'abord $R'_1(x_i) - R'_1(t_0)$: seul R_2 est servi pendant $[StSc_2(t_0), t_0]$ (par définition d'un début de service), et de même du $t_0, StBl_1(t)$ (par définition de t_0), d'où $R'_1(t_0) = R'_1(StBl_1(t))$, d'où $R'_1(x_i) - R'_1(t_0) = R'_1(x_i) - R'_1(StBl_1(t))$. Comme on a égalité entre flux entrant et flux sortant en début de backlog, $R'_1(StBl_1(t) = R_1(StBl_1(t)$, et on a toujours $R'_1 \leq R_1$, d'où

$$\begin{aligned} R'_1(x_i) - R'_1(t_0) &= R'_1(x_i) - R'_1(StBl_1(t)) \\ &\leq R_1(x_i) - R_1(StBl_1(t)) \\ &\leq R_1(x_i) - R_1(t_2^c) && \text{d'après eq. III.14} \\ &\leq \alpha_1(x_i - t_2^c) \end{aligned}$$

Considérons maintenant $R'_1(StBl_2(t)) - R'_1(t_0)$. L'intervalle $[t_0, StBl_2(t))$ est une période de backlog : sur $[t_0, StSc_1(t)[$ seul R_2 est servi (et un seul paquet est servi, de taille l_2), puis sur $[StSc_1(t), StBl_2(t)[$, seul R_1 est servi. On a donc

$$R'(StBl_2(t)) - R'(t_0) = l_2 + R'_1(StBl_2(t)) - R'_1(StSc_1) \geq \beta(StBl_2(t) - t_0) \tag{III.31}$$

Ce qui mène à $R'_1(StBl_2(t)) - R'_1(t_0) \leq \beta(StBl_2(t) - t_0) + l_2$
Reste à borner $StBl_2(t) - t_0$. Soit t' la date d'arrivée du paquet servit à partir de t_0. On a $t' \leq t_0$. Sur l'intervalle $[t', StBl_2(t)]$, il y a au plus arrivée de deux paquets. Donc $StBl_2(t) - t_0 \leq StBl_2(t) - t' \leq (\alpha_1)^- 1_{\inf}(2 \times l_2)$, ce qui donne $StBl_2(t) - t_0 \leq \Delta + \psi_2$

Nous avons, $R'_2(s) - R'_2(t) \geq \beta(s-t) - \beta(\chi"_i + \psi_2) + \beta(\Delta + \psi_2) + (i-1) \times l_2$

— cas 3-1-2

$$\begin{aligned}
R'_1(x_i) - R'_1(t) &\leq R'_1(x_i) - R'_1(StSc_1(t)) \quad \text{(puisque } StSc_1(t) \leq t \text{, eq III.1)} \\
&\leq R'_1(x_i) - R'_1(StBl_1(t)) \quad \text{(car } StBl_1(t) \leq StSc_1(t) \text{, eq. III.3)} \\
&\leq R'_1(x_i) - R_1(StBl_1(t)) \quad \text{(car } R_1(StBl_1(t)) = R'_1(StBl_1(t)) \text{, eq. II.12)} \\
&\leq R_1(x_i) - R_1(StBl_1(t)) \quad \text{(car } R'_1(x_i) \leq R_1(x_i)) \\
&\leq R_1(x_i) - R_1(t^c_3) \quad \text{(D'après l'équation III.14)} \\
&\leq \alpha_1(x_i - t^c_3) \\
&\leq \alpha_1(\chi'_i) \quad \text{(D'après le lemme 4). Ce qui donne alors :}
\end{aligned}$$

Avec, $R'_3(x_i) - R'_3(t) \leq l_3$, on obtient $R'_2(s) - R'_2(t) \geq \beta(s-t) - \alpha_1(\chi'_i) - l_3$.
Par définition de χ'_i, $\beta(\chi'_i) - \alpha_1(\chi'_i) > l_3 + (i-1)l_2$.
Et donc, $R'_2(s) - R'_2(t) \geq \beta(s-t) - \beta(\chi'_i) + (i-1)l_2$

Soit finalement,

$$\begin{aligned}
R'_2(s) - R'_2(t) \geq \min\{&i \times l_2, \beta(s-t) - \beta(\chi'_i) + (i-1) \times l_2, \\
&\beta(s-t) - \beta(\chi"_i + \psi_2) + \beta(\Delta + \psi_2) + (i-1) \times l_2\}
\end{aligned}$$

□

1.1.4 Théorème : service faiblement strict

$$\beta^{np}_{2Ws}(t) = \min \begin{cases} i \times l_2 \\ \beta(t) - \beta(\chi'_i) + (i-1) \times l_2 \\ \beta(t+\Delta) - \beta(\chi"_i + \psi_2) + i \times l_2 \end{cases} \tag{III.32}$$

avec i tel que $\chi(t) = \chi_i$ et $\Delta = \alpha^{-1}(2 \times l_2) - \psi_2$

Théorème 8. *Considérons un serveur offrant à trois flux, R_1, R_2 et R_3, un service faiblement strict dont la courbe est représentée par une fonction non décroissante β. Supposons que le flux R_2 (resp. R_3) émette ses données sous forme de paquets de de taille fixe l_2 (resp. de taille maximale l_3). Si le flux R_1 (resp. R_2) est contraint par une courbe d'arrivée α_1 (resp. α_2), si de plus le flux R_1 possède une priorité non préemptive sur le flux R_2 et qu'à son tour le flux R_2 possède une priorité non préemptive sur le flux R_3, alors le serveur garanti au flux R_2 un service faiblement strict $\beta^{np}_{2Ws}(t)$ défini par l'équation (III.32).*

1.1.5 Preuve

Démonstration. Il suffit juste de montrer que $\beta(t+\Delta) - \beta(\chi"_i + \psi_2) + i \times l_2$ est courbe de service faiblement strict dans le cas 2-1-2. En effet, tout service strict est aussi faiblement strict et la différence entre les équations eq. III.8 et eq. III.32 concerne le cas 2-1-2.

Soit s, un instant donné, définissons x_i le $i^{ième}$ début de service de paquet du flux R_2 après $StBl_2(s)$. Posons $i = \max\{j \in \mathbb{N} | x_j \leq s\}$.

Si $s \geq x_i + w_i$: s n'est pas dans une période de backlog du flux R_2
$R'_2(s) - R'_2(StBl_2(s)) \geq i \times l_2$ (par définition de x_i)

Si $x_i \leq s < x_i + w_i$: s est dans une période de service du flux R_2 ($R'_1(s) = R'_1(x_i)$ et $R'_3(s) = R'_3(x_i)$). D'où
$$\begin{cases} R'_1(s) - R'_1(StBl_2(s)) = R'_1(x_i) - R'_1(StBl_2(s)) \\ R'_3(s) - R'_3(StBl_2(s)) = R'_3(x_i) - R'_3(StBl_2(s)) \end{cases}$$

donc,

$$\begin{aligned} & R'_2(s) - R'_2(StBl_2(s)) \\ & = (R'_2(x_i) - R'_2(StBl_2(s)) + (R'(s) - R'(StBl_2(s) - \Delta)) - (R'(x_i) - R'(StBl_2(s) - \Delta)) \\ & = (R'_2(x_i) - R'_2(StBl_2(s)) + (R'(s) - R'(StBl_2(s) - \Delta)) \\ & - (R'_1(x_i) - R'_1(StBl_2(s) - \Delta)) - (R'_2(x_i) - R'_2(StBl_2(s) - \Delta)) - (R'_3(x_i) - R'_3(StBl_2(s) - \Delta)) \\ & = (R'(s) - R'(StBl_2(s) - \Delta)) - (R'_1(x_i) - R'_1(StBl_2(s) - \Delta)) - (R'_3(x_i) - R'_3(StBl_2(s) - \Delta)). \end{aligned}$$

$$\begin{aligned} R'_1(x_i) - R'_1(StBl_2(s)) &\leq \alpha_1(x_i - t_2^c) \quad \text{(from Eq. III.16)} \\ &\leq \alpha_1(\chi"_i + \psi_2) \quad \text{(D'après le lemme 3). Ce qui donne :} \end{aligned}$$

$$\begin{cases} R'_1(x_i) - R'_1(StBl_2(s)) \leq \alpha_1(\chi"_i + \psi_2) \\ R'_3(x_i) - R'_3(StBl_2(s)) = 0 \end{cases}$$

Par conséquent, $R'_2(s) - R'_2(StBl_2(s)) \geq \beta(s-t) - \alpha_1(\chi"_i + \psi_2)$.
De par la définition de $\chi"_i$, $\beta(\chi"_i + \psi_2) - \alpha_1(\chi"_i + \psi_2) > i \times l_2$.

On a alors,

$$\begin{aligned} R'_2(s) - R'_2(StBl_2(s)) &= (R'(s) - R'(StBl_2(s) - \Delta)) - (R'_1(x_i) - R'_1(StBl_1(s) - \Delta)) \\ &\geq \beta(s - StBl_2(s) + \Delta) - \alpha_1(\chi"_i + \psi_2) \\ &\geq \beta(s - StBl_2(s) + \Delta) - \beta(\chi"_i + \psi_2) + i \times l_2 \\ &\text{(Par définition de } \chi"_i) \end{aligned}$$

□

1.1.6 Généralisation à un nombre quelconque de flux

Corollaire 9 (Généralisation à un nombre quelconque de flux)**.** *Le résultat avec trois flux se généralise aisément pour un nombre quelconque de flux $A_1, \ldots, A_n$. Considérons un flux A_i, et constituons les flux agrégés de la manière suivante : $R_2 = A_i$, $R_1 = \sum_{j=1}^{i-1} A_j$, $R_3 = \sum_{j=i+1}^{n} A_j$. La somme des flux plus prioritaires peut être vu comme un seul flux pour lequel la courbe d'arrivée serait la somme des courbes d'arrivée individuelles. Il en est de même pour la somme des flux les moins prioritaires. Dans le cas particulier où $A_i = A_n$, il suffit juste de considérer $R_3 = \alpha_3 = l_3 = 0$.*

1.1.7 Implantation

Le service résiduel de l'équation III.8 a une forme analytique, mais nous ne savons pas à ce jour l'exprimer complètement à l'aide des opérateurs basiques du calcul réseau.

Il s'agit d'une expression qui dépend d'une suite infinie de valeurs χ_i, χ'_i et $\chi"_i$. Pour chaque i, les valeurs peuvent être calculées avec les opérateurs de base (soustraction, clôture croissante, pseudo-inverse).

Nous pouvons donc calculer, non pas β_2^{np}, mais $\dot{\beta}_2^h$ fonction égale à β_2^{np} jusqu'à un horizon fini h.

Définissons l'opérateur $\overset{h}{=}$ défini par $f \overset{h}{=} g \iff \forall x \leq h, f(x) = g(x)$.

Mais cette approximation finie est suffisante en général.

En fait, lorsque l'on cherche à calculer une déviation horizontale $h(\alpha, \beta)$, il est inutile de considérer les valeurs où $\alpha < \beta$.

Formellement, considérons une fonction sous-additive α et une fonction croissante β. Posons h le plus petit réel tel que, $x > h \implies \beta(x) > \alpha(x)$, ce qui est équivalent à $(\beta - \alpha)^{-}1_{\text{sup}}(0)$. Alors, pour toute fonction $\dot{\beta}$ croissante telle que $\beta \overset{h}{=} \dot{\beta}$, $h(\alpha, \beta) = h(\alpha, \dot{\beta})$.

	Period	Size	α_i	delay β'	delay β"	delay β^{np}
R_1	3	1	$\left\lceil \frac{t}{3} \right\rceil$	4	4	4
R_2	9	3	$3\left\lceil \frac{t}{9} \right\rceil$	6	5	5
R_3	4	1	$\left\lceil \frac{t}{4} \right\rceil$	6	6	6

Tableau III.2 – Comparaison de méthodes sur l'exemple du Tableau III.2

i	χ'_i	χ"$_i$	χ_i
1	2	-4	2
2	7	1	7
3	11	5	11
4	16	10	16

Tableau III.3 – Calcul de β_2

On peut donc ne calculer $\dot{\beta}_2^h$ que jusqu'à h. On peut donc calculer les valeurs de χ_i, χ'_i et χ"$_i$ jusqu'à un point ou $\dot{\beta}_2^h$ dépasse α.

1.2 Exemple d'application

Pour illustrer, reconsidérons l'exemple déjà présenté dans la partie état de l'art (section II.6.9, Tableau III.2) avec un serveur offrant un service strict de courbe $\beta(t) = t$ à trois flux R_1, R_2 et R_3. Chaque flux R_i est contraint par une courbe d'arrivée α_i , et possède des paquets de taille fixe l_i, comme le rappelle le tableau III.2.

Évaluons β_2^{np}, le service résiduel offert au flux R_2. $\Delta = 6$ et le tableau III.3 montre différentes valeurs calculées pour la variable χ_i.

Dans ce cas, notre méthode n'apporte pas d'amélioration par rapport aux méthodes existantes.

1.3 Précision de l'approche

Le théorème 7 propose une expression analytique pour évaluer des bornes sur le délai pire cas de traversée du réseau. On est en droit de se poser la question relative à la précision de ces résultats. Concrètement, il faut répondre à la question de savoir si les bornes dérivées de ces résultats sont atteignables ou alors s'il s'agit juste de sur-approximation. Jusqu'à présent, nous n'avons aucune preuve nous permettant d'affirmer l'une de ces thèse ou son contraire et

Flow i	Period	Size (l_i)	α_i	Vrai pire délai	Nos bornes
R_1	2.5	2.5	$2.5\left\lceil\frac{2t}{5}\right\rceil$	2	2
R_2	3.5	2.5	$2.5\left\lceil\frac{2t}{7}\right\rceil$	3	3
R_3	3.5	2.5	$2.5\left\lceil\frac{2t}{7}\right\rceil$	3.5	3.5

Tableau III.4 – Exemple sur le bus CAN

nous ne connaissons pas non plus de contre exemple permettant de conclure qu'il s'agit de sur-approximation.

Les problématiques d'évaluations de délai pire cas de bout-en-bout se posent aussi sur le bus automobile CAN sur lequel est implémenté le mode de gestion avec priorité statique non préemptive.

Il existe un algorithme, appliqué au bus CAN, qui donne le pire cas atteignable. Comme il est difficile de se comparer formellement à ses résultats, nous allons le faire sur un grand nombre de test aléatoire.

1.3.1 Comparaison sur un exemple

Pour illustrer, considérons l'exemple du tableau III.4. Dans cet exemple, trois flux R_1, R_2 et R_3, classés du plus prioritaire au moins prioritaire, se partagent les services d'un serveur leurs offrant un service strict de courbe $\beta(t) = 2.5t$. Le tableau III.4 montre pour chaque flux les paramètres associés (période, taille de paquet, courbe d'arrivée). Pour cet exemple, et en général pour les configurations applicables sur un bus CAN, [51] donne le vrai délai pire cas.

