Algorithmique du Network Calculus

Laurent Jouhet

2012ENSL0760

THÈSE

en vue d'obtenir le grade de

Docteur de l'École Normale Supérieure de Lyon - Université de Lyon

Discipline : Informatique

Laboratoire de l'Informatique du Parallélisme

École doctorale : Mathématiques et Informatique fondamentale

présentée et soutenue publiquement le 7 novembre 2012

par Monsieur Laurent JOUHET

Algorithmique du Network Calculus

Directeur de thèse : Monsieur Éric THIERRY

Après avis de : Monsieur Christian FRABOUL
Monsieur Laurent HARDOUIN
Monsieur Giovanni STEA

Devant la Commission d'examen formée de :

Madame Anne BOUILLARD, ENS	Membre
Monsieur Éric FLEURY, ENS de Lyon	Membre
Monsieur Christian FRABOUL, ENSEEIHT	Rapporteur
Monsieur Bruno GAUJAL, INRIA Grenoble	Membre
Monsieur Laurent HARDOUIN, ISTIA	Rapporteur
Monsieur Giovanni STEA, Université de Pise	Rapporteur
Monsieur Éric THIERRY, ENS de Lyon	Directeur

MERCI

Le travail présenté dans ce mémoire a été effectué au sein de l'équipe *Modèles de Calcul et Complexité* (MC2) du *Laboratoire de l'Informatique du Parallélisme*, à l'*École Normale Supérieure de Lyon*.

Je tiens à remercier tout d'abord, pour la réalisation de cette thèse :

- Éric Thierry et Anne Bouillard, ainsi que Marc Boyer, pour tous leurs conseils, leurs encouragements, et leur encadrement sans faille ;
- Christian Fraboul, Laurent Hardouin et Giovanni Stea, pour avoir accepté d'être rapporteurs de cette thèse. Ainsi qu'Éric Fleury et Bruno Gaujal pour avoir accepté de faire partie du jury ;
- Anne Reverdy et Monique Rouillon, sans qui je n'aurais pas réussi à écrire ces lignes ; et Catherine Reverdy pour sa relecture rapide et efficace !

Merci aussi à l'administration ainsi qu'aux enseignants-chercheurs de l'ENS de Lyon , en particulier Damien Séon et Jean-Michel Müller.

Pendant ces années, j'ai eu plaisir à apprendre aux côtés de nombreuses personnes :

- à MC2 : Pascal Koiran, Natacha Portier, Nicolas Schabanel, Pablo Arrighi, Jonathan Grattage ;
- dans mon bureau : Florent Becker, Irénée Briquel, Julien Robert, Mathilde Noual, Sylvain Sené, Damien Régnault, Bruno Grenet ;
- aux réunions WEED : en particulier Laurent Hardouin, Bertrand Cottenceau, Euriell Le Corronc, Sébastien Lagrange ;
- et à Chevaleret : Jean Mairesse ; ou à Plume : Daniel Hirschkoff, Alexandre Miquel.

Je tiens évidemment à remercier ma famille et mes amis pour leur soutien :

- Bulletin : Anne et Paul et Margaux, Aurel et Élise, Camilla et Damien et Lucas, Edern et Booli, Euzeb et Élise, Gabi et Gaëlle et Maxime, Glau et Fanny, Guizzmo, Hélène et Bouli, Joan et Olivia, Joris et Hélène, Juju, Kazoo et Claire, Laurin, Marie-Marine, Marjo et Framboise, Mathilde, Pad et Agathe, Phulbert et Nolwenn et Luna, Pierre et Marianne, Tonfa ;
- ainsi que : Marc, Pierre(s), Barbara, Carine, « Joséphine » ;
- et les collocs de Charpennes : Martina, Patricia, Fanny, Gilles ;
- ma famille : Jean-Pierre, Marie, Nathalie, Cyrille, Jules, Jeanne, Claire, Jean-Philippe, Daniel, Florence, Laura, Simon ; ainsi que toute la famille Perrot ;
- les Rouillon : Monique, Philippe, Xavier, Marie, Loreleï, Thaïs, Arnaud, Céline ;
- les Rever-Maistre : Serge et Monique, Mathieu, Nicolas et Catherine et Alice et Thomas, Geneviève et Pierre, Claire-Line et Régis et les enfants, Aude-Marie et Thierry et Siméon, La Zab, Michèle et Robert, Pierre-Marie et Johanna et Emma ;
- mes nouveaux collègues, en particulier Amélie, Aurélie, Alexandra, Isabelle(s), Cécile ; mais aussi Patricia, Dominique, Marc, Stéphanie(s), Alexandre, Colette, Lydie, Clément, Sylvie, Geneviève, Nathalie, Ludivine, Jean-Luc, Jean-Michel, Jean ;
- ainsi que : Fanny (!), Daniel, Alex, Mélina, Vincent ;
- mais aussi, bien sûr : Raphaëlle, Kara et Claude, Amandine et Nico, Lucie, Claire, Isa, Magali, Maud et Romain, Guillaume.