Le tableau III.5 nous donne les valeurs des χ_i associées aux flux R_2 et R_3. Ces valeurs permettent de tracer les courbes de service résiduel de ces flux. Sur la figure III.4, on peut voir que la borne sur le délai subit par le flux R_2, qui est la déviation horizontale qui existe entre sa courbe d'arrivée $\alpha_2 = \alpha_3$ et sa courbe de service résiduel β'_2 est de 3. On constate aussi de la même manière que la borne sur le délai subi par le flux R_3, résultant de la déviation horizontale entre α_3 et β'_3 est 3,5. Sur cet exemple, notre méthode permet d'atteindre le vrai délai pire cas, comme le montre le tableau III.4.

1.3.2 Comparaison 100 000 exemples aléatoires

Nous avons aussi effectué une comparaison sur 100 000 autres exemples. Nous avons considéré un système de n flux ($n \in [2, 10]$) périodiques R_i ($i \in [1, n]$). Chaque flux R_i possède une période T_i, émet des messages sous forme de paquets de taille s_i et est contraint par une

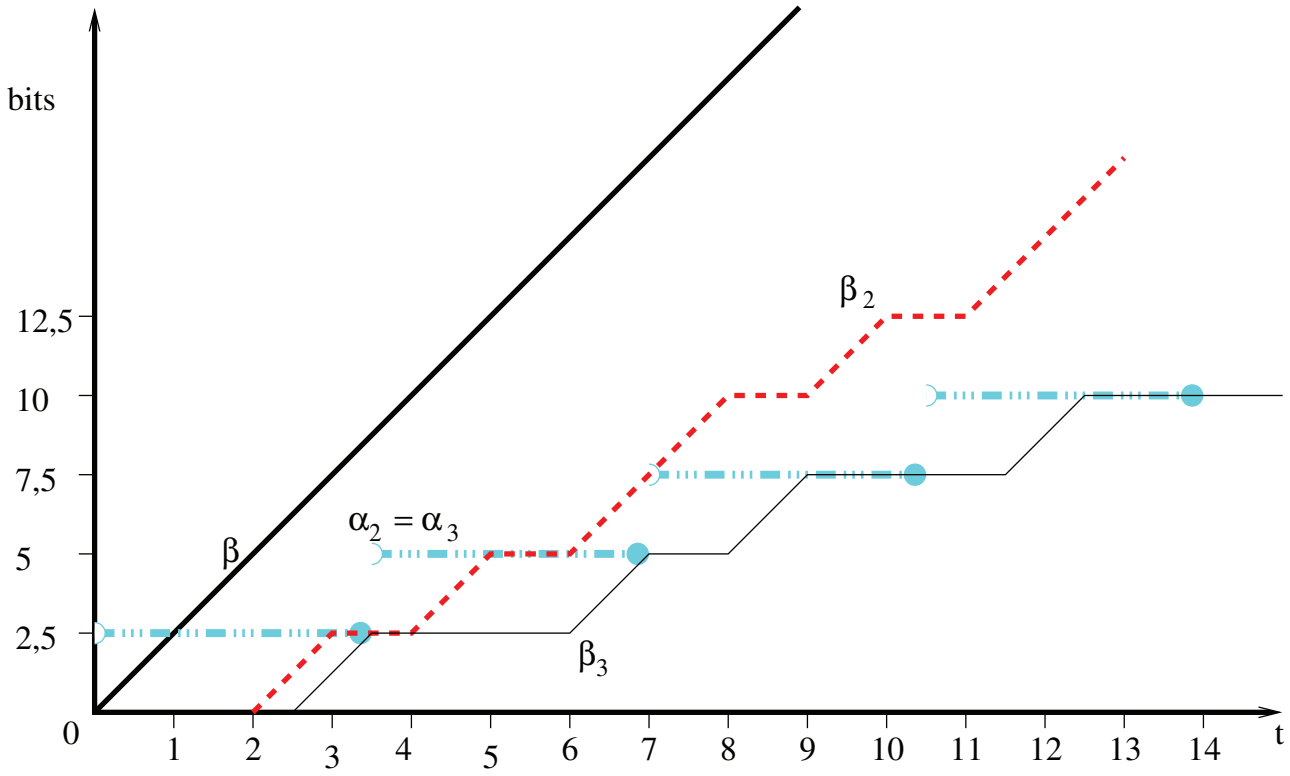


Figure III.4 – Courbes de service résiduel des flux R_2 (β_2) et R_3 (β_3) pour l'exemple CAN

i	Flux R_2	Flux R_3
1	2	5/2
2	4	6
3	6	9
4	7	25/2
5	9	16
6	11	37/2

Tableau III.5 – Valeurs des χ_i pour l'exemple CAN

Nb de flux	Nb de tests	Charge moyenne
2	11047	0.975178
3	10904	0.974754
4	11233	0.975173
5	11190	0.975123
7	11176	0.975112
6	10955	0.97506
8	11160	0.974798
9	11108	0.974796
10	11227	0.974967

Tableau III.6 – Configurations et charges classées par nombre de flux

courbe d'arrivée $\alpha_i = s_i \nu_{T_i,0}$. Le système offre un service strict de courbe $\beta(t) = t$. Pour des besoins de comparaison, la charge du système ϕ est aléatoirement choisie de façon uniforme entre 95% et 100%. Ensuite, le nombre de flux est aussi généré uniformément de façon aléatoire dans l'intervalle $[2, 10]$. Pour chaque flux i, un coefficient ϕ_i est généré aléatoirement entre 1 and 10. La charge totale du flux i, ρ_i, est alors donnée par la relation $\rho_i = \phi \frac{\phi_i}{\sum \phi_i}$. La taille de paquet s_i et la période T_i d'un flux R_i sont choisis de telle sorte que l'expression $\rho_i = \frac{s_i}{T_i}$ soit une fraction irréductible. Le calcul intégrant des opérations de l'algèbre (min,plus) s'est fait à l'aide de la librairie RT@W-PEGASE [32]. Il s'agit d'un outil d'aide à l'analyse et à l'optimisation des réseaux Ethernet commutés. Cette librairie nous a permis de réaliser les mesures des bornes supérieures sur le délai de communication.

10^5 configurations ont été générées, toutes avec une une charge importante pour le serveur (un résumé est donné dans le tableau III.6 et la figure III.5). Pour toutes ces configurations, nous obtenons le vrai délai pire cas, le même qui est calculé avec les résultats de [51].

De telles expérimentations ne donnent pas de preuve sur l'exactitude des résultats, mais nous permettent de nous conforter dans l'intuition selon laquelle nos résultats pourraient être exacts, au moins lorsqu'on se restreint au cas des messages périodiques.

1.4 Bilan de l'approche

Une question importante est la précision des résultats : le calcul réseau est une théorie qui calcule des bornes supérieures pour les délais et consommation mémoire. Cette borne est toujours correcte, mais peuvent parfois être arbitrairement éloignée du véritable pire cas [116].

La question de la précision est nommée "thightness" en anglais. Dans cette thèse, nous utiliserons le mot "exactitude", malgré son ambiguïté avec "correction". L'exactitude ici ne

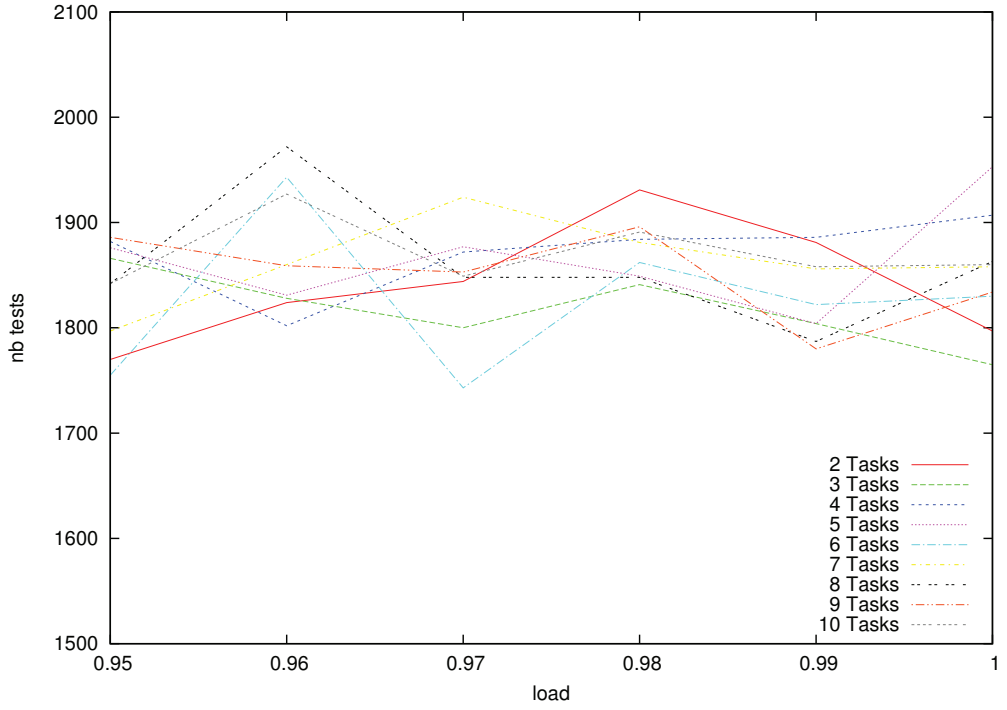


Figure III.5 – Distribution des charges

veut pas dire que le résultat soit erroné ou pas, mais plutôt qu'il soit impossible d'obtenir un meilleur résultat. Du point de vu du calcul réseau, un service résiduel est dit "exact" s'il permet d'obtenir des bornes atteignables et non pas une sur-approximation.

Lors de l'évaluation du service résiduel dans un contexte d'agrégation, la nature du service est cruciale : certains résultats nécessitent l'hypothèse de service strict, d'autres se contentent d'un service simple. En fait, jusqu'à présent, la plupart des résultats ont besoin d'un service strict, et la politique FIFO est la seule qui ne requiert qu'un service simple. Le service résiduel peut aussi être simple ou strict. Ce point est très important dans le cas d'une agrégation concernant plus de deux flux. En effet, contrairement à un service résiduel simple, un service résiduel strict peut être partagée par d'autres flux agrégés.

Nous avons proposé ici de nouvelles courbes de service résiduel dans le contexte de priorité non préemptive. Une étude comparative montre que nos travaux sont aussi précis que ceux qui existent déjà et qu'en plus ils généralisent un certain nombre de résultats précédents [26, 61, 42, 51] : ces travaux prennent en compte n'importe quelle contrat de trafic (courbe d'arrivée), et non pas seulement des flux périodiques ou sporadiques. Ces résultats ne se limitent pas seulement aux courbes de services définies par des fonctions affines, mais supportent la prise en compte de n'importe quelle type de courbe de service. De plus, avec nos travaux, nous fournissons un

service résiduel qui est strict, ce qui constitue un détail technique très important en calcul réseau. La garantie qu'offre un service résiduel strict est celle de pouvoir le partager à nouveau par plusieurs flux ; de plus, avec une politique de priorité qui peut être différente.

2 - GPS/DRR

GPS est une politique idéaliste, conçue explicitement pour permettre un partage équitable des ressources (déjà présentée, page 20). Avec cette politique, chaque flux R_i reçoit une fraction $\rho_i\beta$ du service global, représenté par une courbe β (avec $\sum_i \rho_i = 1$).

Comme cela est mentionné dans [106], le problème avec cette politique est qu'il s'agit d'une approche utopiste ne prenant pas en compte la transmission des messages sous forme de paquets. Tout doit se passer avec cette politique comme si le serveur pouvait servir plusieurs flux simultanément et que le trafic était d'une granularité infiniment fine.

La politique P-GPS (déjà présentée, page 21, et qui sera détaillée au paragraphe III.2.2) est une approximation de GPS qui offre les mêmes garanties, à un paquet près, que la politique GPS dans un contexte de transmission par paquets. Cependant, l'implémentation de la politique P-GPS est relativement complexe. Cette complexité fait en sorte que d'autres approches d'approximation de la politique GPS sont préférées à la politique P-GPS. L'algorithme DRR (Deficit Round Robin, rapidement présenté) est une implémentation pratique du paradigme GPS. Avec la politique DRR, chaque flux R_i reçoit un crédit Q_i, et la politique DRR permet de garantir à chaque flux une fraction $\rho_i = \frac{Q_i}{\sum_{j=1}^{n} Q_j}$ du service total. DRR se compte parmi les approximations de GPS les plus utilisées du fait de sa faible complexité de mise en œuvre ($O(1)$) et sa facilité d'implémentation (comme l'implémentation de [86]). En contrepartie, la latence qu'on obtient avec la politique DRR est plus élevée qu'avec la politique P-GPS. De ce fait, cette latence doit être évaluée avec précautions.

2.1 Généralisation de la définition de GPS

Les travaux traitant de la politique GPS considèrent que le serveur partagé fonctionne à une vitesse constante ($\beta = \lambda_r$). Cette définition peut se généraliser à n'importe quelle type de courbe de service β dans le contexte du calcul réseau : la généralisation de la politique GPS consiste juste à partager le service β de telle sorte que chaque flux reçoive un service de courbe $\rho_i\beta$.

À l'opposé, la politique DRR ne fait aucune allusion sur le type de courbe de courbe de service offert par la serveur. L'étude des travaux d'analyse de performance sur les politiques

DRR montre que ces travaux supposent tous que le serveur partagé émet à une vitesse constante. Nous allons généraliser cette étude à tout type de service.

Théorème 10 (GPS : service résiduel)**.** *Étant donné S, un serveur offrant un service strict de courbe β à n flux $R_1, \ldots, R_n$, selon la politique GPS de paramètres (non nuls) $\phi_1, \ldots, \phi_n$. Ce serveur offre à chacun des flux R_i un service strict de courbe β_i^{gps} donné par l'expression :*

$$\beta_i^{gps} = \frac{\phi_i}{\sum_{j=1}^{n} \phi_j} \beta \tag{III.33}$$

Démonstration. La preuve ressemble à celle de [106], exceptée qu'elle tient compte de n'importe quelle courbe de service β, et ne se limita pas seulement à la courbe $\beta(t) = rt$.

Soit $[t, s[$ une période de backlog du flux R_i, nous avons :

$$\begin{aligned} (R'_i(s) - R'_i(t))\frac{\phi_j}{\phi_i} &\geq R'_j(s) - R'_j(t) \\ \Longrightarrow \sum_{j=1}^{n} \frac{(R'_i(s) - R'_i(t))\phi_j}{\phi_i} &\geq \sum_{j=1}^{n} R'_j(s) - R'_j(t) \end{aligned}$$

et puisque S offre un service strict de courbe β, et que $[t, s[$ est une période de backlog, $\sum_{j=1}^{n} R'_j(s) - R'_j(t) \geq \beta(t)$

$$\Longrightarrow (R'_i(s) - R'_i(t))\frac{\sum_{j=1}^{n} \phi_j}{\phi_i} \geq \beta(t)$$

□

2.2 DRR : une approximation de la politique GPS

L'algorithme DRR a été rapidement présenté au paragraphe I.2.5.3, page 21. L'algorithme exact est présenté page 96.

Une boucle infinie parcours l'ensemble des files. Cet algorithme maintient un compteur `DC[`i`]` associé à chaque flux. Un flux actif (c.-à-d. avec une file non vide) reçoit un crédit Q_i à chaque itération de la boucle. Chaque émission de paquet diminue le compteur d'une valeur équivalente à la taille du paquet émis. On suppose que l'émission est bloquante, c.-à-d. que l'instruction suivante n'est exécutée que lorsque le dernier bit du paquet a été émis (avec un serveur à débit constant R, ce temps est l/R si l est la taille du paquet, mais c'est plus compliqué avec une fonction de service générale β).

Une légère modification a été apportée à la version originale de [121] : la boucle infinie scanne systématiquement toutes les files alors que l'algorithme initial garde une liste des files actives et ne visite que celles-ci.

Pour des besoins de lisibilité de la preuve, une pseudo instruction d'affichage (print) a été ajoutée. Cette instruction dont nous supposons l'exécution en un temps zéro, affiche le temps courant (now()) et le flux courant i sélectionné. Bien entendu, cette instruction ne fait pas parti de l'implémentation réelle.

Nous rappelons aussi qu'aucune supposition n'est faite quant à la politique de gestion de priorité à l'intérieure d'une file d'attente. La seule restriction est que la tête de la file d'attente est la même de la ligne ?? à la ligne ?? lors d'une itération donnée de l'algorithme 1. La tête de file d'attente peut changer d'une itération à une autre, si par exemple, il arrive un paquet de forte priorité entre deux itérations.

```
Input: Quantum par flux : Q_1..Q_n (Entier)
Data: Crédit de transmission par flux : DC[1..n] (Entier)
Data: Counter : k (Entier)
for i= 1 to n do
    DC[i] ← 0 ;
end
while True do
    for i= 1 to n do
        if not empty(i) then
            // Print est une pseudo instruction, utilisée pour simplifier la
               preuve
            print (now(), i) ;
            DC[i] ← DC[i] + Q_i ;
            while (not empty(i)) and (size(head(i)) ≤ DC[i]) do
                send(head(i)) ;
                DC[i] ← DC[i] − size(head(i)) ;
                removeHead( i ) ;
            end
            if empty(i) then
                DC[i] ← 0
            end
        end
    end
end
```

Algorithm 1: algorithme DRR

Modélisation de la politique DRR Nous allons faire quelques suppositions dans la preuve : précisément, nous supposons que

- L'instruction d'envois (ligne ??) est bloquante, l'exécution suivante n'est exécutée que quand le message est envoyé,
- Seule l'instruction d'envoi prend du temps,
- L'instruction d'envoi a toujours une durée non nulle,
- La même file d'attente ne peut être pas considéré comme vide et non vide à la même date, c.-à-d. si une file d'attente i est jugée vide à la ligne ??, alors elle ne peut pas être considérée comme non vide à la même itération à la ligne ??.

Service résiduel

Théorème 11 (Service résiduel DRR)**.** *Soit S un serveur offrant un service strict de courbe β, partagé par n flux $R_1, \ldots, R_n$. Chaque flux R_i émet des paquets de taille maximale l_i^m, et possède un quantum Q_i. Dans ces conditions, S offre à chaque flux R_i un service résiduel strict de courbe β_i^{DRR} définie par :*

$$\beta_i^{DRR}(t) = \left[\frac{Q_i}{F}\beta(t) - \frac{Q_i(L - l_i^m) + (F - Q_i)(Q_i + l_i^m)}{F}\right]^+ \tag{III.34}$$

avec $F = \sum_{i=1}^n Q_i$, $L = \sum_{i=1}^n l_i^m$.

De plus, si chaque taille de paquet est multiple d'une unité de base, et si en plus toutes les valeurs des Q_i sont multiples cette unité de base, alors le service résiduel peut être redéfini de la manière suivante :

$$\beta_i^{\epsilon\text{-}DRR}(t) = \left[\frac{Q_i}{F}\beta(t) - \frac{Q_i(L^\epsilon - l_i^{m\text{-}\epsilon}) + (F - Q_i)(Q_i + l_i^{m\text{-}\epsilon})}{F}\right]^+ \tag{III.35}$$

avec $l_i^{m\text{-}\epsilon} = l_i^m - \epsilon$ and $L^\epsilon = \sum_{i=1}^n l_i^{m\text{-}\epsilon}$.

Dans la pratique, une unité de base existe toujours. sa valeur étant d'au moins $\epsilon = 1$ ou $\epsilon = 8$, selon que la taille est exprimée en bits ou en octets.

Néanmoins, il est parfois plus facile, d'un point de vue de la modélisation, de traiter des longueurs de paquets comme des variables continues. C'est la raison pour laquelle les deux courbes sont présentées.

Nous donnons ici la preuve complète du théorème ci-dessus avec tous les résultats et définitions intermédiaires. Pour simplifier les notations, si R est une fonction cumulative d'un flux, la notation $R(s,t)$ est utilisée à la place de $R(t) - R(s)$.

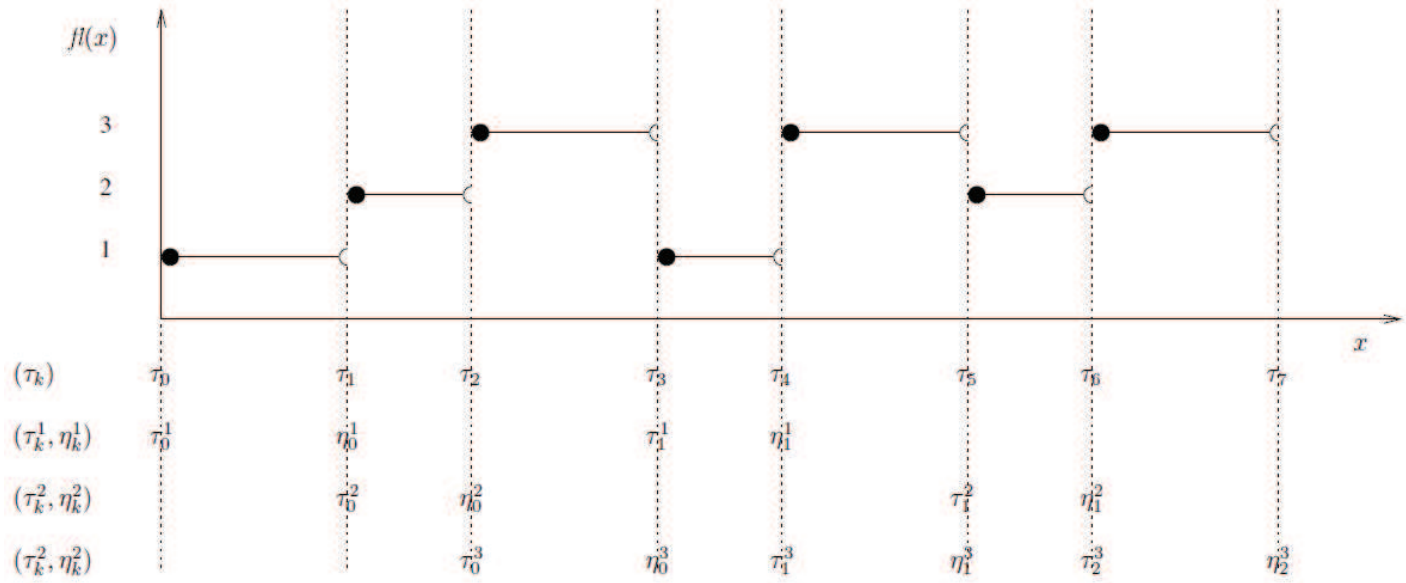


Figure III.6 – Illustration de $fl(x)$, τ_j^i, η_j^i

Le reste de la preuve suppose que $\beta \neq 0$ (sinon, $\beta_i^{\text{DRR}} = \beta_i^{\epsilon\text{-}DRR}$ et le théorème 11 devient trivial).

Lemme 5 (La courbe de service croit infiniment)**.**

$$\beta \neq 0, \beta \text{ courbe de service strict} \implies \lim_{x \to +\infty} \beta(x) = +\infty \tag{III.36}$$

Démonstration. Si β est une courbe de service strict, alors elle est sup-additive [28]. $\beta \neq 0 \implies \exists x, \beta(x) > 0$ et, pour tout n, $\beta(nx) > nx$. □

Ce lemme est nécessaire pour montrer que pour tout instant s, si le $i^{ième}$ flux est dans une période de backlog, alors il sera servi dans le futur (à chaque tour de boucle, seul une quantité finie des autres flux est servie, et donc, le $i^{ième}$ flux finira par être servi).

Définition 13. Trace de l'algorithme DRR *Soit* $[u, v[$ *une période de backlog maximale. Soit* $(\tau_k, fl_k)_{k \in [0,N]}$ *la séquence de couples (instant, flux), affichées à la ligne* **??** *de l'algorithme. (*$u = \tau_0$*). La séquence est complétée par* $\tau_{N+1} = v$*. Remarquons que la séquence des* τ_k *est croissante (*$\tau_k < \tau_{k+1}$*), puisque l'instruction d'envoi est de durée non nulle (Section III.2.2).*

De cette séquence, trois types d'objet sont dérivés : la fonction $fl : [u, v[\mapsto [1, n]$ *qui retourne le flux courant qui bénéficie du service à chaque instant et pour chaque flux* i*, les séquences* τ_j^i *et* η_j^i*, décrivant le début et la fin de la* $j^{ième}$ *opportunité de service. Ces quantités sont visibles sur la figure III.6.*

La fonction $fl : \mathbb{R} \to [1, n]$ *est une projection de la séquence* fl_k *permettant de retourner le flux en cours de service à un moment donné.*

$$fl(t) \stackrel{\text{def}}{=} fl_k \text{ avec } k = \max\{k' \mid \tau_{k'} \leq t\}$$

Pour chaque flux i, τ_j^i *est le début de la* $j^{\text{ème}}$ *opportunité de service, c.-à-d.* τ_k *tel que* $fl_k = i$. *Formellement, si un instant d'une période de backlog* t *existe dans* $[u, v[$ *pour le flux* i $(R'(t) > R(t))$, *alors*

$$\tau_0^i \stackrel{\text{def}}{=} \min\{\tau_k \mid fl_k = i\}$$
$$\tau_j^i \stackrel{\text{def}}{=} \tau_{k_j^i} \text{ avec } k_j^i \text{ s.t. } j = \left|\left\{\tau_p \mid fl_p = i, p \leq k_j^i\right\}\right|$$

Étant donné un flux i, k_j^i *est une sous-séquence croissante de* $\mathbb{N}$. *Les* η_j^i *instants marquent les fins d'opportunité de service :*

$$\eta_j^i \stackrel{\text{def}}{=} \tau_{k+1} \text{ with } \tau_j^i = \tau_k$$

Les instants τ_i^j définis ici sont les mêmes instants que τ_k^i définis dans [71], i étant l'index sur le flux, et j, k une séquence d'index.

Le modèle suppose que, sur l'intervalle $[\tau_i, \tau_{i+1}[$, le flux fl_i est servi et aucun autre ne l'est.

$$\forall \tau_k, \forall j \neq fl_k, \forall t \in [\tau_k, \tau_{k+1}[: R'_j(\tau_k, t) = 0 \tag{III.37}$$

Lemme 6 (Bornes sur le compteur DC)**.** *Soit* $[u, v[$ *une période de backlog maximale, et* τ_k, fl_k, τ_k^i *comme présentés dans la définition 13. Soit* $DC(i, \tau_k)$ *la valeur de* `DC`$[i]$ *quand le code passe à la ligne* **??** *à l'instant* τ_k. *Alors, les propriétés suivantes sont vérifiées [121].*

$$\forall \tau_j^i : 0 \leq DC(i, \tau_j^i) < l_i^m \tag{III.38}$$
$$\forall \tau_j^i : 0 \leq DC(i, \tau_j^i) \leq l_i^m - \epsilon \tag{III.39}$$
$$\forall \tau_j^i : DC(i, \tau_{j+1}^i) - DC(i, \tau_j^i) \leq Q_i \tag{III.40}$$
$$\begin{aligned} &fl(\eta_j^i) \neq i \implies \\ &\quad R'_i(\tau_j^i, \eta_j^i) \leq Q_i + DC(i, \tau_j^i) - DC(i, \tau_{j+1}^i) \end{aligned} \tag{III.41}$$

ce qui entraine

$$\begin{aligned} &\forall t \in [\eta^i_{j+p-1}, \tau^i_{j+p}) : \\ &\quad R'_i(\tau^i_j, t) \leq pQ_i + l^m_i - \epsilon < pQ_i + l^m_i \end{aligned} \tag{III.42}$$

On peut aussi s'intéresser aux périodes de backlog

$$\begin{aligned} &[\tau^i_k, \tau^i_{k+p}[\ i\text{-backlog period} \ \Rightarrow \\ &\quad R'_i(\tau^i_k, \tau^i_{k+p}) \geq pQ_i - l^m_i - \epsilon > pQ_i - l^m_i \end{aligned} \tag{III.43}$$

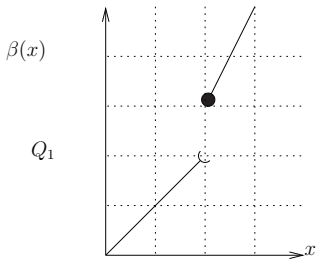
Remarque concernant les équations III.41 et III.42 Les équations III.41 et III.42 doivent satisfaire à la précondition $fl(\eta^i_j) \neq i$, ce qui signifie que cette propriété n'est vérifiée que s'il n'existe pas deux services consécutifs pour le même flux (ce qui revient à dire que le flux i est le seul à être actif). Il s'agit d'une faiblesse de la preuve. Comme il est évident que lorsque qu'un flux est seul actif, il reçoit le service β, il reçoit aussi β^{DRR}_i.

Cette condition est relative à la continuité de β à un instant de prise de décision d'ordonnancement. L'équation III.41 veut signifier que sur un intervalle de service $[\tau^i_j, \eta^i_j]$, le flux R'_i reçoit au plus le service $Q_i + DC(i, \tau^i_j) - DC(i, \tau^i_{j+1})$.