Une pensée aussi pour mes élèves, que j'aurai plaisir à retrouver à la rentrée.

Je tiens enfin à remercier tout particulièrement Anne, qui a su à la fois me pousser dans la bonne direction, et être à mes côtés à chaque instant. Merci.

LE NETWORK CALCULUS EXPLIQUÉ À MA MAMAN

Imaginons une jolie petite échoppe. Imaginons aussi la ribambelle de clients qui va entrer dans la sus-dite échoppe pour faire la queue à un guichet afin d'obtenir un service (acheter un objet, obtenir un renseignement, se plaindre...).

On dispose d'un certain nombre d'informations sur l'arrivée des clients et sur la rapidité des employés au guichet. Par exemple :

- Avant l'ouverture à 9h00, il y a rarement des clients déjà présents, mais en tout cas jamais plus de 20. Jamais.
- Il arrive environ 2 personnes par heure tout au long de la journée, mais jamais plus de 10 par heure. Jamais.
- Le guichetier sait traiter en moyenne 3 cas par heure. Mais même dans le pire des cas, il traite toujours au moins 1 cas par heure. Toujours.

On se propose de répondre à un certain nombre de questions, comme par exemple :

- Est-on sûr que tous les clients seront servis dans la journée ?
- Combien de temps un client risque-t-il d'attendre dans le pire des cas ?
- À un instant donné, dans le pire des cas possibles, quelle va être la longueur de la file d'attente au guichet ?

Les réponses à ces questions peuvent sembler triviales...

On remarquera que les informations sur ce qui se passe habituellement (il y a rarement des clients présents avant l'ouverture ; il arrive environ 2 clients par heure ; un guichetier traite en moyenne 3 cas par heure) ne nous seront pas d'une grande utilité pour répondre à ces questions.

Ce qui nous intéresse, c'est ce qui se passe dans le pire des cas pour chacune des questions auxquelles on veut répondre. Le jour où un client attendra le plus longtemps ne sera pas forcément le jour où la file d'attente sera la plus longue.

Répondre à ces questions n'est pas toujours facile

- Que se passe-t-il s'il y a plusieurs guichets ?
- Que se passe-t-il si les clients peuvent passer les uns devant les autres dans la file d'attente ?
- Ou que les clients avec un dossier incomplet sont renvoyés au début de la file ?
- ...

TABLE DES MATIÈRES

NOTATIONS UTILISÉES

Abréviations et acronymes

AFDX	Avionics Full DupleX switched ethernet	(p. 40)
ATM	Asynchronous Transfert Mode	(p. 15)
DiffServ	Differentiated Services	(p. 40)
IETF	Internet Engineering Task Force	(p. 39)
IntServ	Integrated Services	(p. 40)
IP	Internet Protocol	(p. 15)
LP	Problème d'optimisation linéaire (instance LP)	(p. 127)
NC	Network Calculus	(p. 15)
PBOO	Pay Bursts Only Once	(p. 37)
PMOO	Pay Multiplexing Only Once	(p. 37)
QoS	Qualité de Service	(p. 39)
RTC	Real-Time Calculus	(p. 79)
SED	Systèmes à événements discrets	(p. 41)
SFA	Separate flow analysis	(p. 37)
TFA	Total flow analysis	(p. 37)
ULP	Méthode utilisant une *unique* instance LP	(p. 144)
VCN	Variable Capacity Node	(p. 29)

Notations

$\mathbb{1}_S$	Fonction indicatrice de S	(p. 54)
α	Courbe d'arrivée	(p. 27)
(A, B)	Trajectoire entrée/sortie d'un système ; cf. (F^{in}, F^{out})	(p. 32)
β	Courbe de service	(p. 28)
$\beta_{\lambda,d}$	Segment élémentaire	(p. 100)