Le souci lorsque le flux i est le seul à être actif, c'est que la fin d'opportunité de service η^i_j est égale au début de l'opportunité suivante τ^i_{j+1}. Et en un temps nul, si la fonction de service est discontinue, on peut servir une quantité non nulle de données. Dès lors, la quantité $R'(\tau^i_{j+1})$ n'est pas uniquement la valeur à la fin de la précédente opportunité de service, mais aussi celle au début de la suivante. Et si la fonction de service est discontinue, cela peut être une valeur différente.

Illustrons le souci avec un exemple : considérons $l^m_1 = Q_1$, et une fonction β définie par l'equation III.44.

$$\beta(x) = \begin{cases} x & \text{if } x < Q_1 \\ 2x - \frac{Q_1}{2} & \text{if } x \geq Q_1 \end{cases} \tag{III.44}$$



Intéressons nous à la situation suivante : aux la date s et t, deux trames de taille l^m_1 issues du premier flux arrive au niveau du serveur et sont mises dans la première file d'attente (avec $t - s < Q_1$). Supposons que la sortie du serveur soit exactement β. Le scénario est illustré sur la figure, où pour être plus lisible, nous avons pris pour l'axe des x un pas deux fois plus grand

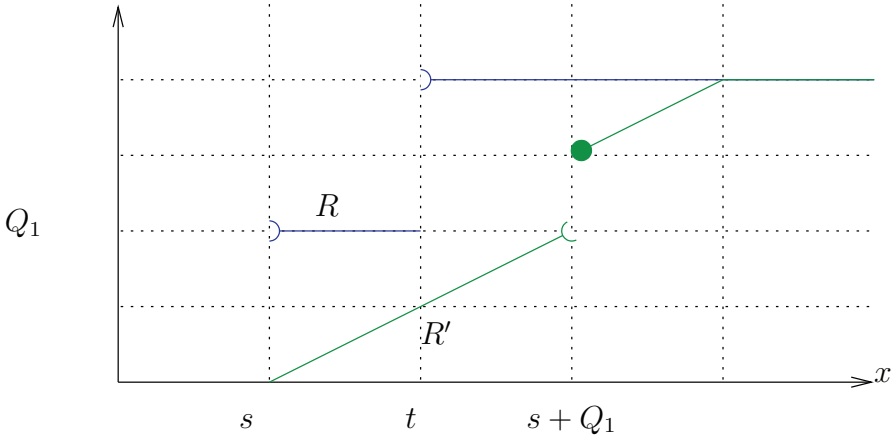


Figure III.7 – Deux services successifs avec fonction discontinue

que l'axe des y.

À la première itération, à l'instant $\eta_0^1 = s$, $\texttt{DC}[1] = 0$, et $DC(\tau_0^1) = 0$. Le premier paquet est envoyé entre les dates s et $s + Q_i$ ($\tau_0^1 = s, \eta_0^1 = s + Q_1$), et le compteur DC est décrémenté de la taille du paquet transmis c.-à-d. Q_1. La seconde itération commence avec $DC(1, \tau_1^1) = 0$. Mais à cet instant, $\tau_1^1 = Q_i$ le serveur commence à servir le deuxième paquet, et transmet la moitié de ce paquet instantanément. Dès lors, $R'(\tau_0^1, \eta_0^1) = \frac{3}{2}Q_i > Q_i + DC(\tau_0^1) - DC(\tau_1^1)$.

Pour faire une preuve qui prenne aussi en compte deux sélections successive du même flux, il aurait fallu distinguer $\lim_{x \xrightarrow{\eta} j^i, x<\eta_j^i} R'_i(x)$ (souvent noté $R'(\eta_j^i-)$) de $R'(\tau_{j+1}^i)$. Nous aurions aussi pu introduire un temps logique à l'intérieur du temps physique, pour modéliser la notion d'actions successives simultanées.

La condition $fl(\eta_j^i) \neq i$, signifiant que la propriété ne s'applique que si le même flux n'est pas servi deux fois dans une séquence, est suffisante pour la preuve, et nous semble une limitation acceptable.

Preuve de Eq. III.38–III.39. La variable $\texttt{DC}[i]$ est initialisée à 0, et décrémentée dans une seule ligne de l'algorithme. Avant la décrémentation de la variable $\texttt{DC}[i]$ de la quantité size(head(i)), la relation size(head(i)) $\leq \texttt{DC}[i]$ est vérifiée, ce qui permet d'assurer la propriété $\texttt{DC}[i] \geq 0$.

À l'inverse, $\texttt{DC}[i]$ n'est incrémenté que si la file d'attente n'est pas vide. Et dans ce cas, tant que $\texttt{DC}[i] \geq size(head(i))$ alors la variable $\texttt{DC}[i]$ est décrémentée. En fin de boucle, $\texttt{DC}[i] < size(head(i)) \leq l_i^m$, ce qui donne Eq. III.38.

La preuve de la borne $\texttt{DC}[i] \leq l_i^m - \epsilon$ nécessite un résultat intermédiaire : $\texttt{DC}[i]$ est toujours un multiple de ϵ.

L'affirmation ci-dessus est vraie à l'étape initiale (lignes ?? et ??), et préservée chaque fois que la variable $DC[i]$ est incrémentée de Q_i ($\texttt{DC}[i] \leftarrow \texttt{DC}[i] + Q_i$) – puisque Q_i est aussi un multiple de ϵ. Il en est de même quand la variable $\texttt{DC}[i]$ est décrémentée ($\texttt{DC}[i] \leftarrow \texttt{DC}[i] -$ size(head(i))) – puisque les tailles de paquet sont des multiples de ϵ. Finalement, du fait que $\texttt{DC}[i] < l_i^m$, et que $\texttt{DC}[i]$ et l_i^m soient tous multiples de ϵ, il s'ensuit que $\texttt{DC}[i] \leq l_i^m - \epsilon$. □

Preuve de l'équation III.40. $\texttt{DC}[i]$ n'est incrémenté qu'une seule fois par itération et la valeur de l'incrément est Q_i. □

Preuve de l'équation III.41. Introduisons $DC'(i, \tau_k^i)$, la valeur de $\texttt{DC}[i]$ entre les lignes ?? and ?? ($DC'(i, \tau_k^i) \geq 0$). Du fait que la variable $\texttt{DC}[i]$ est décrémentée d'une valeur correspondant à la quantité de données envoyée, il s'en suit que :

$$R_i'(\tau_j^i, \eta_j^i) = Q_i + DC(i, \tau_j^i) - DC'(i, \tau_j^i) \tag{III.45}$$

Et $DC(i, \tau_{j+1}^i) = DC'(i, \tau_j^i)$ ou $DC(i, \tau_{j+1}^i) = 0$ en fonction du résultat du test de la ligne ??. Ce qui donne $DC(i, \tau_{j+1}^i) \leq DC'(i, \tau_j^i)$. □

Preuve de l'équation III.42. C'est une conséquence directe de l'équation III.41, qui donne une borne sur la période de service ; l'équation III.37, qui permet de garantir l'absence de de sortie de données entre deux périodes de services consécutives et l'équation III.39 qui donne une borne sur les valeurs de $DC(\cdot, \cdot)$.

$$\begin{aligned} R_i'(\tau_j^i, t) &= \sum_{k=0}^{p-1} R_i'(\tau_{j+k}^i, \tau_{j+k+1}^i) \\ &\leq \sum_{k=0}^{p-1} (Q_i + DC(i, \tau_{j+k}^i) - DC(i, \tau_{j+k-1}^i)) \\ &= (\sum_{k=0}^{p-1} Q_i) + DC(i, \tau_j^i) - DC(i, \tau_{j+k-1}^i) \\ &\leq pQ_i + l_i^m - \epsilon \end{aligned}$$

□

Preuve de l'équation III.43. La même décomposition que celle de la preuve précédente est faite ici. La différence réside dans l'utilisation de l'équation III.45 au lieu de l'équation III.41. Nous utilisons aussi le fait que dans une période de backlog d'un flux i, $DC'(i, \tau_j^i) = DC(i, \tau_{j+1}^i)$. Et

donc, la somme $\sum_{k=j}^{j+p-1}$ of eq. III.45 conduit à $R'_i(\tau^i_j, \tau^i_{j+p}) = pQ_i + DC(i, \tau^i_j) - DC'(i, \tau^i_{j+p-1})$, et avec l'équation III.39 les deux expressions DC et DC' peuvent être supprimées. □

Lemme 7 (Nombre de cycles)**.** *Soit s un instant d'une période de backlog d'un flux i, dans une période maximale de backlog $[u, v[$. Soit τ^i_k le début de la prochaine période de service ($\tau^i_k = \min\{\tau^i_m \mid \tau^i_m \geq s\}$). Chacun des autres flux a bénéficié du service au plus $p+1$ fois entre s et τ^i_{k+p}.*

$$\forall j \neq i, \left|\left\{\tau^j_m \mid s \leq \tau^j_m < \tau^i_{k+p}\right\}\right| \leq p+1 \tag{III.46}$$

Démonstration. Comme les instructions de la boucle sont toujours exécutées dans le même ordre, et puisque $[s, \tau^i_{k+p}[$ est une période d backlog pour R_i, la condition de la ligne **??** est toujours vraie. Par conséquent, chaque flux est sélectionné au plus une seule fois dans l'intervalle $[s, \tau^i_k[$, il y a p itérations de boucle entre τ^i_k et τ^i_{k+p} et chacun des autres flux n'est sélectionné qu'une seule fois par itération. □

Démonstration. (du théorème 11) Soit i un flux avec R_i comme fonction d'entrée et R'_i sa fonction de sortie, soient $s \leq t$ deux instants de sa période de backlog. Soit $[u, v[$ sa période de backlog maximale qui inclut s et t. considérons la séquence τ_i, fl_i τ^j_i, avec les significations de la définition 13

Soit τ^i_k le début de la première opportunité de service du flux i après la date s (Il y en a certainement une puisque s marque une période de backlog de i et qu'avec la politique DRR, le serveur transmet les données tant qu'il y en a). Soit $\mathcal{P}$, l'ensemble constitué de toutes les opportunités de service du flux i entre s et t, et soit p sa cardinalité (s'il n'y a pas d'opportunité de service entre s et t, alors $\mathcal{P} = \emptyset$ et $p = 0$).

$$\begin{aligned}
\tau^i_k &\stackrel{\text{def}}{=} \min\left\{\tau^i_m \mid \tau^i_m \geq s\right\} \\
\mathcal{P} &\stackrel{\text{def}}{=} \{\tau_m \mid s \leq \tau_m, \tau_{m+1} \leq t, fl_m = i\} \\
&= \left\{\tau^i_j \mid s \leq \tau^i_j, \eta^i_j \leq t\right\} \\
p &= |\mathcal{P}|
\end{aligned}$$

Étape 1 $R'_i(s,t) \geq pQ_i - l^m - \epsilon$

Supposons que $p > 0$ (le cas $p = 0$ sera traité plus tard). Dans ce cas, $\mathcal{P} = \left\{\tau^i_k, \ldots, \tau^i_{k+p-1}\right\}$. Puisque $\tau^i_k \geq s$, et $R'_i \in \mathcal{F}$ est non décroissant, nous avons

$$R'_i(\tau^i_k) \geq R'_i(s) \tag{III.47}$$

Considérons τ^i_{k+p}.

Si t est dans une période de service de i ($fl(t) = i$), alors $\tau^i_{k+p} \leq t$ ($fl(t) = i$ implique qu'il existe k' tel que $\tau^i_{k'} \leq t < \eta^i_{k'}$, d'où $\tau^i_{k'} \notin \mathcal{P}$ et donc $k' > k + p - 1$, c.-à-d. $k \geq k + p$, et comme τ^i est croissant, $t \geq \tau^i_{k+p}$) $R'_i(\tau^i_{k+p}) \leq R'_i(t)$. Si t n'est pas dans une période de service du flux i, $\tau^i_{k+p} > t$ et donc $R'_i(\tau_{k+p}) = R'_i(t)$ (par application successive de l'équation III.37). Finalement, dans les deux cas on a bien :

$$R'_i(\tau^i_{k+p}) \leq R'_i(t) \tag{III.48}$$

On obtient alors $R'_i(s,t) = R'_i(t) - R'_i(s) \geq R'_i(\tau^i_{k+p}) - R'_i(\tau^i_k) \geq pQ_i - l^m_i - \epsilon$ de l'équation III.43 (les hypothèses de backlog $[\tau^i_k, \tau^i_{k+p}[$ viennent du fait que $[\tau^i_k, \tau^i_{k+p}[\subset [s,t[$ et $[s,t[$)

Si $p = 0$ alors, $pQ_i - l^m_i - \epsilon \leq 0 \leq R'_i(s,t)$. Ceci prouve l'étape 1 et donc fourni une borne supérieure pour $R'_i(s,t)$, vu comme une fonction de f p :

$$R'_i(s,t) \geq pQ_i - l^m_i - \epsilon \tag{III.49}$$

Nous devons maintenant trouver une borne inférieure de p basée sur $\beta(t-s)$.

Étape 2. $\frac{\beta(t-s) - (R'_i(s,t) + \sum_{j \neq i}(l^m_j - \epsilon))}{\sum_{j \neq i} Q_j} \leq p + 1$

Puisque le serveur offre un service strict, nous avons la relation suivante :

$$\beta(t-s) \leq R'(s,t) = R'_i(t) - R'_i(s) + \sum_{j \neq i} R'_j(t) - R'_j(s)$$

Si t est dans une période de service de i, $\tau^i_{k+p} \leq t$, cependant, pour tout $j \neq i$, $R'_j(\tau^i_{k+p}) = R'_j(t)$ (par application successives de l'équation. III.37). Sinon, $\tau^i_{k+p} \geq t$ et $R'_j(\tau^i_{k+p}) \geq R'_j(t)$. Soit dans les deux cas, $R'_j(\tau^i_{k+p}) \geq R'_j(t)$.

$$\beta(t-s) \leq R'_i(t) - R'_i(s) + \sum_{j \neq i} R'_j(\tau^i_{k+p}) - R'_j(s)$$

D'après le lemme 7, il y a au plus $p + 1$ cycles pour chaque flux R_j entre s et $R'_j(\tau^i_{k-1})$, c.-à-d. $R'_j(\tau^i_{k+p}) - R'_j(s) \leq (p+1)Q_j + l^m_j - \epsilon$ (en utilisant l'équation III.42).

Une simple somme conduit à une borne inférieure sur de p exprimée en fonction de β, Q_j, l^m_j

et $R'_i(s,t)$.

$$\beta(t-s) \leq R'_i(s,t) + \sum_{j\neq i}\Big((p+1)Q_j + l_j^m - \epsilon\Big) \tag{III.50}$$

$$= R'_i(s,t) + (p+1)\sum_{j\neq i} Q_j + \sum_{j\neq i}\Big(l_j^m - \epsilon\Big) \tag{III.51}$$

Le reste nécessite juste des manipulations algébriques simples des équations III.49 et III.50.

Pour des raisons de simplification, nous notons $F = \sum_{j=1}^{n} Q_j$, $l_i^{m\text{-}\epsilon} = l_i^m - \epsilon$, et $L^\epsilon = \sum_{j=1}^{n} l_i^{m\text{-}\epsilon}$. Et donc, $\sum_{j\neq i} Q_i = F - Q_i$, et $\sum_{j\neq i} l_j^m - \epsilon = L^\epsilon - l_i^{m\text{-}\epsilon}$.

$$\begin{aligned} \textit{eq. III.50} \iff \beta(t-s) - (R'_i(s,t) + (L^\epsilon - l_i^{m\text{-}\epsilon}) &\leq (p+1)(F-Q_i) \\ \frac{\beta(t-s) - R'_i(s,t) - (L^\epsilon - l_i^{m\text{-}\epsilon}) - (F-Q_i)}{F-Q_i} &\leq p \end{aligned}$$

En introduisant ces bornes avec p dans l'équation III.49, on aboutit à :

$$\begin{aligned} R'_i(s,t) &\geq \frac{\beta(t-s) - R'_i(s,t) - (L^\epsilon - l_i^{m\text{-}\epsilon}) - (F-Q_i)}{F-Q_i} Q_i - l_i^m \\ &\iff (F-Q_i)R'_i(s,t) \geq Q_i\beta(t-s) - Q_i R'_i(s,t) \\ &\quad - Q_i\left((L^\epsilon - l_i^{m\text{-}\epsilon}) + (F-Q_i)\right) - (F-Q_i)l_i^{m\text{-}\epsilon} \\ &\iff F R'_i(s,t) \geq Q_i\beta(t-s) \\ &\quad - Q_i\left((L^\epsilon - l_i^{m\text{-}\epsilon}) + (F-Q_i)\right) - (F-Q_i)l_i^{m\text{-}\epsilon} \\ \iff R'_i(s,t) &\geq \frac{Q_i}{F}\beta(t-s) - \frac{Q_i(L^\epsilon - l_i^{m\text{-}\epsilon}) + (F-Q_i)(Q_i + l_i^{m\text{-}\epsilon})}{F} \end{aligned}$$

De plus, puisque $R'_i(t)$ est non décroissant, $R'_i(t) - R'_i(s) \geq 0$, et donc, l'opérateur $[\cdot]^+$ peut être appliqué. □

Choix des paramètres Certains travaux [121, 71] ne considèrent qu'un seul paquet de taille maximale, qui est commun à tous les flux. Il apparaît cependant plus judicieux et raisonnable de considérer un paquet de taille maximale par flux (les flux avec des paquets de petite taille tel que la voix sur IP coexistent avec les flux dotés de paquets plus volumineux comme du contenu web). Cette considération permet d'obtenir des bornes plus fines [89, 87, 86].

L'unité de base ϵ a été introduit pour faciliter la comparaison avec [71]. Si les tailles de paquet et les variables Q_i sont discrètes, alors les intervalles à base de $DC[i]$ ou l_i^m peuvent

toujours être réduits. Par exemple, on peut écrire $[0, l_i^m - \epsilon + Q_i]$ et $[0, l_i^m - \epsilon]$ à la place de $[0, l_i^m + Q_i)$ et $[0, l_i^m)$ respectivement. Cela permet de faire un gain de ϵ sur la latence de chaque flux dès lors que le terme l_i^m apparaît. Le gain généré par une telle modélisation de la granularité de la taille des paquets en contexte discret est cependant négligeable dans certains cas pratique où ϵ n'est qu'un octet et l_i^m une centaine ou un millier d'octets. Pour des paquets de 100 octets, le gain représente un pour-cent.

Évaluation du pessimisme de l'approche Le pessimisme global de notre approche peut être évaluée en mesurant l'écart qui existe entre la courbe de service de départ et la somme de toutes les courbes de service résiduel.

Considérons un serveur S offrant un service strict de courbe β et partagé par n flux $(R_1, \ldots, R_n) \xrightarrow{S} (R'_1, \ldots, R'_n)$. Cela signifie que pour chaque période de backlog $[s, t[$:

$$\sum_{i=1}^{n} R'_i(t) - R'_i(s) \geq \beta(t - s) \tag{III.52}$$

Supposons pour chaque flux qu'un service résiduel β_i peut être obtenu, alors pour toute période de backlog $[s_i, t_i)$ du flux R_i

$$R'_i(t_i) - R'_i(s_i) \geq \beta_i(t_i - s_i) \tag{III.53}$$

Supposons maintenant que $[s, t[$ soit une période de backlog de chacun des flux R_i (ce qui peut être le cas par exemple si tous les flux émettent en même temps à la date s et que l'instant t est le premier auquel une des files d'attente est vide). Dans ce cas, on peut écrire les deux relations suivantes :

$$\sum_{i=1}^{n} R'_i(t) - R'_i(s) \geq \beta(t - s)$$
$$\sum_{i=1}^{n} R'_i(t) - R'_i(s) \geq \sum_{i=1}^{n} \beta_i(t - s)$$

Dans le cas du DRR, qui est une approximation de GPS, cette propriété $\sum_i \beta_i \leq \beta$ est vraie même en dehors des périodes de backlog. puisque la somme des services résiduel GPS est égale au service initial ($\sum_i \beta_i^{gps} = \beta$, cf. eq. III.33). et que le service résiduel DRR est plus petit que le service résiduel GPS ($\beta_i^{DRR} \leq \beta_i^{gps}$, cf. eq.).

Le pessimisme total induit par la décomposition en services résiduels DRR peut se définir

de la manière suivante :

$$\text{Pess} = \beta - \sum_{i=1}^{n} \beta_i \tag{III.54}$$

Le pessimisme du théorème 11 s'exprime alors comme suit :

$$\text{DRR-Pess} = \beta - \sum_{i=1}^{n} \beta_i^{\text{DRR}} \tag{III.55}$$

Pour simplifier les notations, β_i^{DRR} est utilisé à la place de $\beta_i^{\epsilon\text{-DRR}}$ (nous avons déjà montré que l'impact de ϵ est négligeable).

$$\begin{aligned}
&F(\beta - \sum_{i=1}^{n} \beta_i^{\text{DRR}}) \\
\geq & F\left(\beta - \sum_{i=1}^{n} \frac{Q_i}{F}\beta + \sum_{i=1}^{n} \frac{Q_i(L - l_i^m) + (F - Q_i)(Q_i + l_i^m)}{F}\right) \\
= & \sum_{i=1}^{n} (Q_i(L - l_i^m) + (F - Q_i)(Q_i + l_i^m)) \\
= & \sum_{i=1}^{n} (Q_i L - 2Q_i l_i^m + FQ_i + Fl_i^m - Q_i^2) \\
= & F^2 + 2FL - 2\sum_{i=1}^{n} (Q_i l_i^m) - \sum_{i=1}^{n} Q_i^2 \\
= & \left(\sum_{i=1}^{n} Q_i\right)^2 + 2\sum_{i=1}^{n} Q_i \sum_{i=1}^{n} l_i^m - 2\sum_{i=1}^{n} (Q_i l_i^m) - \sum_{i=1}^{n} Q_i^2 \\
= & \left(\sum_{i=1}^{n} Q_i + l_i^m\right)^2 - \sum_{i=1}^{n} (Q_i + l_i^m)^2 + \left(\sum_{i=1}^{n} l_i^m\right)^2 - \sum_{i=1}^{n} (l_i^m)^2
\end{aligned}$$

Et donc, nous obtenons

$$\begin{aligned}
&\text{DRR-Pess} \\
\geq & \frac{1}{F}\left[\left(\sum_{i=1}^{n} Q_i + l_i^m\right)^2 - \sum_{i=1}^{n} (Q_i + l_i^m)^2 + \left(\sum_{i=1}^{n} l_i^m\right)^2 - \sum_{i=1}^{n} (l_i^m)^2\right]^+
\end{aligned}$$

Puisque $Q_i \geq l_i^m$, ce terme peut être borné. Ce qui donne :

$$\text{DRR-Pess} \geq 3\frac{\left(\sum_{i=1}^{n} l_i^m\right)^2 - \sum_{i=1}^{n}(l_i^m)^2}{\left(\sum_{i=1}^{n} Q_i\right)} \tag{III.56}$$

Cette borne inférieure est atteinte lorsque $Q_i = l_i^m$. Si toutes les l_i^m ont la même valeur l^m, la relation devient :

$$\text{DRR-Pess} \geq nl^m - \frac{l^m}{n} \tag{III.57}$$

Ce résultat confirme l'intuition selon laquelle l'algorithme est en quelque sorte optimal lorsque $Q_i = l_i^m$, et montre que notre algorithme « perd » grosso-modo une trame par flux, ce qui engendre un pessimisme que croit linéairement avec le nombre de flux. Ce comportement était prévisible dans la mesure où nos courbes de service résiduel fournissent une mesure du service pire cas pour chacun des flux i.

3 - Intégration NP-SP avec DRR

3.1 Types de service

Dans le théorème 11, le service offert par le serveur partagé doit être strict, et le service résiduel offert à chaque flux est aussi strict. Le fait pour un service d'être strict est une propriété très importante : en général, les implémentations réelles offrent un service strict. Le challenge consiste à obtenir une courbe de service *résiduel* strict à partir du service initial offert par le serveur. Obtenir un service résiduel strict offre la possibilité de le partager entre plusieurs autres flux et donc d'en extraire aussi un service résiduel pour chacun des flux le partageant dans un contexte d'ordonnancement hiérarchiquement. Par exemple, une des files d'attente ordonnancée selon la politique DRR, peut être constituée de plusieurs autres flux ordonnancées avec une autre politique. Dans ce cas, puisqu'un service résiduel est assigné à chaque file, et comme de plus ce service résiduel est strict, il peut servir de base à l'évaluation des performances des flux constituant chaque file d'attente.

Par exemple, comme la latence induite par la politique de service DRR peut être importante, pour certains flux, les routeurs commerciaux utilise la politique d'ordonnancement hiérarchique NP-SP / DRR. Avec cette politique, les flux sont dans un premier temps agrégés en classes de service, avec un premier niveau d'ordonnancement conforme à la priorité statique (NP-SP). À l'intérieur de chaque classe de service, les flux sont ordonnancés conformément à la politique d'ordonnancement DRR. Pour pouvoir analyser les performances des flux composant chaque classe de service à l'aide de nos résultats, il est nécessaire que le service résiduel de la politique

	Priorité	Période	Taille	α_i	Délai	Q_i
R_1	1	6	2	$2 \times \lceil \frac{t}{6} \rceil$	5	-
$R_{2,1}$	2	18	3	$3 \times \lceil \frac{t}{18} \rceil$	35	1
$R_{2,2}$	2	18	3	$3 \lceil \frac{t}{18} \rceil$	22	2
R_3	3	8	2	$2 \times \lceil \frac{t}{8} \rceil$	12	-

Tableau III.7 – Paramètres et délais

	Priorité	α_i
R_1	1	$2 \times \lceil \frac{t}{6} \rceil$
R_2	2	$6 \times \lceil \frac{t}{18} \rceil$
R_3	3	$2 \times \lceil \frac{t}{8} \rceil$

Tableau III.8 – Paramètres agrégés

NP-SP soit strict ou faiblement strict. Par contre, la gestion d'une politique DRR/NP-SP nécessite d'avoir un service strict.

3.2 Exemple d'application

Cet exemple montre comment on peut gérer un ordonnancement hiérarchique. Considérons un serveur offrant un service strict de courbe $\beta(t) = t$ à quatre flux : R_1, R_{21}, R_{22} et R_3. Les caractéristiques de ces flux sont données dans le tableau III.2 (priorité, période et taille de paquet).

On peut remarquer que les flux R_{21} et R_{22} ont la même priorité. En cas de conflit entre les paquets de ces deux flux, on utilisera la politique DRR avec pour paramètres les données de la colonne Q_i du tableau III.7

3.2.1 Agrégation NP-SP

Avec ces informations, on peut en déduire des courbes d'arrivées auxquelles ces flux se conforment (colonne α_i du tableau III.2). Tout se passe alors comme si les flux R_{21} et R_{22} étaient agrégés et ne formaient qu'un seul et même flux. On aurait donc trois flux avec les caractéristiques du tableau III.8.

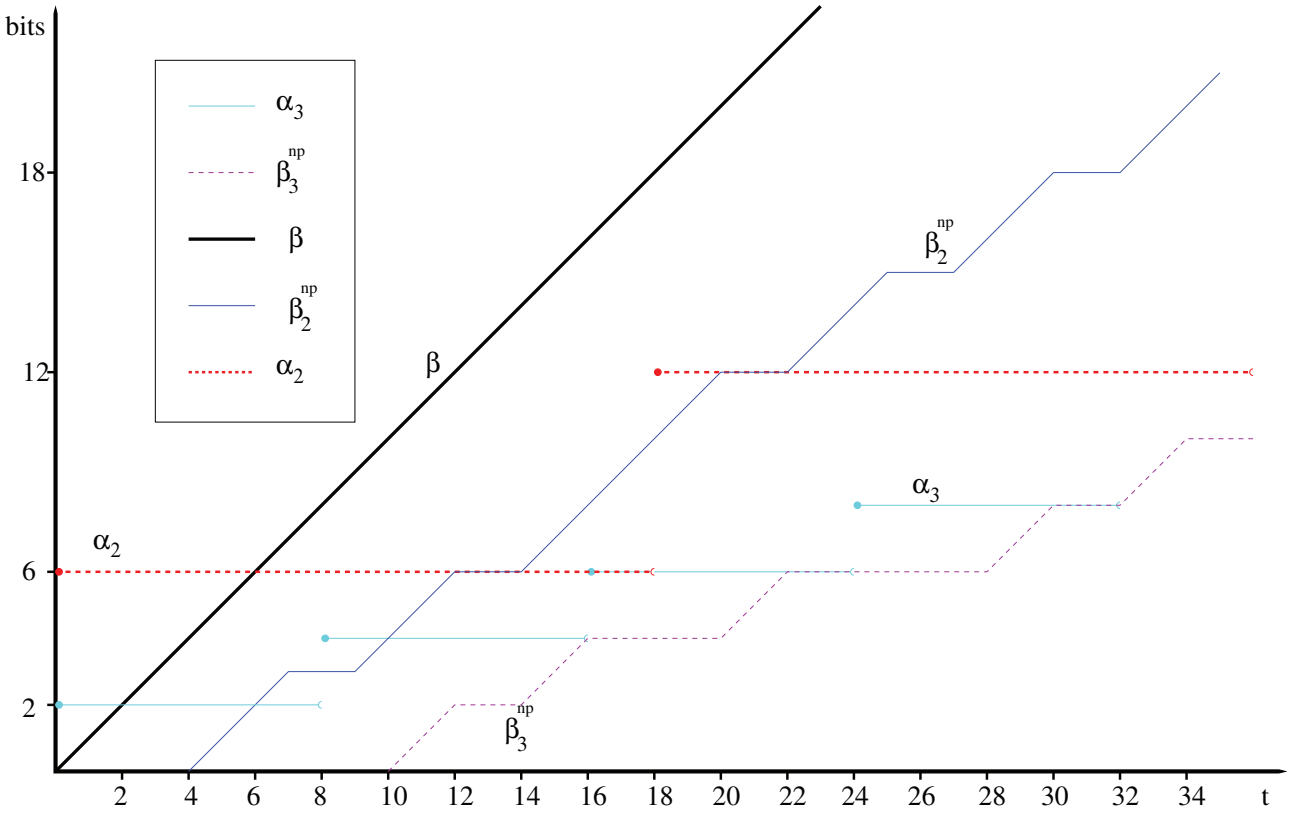


Figure III.8 – Valeurs des χ_i pour les courbes de service résiduel des flux R_2 (β_2) et R_3 (β_3)

i	χ_i pour R_2	χ_i pour R_3
1	4	10
2	9	14
3	14	20
4	17	28
5	22	32
6	27	38
7	32	46
8	35	50

Tableau III.9 – Valeurs de χ_i pour le calcul de β_2 et β_3

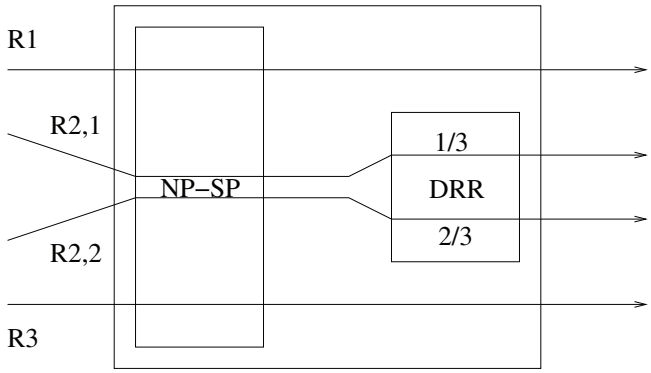


Figure III.9 – Illustration de l'agrégation

Cas du flux R_1 C'est le flux le plus prioritaire. Sa courbe de service résiduelle est $\beta_1^{np} = \beta - l_2 = t - 3$. Ce qui nous donne un délai de $h(\alpha_1, \beta_1^{np}) = 5$.

Cas du flux agrégé R_2 Pour le flux agrégé R_2, le calcul des valeurs de χ_i (tableau III.9) nous permet de tracer la courbe de service résiduel β_2 de R_2. La figure III.8 nous présente la courbe β_2 du service résiduel qui sera partagé entre les flux R_{21} et R_{22} dans paragraphe III.3.2.2.

Cas du flux R_3 De même que pour le flux agrégé R_2, les valeurs des χ_i nécessaires pour tracer la courbe de service résiduel β_3 du flux R_3 se trouvent dans le tableau III.9. La courbe β_3 est représentée dans la figure III.8. On peut aussi lire sur cette figure la borne sur le délai subi par ce flux. Elle est donnée par la déviation horizontale entre $\alpha 3$ et β_3 (h($\alpha 3$, β_3)=12).

3.2.2 Gestion des flux R_{21} et R_{22} avec DRR

Sur la figure III.10 nous avons repris la répartition du service résiduel β_2^{np} du flux R_2. En plus, nous avons ajouté les courbes de service résiduel des flux R_{21} et R_{22} ($\beta_{2,1}^{np/drr}$ pour R_{21} et $\beta_{2,2}^{np/drr}$ pour R_{22}). Dans cette figure, $\beta_{2,i}^{np/gps} = \frac{Q_i}{\sum_{i=1}^{n} Q_i} \beta_2^{np}$ est l'expression qui apparaît dans l'équation III.35 de telle sorte qu'on ait $\beta_{2,i}^{np/drr} = \beta_{2,i}^{np/gps} - 2$. On peut alors en déduire les différents délais de ces flux : $h(\alpha_{2,1}, \beta_{2,2}^{np/drr}) = 22$ pour le flux R_{22} ($\alpha_{2,1} = \alpha_{2,2}$). Les différents délais pour chaque flux se trouvent dans le tableau III.2 (colonne Délai).

4 - Conclusion

La première contribution de ce chapitre concerne la politique d'ordonnancement à priorité statique non préemptive (NP-SP). Nous avons amélioré l'ensemble des résultats dont nous

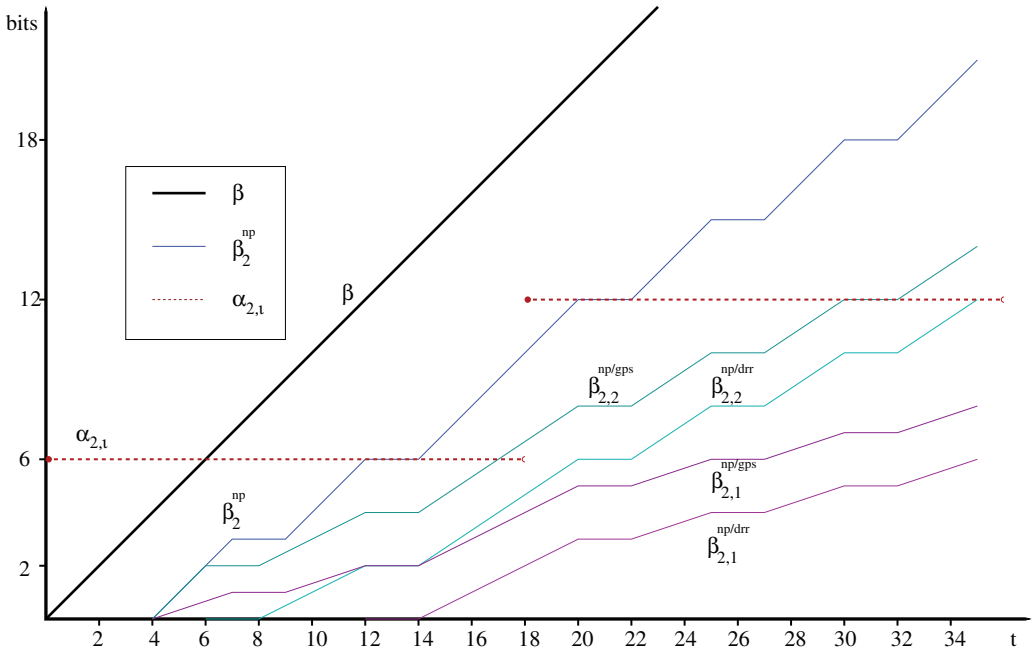


Figure III.10 – Services résiduels

avons connaissance à ce jour : nous généralisons les travaux précédents faits en calcul réseau, en offrant des résultats numériques aussi bons, et en calculant un service résiduel strict ou faiblement strict. Nous généralisons aussi les travaux sur l'ordonnancement de trames CAN, puisque nous obtenons des résultats aussi bon mais que nos hypothèses sont plus larges (nous n'avons pas à nous restreindre à des entrées périodiques et un débit constant). Il faut tout de même souligner deux limites : d'abord, notre travail ne s'applique qu'aux flux à paquets de taille fixe, et ensuite, nous n'avons pas de comparaison formelle entre nos travaux et les autres car nous calculons une formule analytique alors que les autres approches appliquent des algorithmes. Nous avons contourné le problème en comparant sur un très grand nombre de tests.

La seconde contribution concerne les politiques à partage équitable, GPS et DRR. Nous avons montré comment la politique GPS, initialement définie comme le partage d'un débit constant, peut se généraliser au partage de n'importe quel service strict. Comme cette politique ne peut pas être implémentée dans un ordonnanceur par paquet, nous avons étudié sa meilleure implantation, DRR. Nous avons définit son service résiduel, strict, qui généralise les travaux précédents, qui supposaient que DRR ne servait qu'à partager un débit constant.

Pour illustrer l'intérêt de cette généralisation, nous avons étudié la politique hiérarchique NP-SP/DRR. Dans cette politique, les flux sont tout d'abord servies par priorité : on peut imaginer trois niveaux de priorité, un premier niveau de très haute priorité pour des flux très

contraint, un second niveau pour les flux temps-réel à contrainte plus faible, et un dernier niveau pour le “best-effort”. On peut ensuite vouloir partager la bande passante entre les flux temps-réel avec une politique DRR. Le fait d’avoir calculé des services résiduels *stricts* permet de simplement composer les résultats obtenus sur NP-SP et DRR sans refaire une analyse spécifique pour NP-SP/DRR. Cet ordonnancement hiérarchique a été utilisé dans [35] pour évaluer la possibilité d’un partage de bande passante dans l’AFDX.

Conclusion générale et perspectives

5 - Résumé des travaux

5.1 Contexte

Les systèmes embarqués sont de plus en plus des systèmes communicants. Par exemple, dans un avion moderne, on trouve plusieurs applications qui s'échangent des informations à travers le réseau avionique. Bien entendu, un comportement correct de ces applications se doit d'être garanti. Du point de vue du réseau, cela signifie que lorsqu'une information y est transmise, il faudrait que celle-ci arrive à destination dans les délais. Dans un contexte de temps-réel embarqué, un résultat juste mais hors-délai est un résultat inutilisable. D'où la nécessité de se doter des moyens d'évaluation du délai pire cas d'un bout du réseau à l'autre bout. Plusieurs méthodes se sont penchées sur cette problématique : le model-checking, les méthodes à base de trajectoire, le modèle "Event stream" et le calcul réseau. Dans cette thèse, nous utilisons le calcul réseau pour évaluer le délai de traversée pire cas au sein d'un réseau embarqué temps réel.

5.2 Problématique

Le calcul réseau (network calculus) est une théorie basée sur l'algèbre min-plus. Il offre un cadre formel de modélisation des réseaux de communication. Il a été utilisé pour certifier le réseau AFDX embarqué dans l'A380 d'Airbus.

Dans un premier temps, nous nous sommes concentrés sur la politique de priorité statique non préemptive, présente dans la norme AFDX, avec l'objectif de réduire le plus possible les sur-approximations existantes dans sa modélisation.

Après la politique de priorité statique, qui est une politique de partage de bande passante qui suppose une hiérarchie stricte entre flux, nous nous sommes intéressé aux politiques de

partage de charge équilibrées, issues du modèle théorique GPS, qui est couramment implanté par la politique DRR.

5.3 Solution apportée

Le calcul réseau permet de garantir une certaine qualité de service à des flux du réseau à condition que ceux-ci respectent certaines contraintes. La contrainte sur un flux se modélise par une courbes d'arrivée, tandis-que la qualité de service se modélise au moyen d'une courbe du service. Si un flux, respectant une contrainte représentée par une courbe d'arrivée, traverse un serveur qui lui offre un service représenté par une courbe de service, alors ce flux subira un délai borné par la déviation horizontale qui existe entre la courbe d'arrivée et la courbe de service. En calcul réseau, l'expression des courbes d'arrivées des flux se fait plus ou moins bien. La difficulté réside dans l'expression de la courbe de service offert par un nœud. Cette difficulté est encore plus grande lorsque plusieurs flux traversent un ou plusieurs nœuds. Le service résiduel offert à chaque flux dépend alors de la politique d'ordonnancement adoptée.

Pour le cas des priorités statiques, nous somme partis de l'hypothèse selon laquelle lorsqu'un flux est servi, il devrait bénéficier de toute la puissance de calcul du serveur. Par conséquent, sa courbe de service devrait avoir une allure de courbe en escalier, contrairement à ce qui existe dans la littérature. Cette courbe en escalier alterne entre moment de service (pente de l'escalier) et les moments de non service (marche de l'escalier). La solution proposée vise à raffiner le service résiduel offert à un flux afin que la borne sur le délai subi par ce flux soit la plus proche possible du vrai délai. Des comparaisons, analytiques quand c'était possible ou expérimentales sinon, indiquent que nos résultats sont toujours aussi bons ou meilleurs que les travaux précédents, tout en étant plus généraux.

Nous avons aussi traité de la politique de service DRR. Nous proposons une expression du service résiduel pour un serveur offrant des services à plusieurs flux au moyen de cette politique. Nous avons enfin montré comment réaliser un partage hiérarchique de service avec la politique à priorité statique non préempive et la politique DRR.

6 - Perspectives de recherche

Dans l'état actuel des travaux, nous avons proposé une évaluation du service résiduel dans un contexte d'agrégation avec priorité statique non préemptive. Ce résultat permet de généraliser ceux existants, dans la limite de paquets de taille fixe. Il offre plus de précisions que ceux qui existent déjà dans la littérature du calcul réseau. Il est aussi précis que les résultats issus de

l'ordonnancement, qui sont réputés avoir le vrai pire délai avec cependant un champ d'application restreint (service à débit constant, ensemble de tâches périodiques). De plus, le fait pour notre approche d'être plus générale fait qu'on réussi à gérer plusieurs flux qui partagent des politiques de service différentes. L'inconvénient majeur de nos travaux réside dans l'hypothèse selon laquelle les flux considérés ont des paquets de même taille. Une généralisation de ces travaux aux flux à paquets de taille variable est nécessaire pour une exploitation industrielle.

Nous avons aussi étudié la politique de partage de charge DRR, avec là aussi des résultats généraux (tout type de courbe de service, tout type de flux d'entrée) avec là aussi des résultats aussi précis que l'existant. Des premiers travaux ont été réalisés pour évaluer l'applicabilité de DRR dans le contexte AFDX [35]. Ils concluent sur l'intérêt de la méthode, mais soulignent que DRR devra peut-être être adapté au contexte avionique, ce qui impliquera de modifier les analyses.

De manière plus générale, la gestion des paquets est à ce jour insatisfaisante en calcul réseau, comparée à la maturité d'autres résultats. Un cadre générique a été ébauché dans [24, 25] et mériterait plus d'attentions. Un premier travail pourrait être de reprendre nos travaux sur priorité statique par paquets et DRR dans ce cadre là pour voir son intérêt et ses limites.

Références bibliographiques

[1] ARINC 664. Aircraft data network, part 1 : Systems concepts and overview, 2002. 26

[2] ARINC 664. Aircraft data network, part 2 : Ethernet physical and data link layer specification, 2002. 26

[3] ARINC 664. Aircraft data network, part 7 : Deterministic networks, 2003. 26

[4] M. Adnan, J.-L. Scharbarg, J. Ermont, and C. Fraboul. Model for worst case delay analysis of an AFDX network using timed automata. In *Proc. of the 15th IEEE Conference on Emerging Technologies and Factory Automation (ETFA) – Real-time (and networked) embedded systems WiP track*, pages 1–4, sept. 2010. 36, 37, 38

[5] M. Adnan, J.-L. Scharbarg, J. Ermont, and C. Fraboul. An improved timed automata model for computing exact worst-case delays of afdx periodic flows. In *Proc. of the 15th IEEE Conference on Emerging Technologies and Factory Automation (ETFA) – Real-time (and networked) embedded systems WiP track*, pages 1 –4, sept. 2011. 36, 37, 38

[6] M. Adnan, J.-L. Scharbarg, and C. Fraboul. Minimizing the search space for computing exact worst-case delays of afdx periodic flows. In *Proc. of the 6th IEEE International Symposium on Industrial Embedded Systems*, pages 294–301, 2011. 36, 37

[7] AEEC. Arinc 825-2 : General standardization of can (controller area network) bus protocol for airborne use. Technical report, Airlines Electronic Engineering Committee, July 2011. 11, 70

[8] L. Almeida, P. Fonseca, and P. Fonseca. Flexible time-triggered communication on a controller area network. In *In Proceedings of the Work-In-Progress Session of the IEEE Real-Time Systems Symposium (RTSS'98*, pages 23–26. IEEE Computer Society, 1998. 22

[9] Luis Almeida, Eduardo Tovar, Francisco Vasques, Campo Universitario, Mecanica Gestao Industrial, Rua Dr, Roberto Frias, L. Almeida, E. Tovar, J. A. Fonseca, J. A. Fonseca, and F. Vasques. Schedulability analysis of real-time traffic in worldfip networks : an integrated approach. *IEEE Transactions on Industrial Electronics*, 3(49), 2002. 12

[10] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126 :183–235, 1994. 35

[11] Madhukar Anand, Steve Vestal, Samar Dajani-Brown, and Insup Lee. Formal modeling and analysis of the afdx frame management design. In *Proceedings of the Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, ISORC '06, pages 393–399, Washington, DC, USA, 2006. IEEE Computer Society. 36, 38

[12] A.B. Arjona. *Vérification et synthèse de systèmes temporisés par des méthodes d'obervation et d'analyse paramétrique*. PhD thesis, Ecole Nationale Supérieur de l'Aéronautique et de l'Espace, 1998. 38

[13] Henri Bauer. *Analyse pire cas de flux hétérogènes dans un réseau embarqué avion*. PhD thesis, Institut National Polytechnique de Toulouse, 2011. 2, 38, 39, 40, 41, 44, 45, 47, 48, 49

[14] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. Applying and optimizing trajectory approach for performance evaluation of afdx avionics network. In *Emerging Technologies and Factory Automation (ETFA), Palma de Mallorca, 22/09/2009-25/09/2009*, pages 1–8, http ://www.ieee.org/, septembre 2009. IEEE. 39, 48, 49

[15] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. Improving the worst-case delay analysis of an afdx network using an optimized trajectory approach. *IEEE Trans. Industrial Informatics*, 6(4) :521–533, 2010. 39, 48, 49

[16] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. Applying trajectory approach with static priority queuing for improving the use of available afdx resources). In *International Conference on Real-Time and Network Systems (RTNS), Toulouse, 04/11/2010-05/11/2010*, pages 69–78, http ://hal.inria.fr, 2011. HAL-INRIA. (distinction décernée : Best paper award). 39, 48, 49

[17] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. Worst-case end-to-end delay analysis of an avionics afdx network. In *Design, Automation and Test in Europe (DATE), Dresden, 08/03/2010-12/03/2010*, pages 1–6. EDAA, mars 2011. 39, 48, 49

[18] Klaus Bender, editor. *Profibus : the fieldbus for industrial automation*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. 23

[19] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen. *Systems and Software Verification : Model-Checking Techniques and Tools*. Springer Publishing Company, Incorporated, 1st edition, 2010. 38

[20] B. Bérard and L. Sierra. Comparing verification with hytech, kronos and uppaal on the railroad crossing example. Research Report LSV-00-2, CNRS-ENS Cachan, Jan. 2000. 38

[21] J. Berwanger, M. Peller, and R. Griessbach. Byteflight a new protocol for safety critical applications. In *Proc. of the FISITA World Automotive Congress*, Seoul, 2000. 22

[22] Benny Bing. Measured performance of the ieee 802.11 wireless lan. *Local Computer Networks, Annual IEEE Conference on*, 0 :34, 1999. 11

[23] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus*. Springer–Verlag, 2001. 54, 61, 63, 66, 70

[24] Anne Bouillard, Nadir Farhi, and Bruno Gaujal. Packetization and aggregate scheduling. Technical Report 7685, INRIA, 2011. 117

[25] Anne Bouillard, Nadir Farhi, and Bruno Gaujal. Packetization and packet curves in network calculus. In *Proc. of the 6th International Conference on Performance Evaluation Methodologies and Tools (ValueTools 2012)*, Cargese, France, October, 9–12 2012. Invited Paper. 117

[26] Anne Bouillard, Laurent Jouhet, and Eric Thierry. Service curves in network calculus : dos and don'ts. Rapport de recherche INRIA 7094, INRIA, November 2009. 54, 64, 65, 66, 67, 68, 69, 93

[27] Anne Bouillard, Laurent Jouhet, and Eric Thierry. Tight performance bounds in the worst-case analysis of feed-forward networks. Technical Report 7012, INRIA, 2009. 70

[28] Anne Bouillard, Laurent Jouhet, and Eric Thierry. Comparison of different classes of service curves in network calculus. In *Proc. of the 10th International Workshop on Discrete Event Systems (WODES 2010)*, Technische Universität Berlin, August 30 - September 1 2010. 70, 98

[29] Anne Bouillard, Laurent Jouhet, and Eric Thierry. 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, march 2010. 70

[30] Anne Bouillard and Giovanni Stea. Exact worst-case delay for fifo-multiplexing tandems. In *Proc. of the 6th International Conference on Performance Evaluation Methodologies and Tools (ValueTools 2012)*, Cargese, France, October, 9–12 2012. 70, 71

[31] Anne Bouillard and Éric Thierry. An algorithmic toolbox for network calculus. *Discrete Event Dynamic Systems*, 18(1) :3–49, March 2008. 70

[32] Marc Boyer, Jörn Migge, and Marc Fumey. PEGASE, a robust and efficient tool for worst case network traversal time. In *Proc. of the SAE 2011 AeroTech Congress & Exhibition*, Toulouse, France, 2011. SAE International. 48, 70, 92

[33] Marc Boyer, Jörn Migge, and Nicolas Navet. An efficient and simple class of functions to model arrival curve of packetised flows. In *Proceedings of the 1st International Workshop on Worst-Case Traversal Time (WCTT'2011)*, pages 43–50, New York, NY, USA, novembre 2011. ACM. 70

[34] Marc Boyer, Nicolas Navet, and Marc Fumey. Experimental assessment of timing verification techniques for afdx. In *Proc. of the 6th Int. Congress on Embedded Real Time Software and Systems*, Toulouse, France, February 2012. 70

[35] Marc Boyer, Nicolas Navet, Marc Fumey, Jörn Migge, and Lionel Havet. Combining static priority and weighted round-robin like packet scheduling in afdx for incremental certification and mixed-criticality support. In *Proc. of the 5th European Conference for Aeronautics and Space Sciences – Real Time Avionics and Networks Session (EUCASS 2013)*, Munich, Germany, July 1st–5th 2013. 4, 113, 117

[36] Marc Boyer, Giovanni Stea, and William Mangoua Sofack. Deficit round robin with network calculus. In *Proc. of the 6th International Conference on Performance Evaluation Methodologies and Tools (ValueTools 2012)*, Cargese, France, October, 9–12 2012. 3

[37] Giorgio C. Buttazzo. *Hard Real-time Computing Systems : Predictable Scheduling Algorithms And Applications*. Springer Publishing Company, Incorporated, 2nd edition, 2005. 5

[38] Tom S. Chan. *Time-Division Multiple Access*, pages 769–778. John Wiley & Sons, Inc., 2007. 13

[39] Hussein Charara. *Évaluation des performances temps reel de reseaux embarques avioniques*. PhD thesis, Institut National Polytechnique de Toulouse, 2004. 37

[40] Hussein Charara, Jean-Luc Scharbarg, Jerome Ermont, and Christian Fraboul. Methods for bounding end-to-end delays on an afdx network. In *Euromicro Conference on Real-Time Systems (ECRTS) Dresden (Germany) 05/07/2006-07/07/2006*, Los Alamitos, CA, USA, 2006. IEEE Computer Society. 36, 37, 38

[41] Chang Cheng-Shang. *Performance Guarantees in Communication Networks*. Springer-Verlag, 2000. 54, 70

[42] Devesh B. Chokshi and Purandar Bhaduri. Modeling fixed priority non-preemptive scheduling with real-time calculus. In *RTCSA '08 : Proc. of the 2008 14th IEEE int. conf. on Embedded and Real-Time Computing Systems and Applications*, Washington, DC, USA, 2008. IEEE Computer Society. 68, 69, 93

[43] Devesh B. Chokshi and Purandar Bhaduri. Performance analysis of flexray-based systems using real-time calculus, revisited. In *Proc. of the 2010 ACM Symposium on Applied Computing (SAC'10)*, pages 351–356, New York, NY, USA, 2010. ACM. 70

[44] CIAME. *Réseaux de terrain : Description et critères de choix*. IC2 : Série Systèmes automatisés. Hermès, 1999. 22

[45] COINC home page. http ://www.istia.univ-angers.fr/˜lagrange/software.php. 70

[46] Alan Colvin. Csma with collision avoidance. *Computer Communications*, 6(5) :227 – 235, 1983. 11

[47] Francis Cottet, Joëlle Delacroix, Kaiser Claude, and Zoubir Mammeri. *Ordonnancement temps réel - Cours et exercices corrigés* . Hermès, janvier 2000. 14

[48] Francis Cottet, Joëlle Delacroix, Claude Kaiser, and Zoubir Mammeri. Ordonnancement temps réel : Ordonnancement réparti . In J. Blouet, G. Bonnier, F. Louage, and J.C. Pascal, editors, *Techniques de l'ingénieur, traité Informatique industrielle* . Techniques de l'ingénieur, 249, rue de Crimée 75925 Paris, mars 2000. Pages de la publication : 26pages. 15, 16

[49] Rene L. Cruz. A calculus for network delay, part I : Network elements in isolation. *IEEE Transactions on information theory*, 37(1) :114–131, January 1991. 2, 54

[50] Rene L. Cruz. A calculus for network delay, part II : Network analysis. *IEEE Transactions on information theory*, 37(1) :132–141, January 1991. 2, 54, 70

[51] Robert I. Davis, Alan Burns, Reinder J. Bril, and Johan J. Lukkien. Controller area network (CAN) schedulability analysis : Refuted, revisited and revised, 2007. 29, 33, 34, 90, 92, 93

[52] B. Drury and Institution of Electrical Engineers. *The Control Techniques Drives and Controls Handbook*. Iet Power and Energy Series. Institution of Engineering and Technology, 2009. 23

[53] BAJIC Eddy and BOUARD Bruno. Réseau Profibus. *Techniques de l'ingénieur. Informatique industrielle ISSN 1632-3831*, S3, noS8160 :S8160.1–S8160.8, 2000. [Note(s) : S8160.1 [19 p]. 23

[54] Wilfried Elmenreich and Richard Ipp. Introduction to ttp/c and ttp/a, 2003. 22

[55] Hartwich F., Müller B., Führer T., Hugel R., and Robert Bosch GmbH. Timing in the ttcan network. In *Proceedings of the 8th International CAN Conference*, Las Vegas, 2002. 22

[56] P. Ferrari, A. Flammini, and S. Vitturi. Performance analysis of profinet networks. *Comput. Stand. Interfaces*, 28(4) :369–385, April 2006. 25

[57] O. G. Gabbard and P. Kaul. Time-division multiple access. In *EASCON '74 ; Electronics and Aerospace Systems Convention*, pages 179–184, 1974. 13

[58] Robert Bosch GmbH. Can specification version 2.0. Postfach 30 02 40, 1991. D-70442 Stuttgart. 27

[59] Jérôme Grieu. *Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques*. Thèse de Doctorat, Institut National Polytechnique Toulouse, Toulouse, 2004. 2, 25, 30, 48, 53

[60] Nan Guan, Zonghua Gu, Wang Yi, and Ge Yu. Improving scalability of model-checking for minimizing buffer requirements of synchronous dataflow graphs. In *Proceedings of the 2009 Asia and South Pacific Design Automation Conference*, ASP-DAC '09, pages 715–720, Piscataway, NJ, USA, 2009. IEEE Press. 36

[61] W. Haid and L. Thiele. Complex task activation schemes in system level performance analysis. In *ESWeek'07 : Proc. of the 5th IEEE/ACM int. conf. on Hardware/Software Codesign and System Synthesis (Salzburg, Austria, September 30 - October 03, 2007)*, pages 173–178, New York, NY, USA, 2007. ACM. 65, 66, 67, 68, 69, 93

[62] Rafik Henia, Arne Hamann, Marek Jersak, Razvan Racu, Kai Richter, and Rolf Ernst. System level performance analysis - the symta/s approach. In *IEE Proceedings Computers and Digital Techniques*, 2005. 49, 50, 53

[63] Pierre-Emmanuel Hladik and Anne-Marie Déplanche. Extension au réseau can des problèmes de placement. Research Report RI 2005_4, Institut de Recherche en Communication et Cybernétique de Nantes, Jun. 2005. 27

[64] Condor Engineering Incorporated. Mil-std-1553 designer guide. ILC data service corporation, 1982. 26

[65] ISO 11519-2 International Standard Organization. Road vehicles - low speed serial data communication - part 2 : Low speed controller area network. ISO, 1994. 24

[66] ISO 11898 International Standard Organization. Road vehicles - interchange of digital information - controller area network for high-speed communication. ISO, 1994. DO-178B, Washington DC. 24

[67] Jürgen Jasperneite and J. Feld. Profinet : an integration platform for heterogeneous industrial communication systems. In *Proceedings of 10th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2005, September 19-22, 2006, Catania, Italy*. IEEE, 2005. 25

[68] FELD Joachim. PROFINET - scalable factory communication for all applications. In *Proceedings of the 2004 IEEE International Workshop on Factory Communication Systems*, WFCS 2004, pages 33 – 38, Piscataway NJ, ETATS-UNIS, 2004. IEEE Computer Society. 25

[69] Patrice KADIONIK. Le bus can. école nationale supérieure électronique , informatique & radiocommunication Bordeaux, 2001. 28

[70] Richter Kai. *Compositional Performance Analysis*. PhD thesis, Technical University of Braunschweig, 2004. 52

[71] S.S. Kanhere and H. Sethu. On the latency bound of deficit round robin. In *Proc. of 11th Int. Conf. on Computer Communications and Networks (ICCCN 2002)*, pages 548 – 553, oct. 2002. 99, 105

[72] Georges Kemayo, Frédéric Ridouard, Henri Bauer, and Pascal Richard. Optimism due to serialization in the trajectory approach for switched ethernet networks. In *Proc. of the 7th Junior Researcher Workshop on Real-Time Computing (JRWRTC 2013)*, Sophia Antipolis, France, October 16-18 2013. 39, 48

[73] Georges Kemayo, Frédéric Ridouard, Henri Bauer, and Pascal Richard. Optimistic problems in the trajectory approach in fifo context. In *Proc. of the 18th IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA 2013)*, 2013. 39, 48

[74] Omar KERMIA. *Ordonnancement temps réel multiprocesseur de tâches non-préemptives avec contraintes de précédence, de périodicité stricte et de latence*. PhD thesis, Université Paris XI, 2009. 32

[75] Hermann Kopetz, Astrit Ademaj, Petr Grillinger, and Klaus Steinhammer. The time-triggered ethernet (tte) design. In *Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, ISORC '05, pages 22–33, Washington, DC, USA, 2005. IEEE Computer Society. 26

[76] Hermann Kopetz and Günther Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1) :112–126, 2003. 24

[77] Hermann Kopetz and Günter Grünsteidl. TTP-A protocol for fault-tolerant real-time systems. *Computer*, 27(1) :14–23, January 1994. 24

[78] Hermann Kopetz, Michael Holzmann, and Wilfried Elmenreich. A universal smart transducer interface : TTP/A. *Comput. Syst. Sci. Eng.*, 16(2) :71–77, 2001. 22

[79] Anis Koubaa and Ye-Qiong Song. Evaluation de performances d'Ethernet commuté pour des applications temps réel. In *10th International Conference on Real Time and Embedded Systems - RTS'2002*, Paris/France, 2002. Colloque avec actes et comité de lecture. internationale. A02-R-344 || koubaa02g A02-R-344 || koubaa02g. 25

[80] J.F. Kurose, K.W. Ross, and S. Pauquet. *Analyse structurée des réseaux : Des applications de l'internet aux infrastructures de télécommunication*. Pearson Education, 2003. 10, 11, 12

[81] Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2) :134–152, October 1997. 37

[82] Michaë Lauer, Jérôme Ermont, Claire Pagetti, and Frédéric Boniol. Analyzing end-to-end functional delays on an ima platform. In *Proceedings of the 4th international conference on Leveraging applications of formal methods, verification, and validation - Volume Part I*, ISoLA'10, pages 243–257, Berlin, Heidelberg, 2010. Springer-Verlag. 38

[83] Gerard Le Lann. Critical issues for the development of distributed real-time computing systems. Technical Report RR-1274, INRIA, August 1990. Projet REFLECS. 5

[84] Guy Leduc, Olivier Bonaventure, Eckhart Koerner, Luc Léonard, Charles Pecheur, and David Zanetti. Specification and verification of a ttp protocol for the conditional access to services, 1996. 22

[85] Luciano Lenzini, Linda Martorini, Enzo Mingozzi, and Giovanni Stea. Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree network. *Performance Evaluations*, 63 :956–987, 2005. 71

[86] Luciano Lenzini, Eenzo Mingozzi, and Giovanni Stea. Aliquem : a novel DRR implementation to achieve better latency and fairness at O(1) complexity. In *Proc. of 10th IEEE Int. Workshop on Quality of Service (IWQoS 2002)*, pages 77 – 86, 2002. 94, 105

[87] Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. Full exploitation of the deficit round robin capabilities by efficient implementation and parameter tuning. Technical report, Dipartimento di Ingegneria della Informazione, University of Pisa, October 2003. 105

[88] Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. Delay bounds for FIFO aggegates : a case study. *Computer Communications*, 28 :287–299, 2004. 65

[89] Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. Tradeoffs between low complexity, low latency and fairness with deficit round robin schedulers. *IEEE/ACM Transactions on Networking*, 12(4) :681–693, August 2004. 105

[90] Joseph Y.-T. Leung and M. L. Merrill. A note on preemptive scheduling of periodic, real-time tasks. *Inf. Process. Lett.*, pages 115–118, 1980. 19

[91] Xiaoting Li. *Worst case delay analysis of real-time switched Ethernet networks with flow local synchronisation*. PhD thesis, Univ. of Toulouse – INP Toulouse, September 2013. 48

[92] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1) :46–61, January 1973. 19

[93] Zhao Luxi, Xiong Huagang, and Li Qiao. Using memo recursive computation in the trajectory approach for the worst-case delay analysis of afdx networks. In *Proceedings of the 2nd International Conference on Electrical and Control Engineering (ICECE2011)*, pages 5633 – 5638. IEEE, 2011. 48

[94] Zoubir Mammeri. Méthodes d'accès au support de transmission dans les réseaux filaires et sans fils (MAC : Medium Access Control). M1 Info - Cours de Réseaux - Université Paul Sabatier, 2011. 9, 10, 12, 13

[95] William . Mangoua Sofack and Marc Boyer. Non preemptive static priority with network calculus. In *Proc. of the 16th IEEE Conference on Emerging Technologies Factory Automation (ETFA'2011)*, pages 1–8. IEEE, sept. 2011. 3

[96] William . Mangoua Sofack and Marc Boyer. Non preemptive static priority with network calculus : Enhancement. In *Proc. of the Workshop on Network Calculus (WoNeCa 2012)*, Kaiserslautern, Germany, March 2012. 3

[97] William Mangoua Sofack and Marc Boyer. Generalisation of GPS and P-GPS in network calculus. In *Proc. of the Work in Progress session of 9th IEEE Workshop on Factory Communication Systems (WFCS'2012)*, Lemgo/Detmold, Germany, May 21–24 2012. 3

[98] Marco Ajmone Marsan, Giovanni Chiola, and Andrea Fumagalli. An accurate performance model of CSMA/CD bus LAN. In *Advances in Petri Nets 1987, covers the 7th European Workshop on Applications and Theory of Petri Nets*, pages 146–161, London, UK, UK, 1987. Springer-Verlag. 10

[99] Steven MARTIN. *Maîtrise de la dimension temporelle de la qualité de service dans les réseaux*. PhD thesis, Université Paris XII Val de Marne, July 2004. 19, 39, 40, 41, 42, 43, 44, 47, 48

[100] Steven Martin and Pascale Minet. Schedulability analysis of flows scheduled with fifo : application to the expedited forwarding class. In *Proceedings of the 20th international conference on Parallel and distributed processing*, IPDPS'06, pages 180–180, Washington, DC, USA, 2006. IEEE Computer Society. 39, 48

[101] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993. 37

[102] Ahlem Mifdaoui. *Spécification et validation d'un réseau de communication de type Ethernet Commuté pour systèmes avioniques militaires de nouvelles générations*. PhD thesis, Institut National Polytechnique de Toulouse, December 2007. 26, 27

[103] Amir Muhammad and Michael J. Pont. A time-triggered communication protocol for can-based networks with a fault-tolerant star topology. In *Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology*, CIT '10, pages 2347–2354, Washington, DC, USA, 2010. IEEE Computer Society. 22

[104] Nicolas Navet, YeQiong Song, Françoise Simnot-Lion, and Cédric Wilwert. Trends in automotive communcation systems (invited paper). *Proceedings of the IEEE, special issue on Industrial Communications Systems*, 96(6) :1204–1223, June 2005. 24

[105] Hu Ning, Lv Tangqi, and Huang Ning. Applying trajectory approach for computing worst-case end-to-end delays on an afdx network. *Procedia Engineering*, 15 :2555–2560, December 2011. 49

[106] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks : the single-node case. *IEEE/ACM Transactions on Networking (TON)*, 1, 1993. 20, 21, 65, 94, 95

[107] D. Paret. *Le Bus CAN, applications : CAL, CANopen, DeviceNet, OSEK, SDS*. Electronique. Série électronique appliquée. Dunod, 1999. 28

[108] PI. Profinet théorie et pratique – system description. Technical report, PROFIBUS and PROFINET International (PI), 2003. 25, 26

[109] Victor Pollex, Henrik Lipskoch, Frank Slomka, and Steffen Kollmann. Runtime improved computation of path latencies with the real-time calculus. In *Proc. of the 1st International Workshop on Worst-Case Traversal Time*, WCTT '11, pages 58–65. ACM, 2011. 56

[110] Traian Pop, Paul Pop, Petru Eles, Zebo Peng, and Alexandru Andrei. Timing analysis of the flexray communication protocol. *Real-Time Syst.*, 39(1-3) :205–235, August 2008. 22

[111] PROFIBUS. PROFInet Théorie et pratique. PROFIBUS International : Centre technique, 2003. D-70442 Stuttgart. 23

[112] Kai Richter, Marek Jersak, and Rolf Ernst. A formal approach to mpsoc performance verification. *Computer*, 36(4) :60–67, April 2003. 51

[113] Kai Richter, Razvan Racu, and Rolf Ernst. Scheduling analysis integration for heterogeneous multiprocessor soc. In *In Proceedings 24th International Real-Time Systems Symposium (RTSS'03), Cancun*, 2003. 51

[114] J. Rox and R. Ernst. Exploiting inter-event stream correlations between output event streams of non-preemptively scheduled tasks. In *Proc. of Int. Conf. on the Design, Automation and Test in Europe (DATE 2010)*, pages 226 –231, march 2010. 53

[115] Ece Guran Schmidt and Klaus Schmidt. Message scheduling for the FlexRay protocol : the dynamic segment. *IEEE Transactions on vehicular technology*, 58(5) :2160–2169, 2009. 22

[116] J.B. Schmitt, F.A. Zdarsky, and M. Fidler. Delay bounds under arbitrary multiplexing : When network calculus leaves you in the lurch... In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1669 –1677, april 2008. 71, 92

[117] Philippe Schnoebelen, Béatrice Bérard, Michel Bidoit, François Laroussinie, and Antoine Petit. *Vérification de logiciels : techniques et outils du model-checking*. Vuibert, April 1999. 35

[118] P. Sénac. *Contribution à la modélisation des systèmes multimédias et hypermédias*. PhD thesis, Université Paul Sabatier, 1996. 5

[119] C. Servin. *Réseaux & télécoms : Cours avec 129 exercices corrigés*. Sciences sup. Dunod, 2006. 10, 11, 12

[120] Michael Short and Michael J. Pont. Fault-tolerant time-triggered communication using can. *IEEE Trans. Industrial Informatics*, 3(2) :131–142, 2007. 22

[121] M. Shreedhar and George Varghese. Efficient fair queueing using deficit round robin. *SIGCOMM Computer Communication*, 25 :231–242, October 1995. 96, 99, 105

[122] Klaus Steinhammer, Petr Grillinger, Astrit Ademaj, and Hermann Kopetz. A time-triggered ethernet (tte) switch. In *Proceedings of the conference on Design, automation and test in Europe : Proceedings*, DATE '06, pages 794–799, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association. 26

[123] Andy Swales. Open modbus/tcp specification. Research Report Release 1.0, Schneider Electric, March 1999. 12, 23

[124] Jean-Pierre Thomesse. Les réseaux de terrain. In *Institut National Polytechnique de Lorraine*, page 41 p. TRIO - INRIA Lorraine - LORIA, 2001. thomesse01b A01-R-321 || thomesse01b. 22

[125] Jean-Pierre Thomesse. The WorldFIP Fieldbus. In Richard Zurawski, editor, *Industrial Information Technology Handbook*, Industrial Electronics Series, page 26 p. CRC Press, 2004. Contribution à un ouvrage. A03-R-372 || thomesse03a A03-R-372 || thomesse03a. 12

[126] K. Tindell, A. Burns, and A. Wellings. Calculating controller area network (can) message response times. *Control Engineering Practice*, 3 :1163–1169, 1995. 28, 33

[127] K. W. Tindell. An extendible approach for analysing fixed priority hard real-time tasks. *JOURNAL OF REAL-TIME SYSTEMS*, 6, 1994. 33

[128] Ken Tindell and Alan Burns. Guaranteed message latencies for distributed safety-critical hard real-time control networks. In *Proceedings of 1st International CAN Conference*, pages 1 – 11, 1994. 33

[129] Ken Tindell, H. Hanssmon, and Andy J. Wellings. Analysing real-time communications : Controller area network (can). In *IEEE Real-Time Systems Symposium'94*, pages 259–263, 1994. 33

[130] Tao Wan and Asrar U. Sheikh. Performance analysis of buffered csma/cd systems. *Wirel. Pers. Commun.*, 18(1) :45–65, July 2001. 10

[131] Ernesto Wandeler. *Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems*. PhD thesis, ETH Zurich, September 2006. 54

[132] Ernesto Wandeler, Lothar Thiele, Marcel Verhoef, and Paul Lieverse. System architecture evaluation using modular performance analysis : a case study. *Int. J. Softw. Tools Technol. Transf.*, 8(6) :649–667, October 2006. 54

[133] Pengcheng Zhang, Henry Muccini, and Bixin Li. A classification and comparison of model checking software architecture techniques. *Journal of Systems and Software*, 83(5) :723–744, 2010. 37

Le calcul réseau (network calculus) est une théorie basée sur l'algèbre min-plus. Il offre un cadre formel de modélisation des réseaux de communication. Il a été utilisé pour certifier le réseau AFDX embarqué dans l'A380 de Airbus. Seulement, les bornes sur le délai annoncés par ces travaux de certification souffrent d'une sur-approximation dans le cas précis de l'agrégation dans un contexte de priorité statique non préemptive. L'objectif de nos travaux est de réduire cette sur-approximation. Dans cette thèse, nous proposons un service résiduel permettant d'obtenir de meilleurs bornes sur le délai dans le cas de la politique à priorité statique non préemptive et de la politique DRR. Nous montrons aussi comment ces deux politiques peuvent être combinées dans une politique hiérarchique à deux niveaux.

Mots clés : Calcul réseau, QoS, réseaux temps réel embarqués, analyse de performances

The thesis addresses performance analysis of embedded real time network using network calculus. Network calculus is a theory based on min-plus algebra. We use network calculus to assess the quality of service of a residual flow in two context : aggregation with non-preemptive priority policy and DRR policy. The main contribution concerns the evaluation of residual service, given to each flow. We also present how to handle DRR and non-preemptive priority policy hierrachically.

Keywords : Network calculus, QoS, embedded real time networks, performance analysis