B_{max}	Charge pire-cas	(p. 32)
$\mathcal{BP}$	Période chargée (*backlogged period*)	(p. 29)
$\beta_{R,T}$	Courbe de service : latence-débit	(p. 31)
$\mathcal{D}$	$\left\{f \ : \mathbb{N} \to \overline{\mathbb{R}}\right\}$	(p. 21)
$dist(t, S)$	Distance à S	(p. 54)
D_{max}	Délai pire-cas	(p. 32)
δ_T	Courbe de service : délai pur	(p. 31)
$\widehat{f} \in \mathcal{F}^+$	Transformée de Legendre-Fenchel de $f \in \mathcal{F}^+$	(p. 153)
f^*	Clôture sous-additive de f	(p. 24)
$f \oslash g$	Déconvolution de f et g	(p. 24)
$f * g$	Convolution de f et g	(p. 24)
$\mathcal{F}$	$\left\{f \ : \mathbb{R}_+ \to \overline{\mathbb{R}}\right\}$	(p. 21)
$\mathcal{F}_{\nearrow}$	Fonctions de $\mathbb{R}+$ dans $\mathbb{R}+$, croissantes, continues à gauche, et qui vérifient $f(0) = 0$	(p. 21)
(F^{in}, F^{out})	Trajectoire entrée/sortie d'un système ; cf. (A, B)	(p. 28)
$\mathcal{F}[X, Y]$	Ensemble des fonctions affines par morceaux de $\mathcal{F}$ « de X dans Y »	(p. 52)
$h(\alpha, \beta)$	Distance horizontale entre α et β	(p. 32)
λ_r	Courbe de service : débit constant	(p. 31)
min, max	Minimum, maximum	(p. 23)
$\mathcal{N}$	Réseau : serveurs et flux	(p. 26)
$\oplus, \otimes$	Lois d'un dioıde	(p. 22)
ψ	Convolution multidimensionnelle	(p. 117)
$\hat{R}[s, t)$	Conversion NC vers RTC	(p. 80)
$\check{R}(t)$	Conversion RTC vers NC	(p. 80)
r_i, τ_i	Pente et distance horizontale d'un « morceau » dans une fonction affine par morceaux	(p. 153)
$\overline{\mathbb{R}}$	$\mathbb{R} \cup \{-\infty; +\infty\}$	(p. 21)
$\mathcal{S}_{\mathcal{T}}(\beta)$	Ensemble de toutes les trajectoires qui admettent β comme courbe de service de type $\mathcal{T}$, avec $\mathcal{T} \in \{simple, strict, wstrict, vcn, asc, s3c\}$	(p. 29)
$start(t)$	Début de la période chargée de t	(p. 29)
$\mathcal{T}$	Type de courbe de service, parmi *vcn*, *strict*, *wstrict*, *simple*	(p. 91)
$v(\alpha, \beta)$	Distance verticale entre α et β	(p. 32)
X	Représente des ensembles de nombres, par exemple N ou $\mathbb{R}_+$	(p. 51)

Notations : optimisation linéaire

$\mathbb{F}$	Ensemble des flux	(p. 125)
$F_i^{(0)}$	Fonction cumulée à l'entrée du réseau	(p. 125)
$F_i^{(j)}$	Fonction cumulée à la sortie du serveur j	(p. 125)
i	Flux i	(p. 125)
j	Serveur j	(p. 125)
$j \in i$	Le serveur j appartient à μ_i, la suite des serveurs qui sont traversés par le flux i	(p. 125)
$j\pi$	Chemin commençant par le serveur j suivi du chemin π	(p. 125)
μ_i	La suite des serveurs traversés par le flux i	(p. 125)
$\mathcal{N}$	Réseau	(p. 125)
opt_λ	Valeur optimale d'un ensemble de solution λ	(p. 132)
(P1), (P2)	Prédicats que doivent vérifier les t_π	(p. 128)
π	Chemin dans le réseau	(p. 125)
Π_j, Π_i, Π	Ensemble des chemins qui terminent au serveur j, ou au noeud $last(i)$, ou en $last(i)$ avec i le flux observé	(p. 127)
$\text{prec}_i(j)$	Le prédécesseur de j dans la suite μ_i	(p. 125)
$\sigma_{i,\ell} + \rho_{i,\ell} t$	Fonction affine	(p. 125)
$\mathbb{S}$	Ensemble des serveurs	(p. 124)
$\text{Sol}(\mathcal{T})$	Ensemble des solutions d'un ensemble de contraintes temporelles	(p. 128)
$\mathcal{T}(\pi, \pi')$	La comparaison entre π et π' dans Π'. Peut prendre une valeur parmi : $=, <, >, \leq$ ou $\geq$	(p. 128)
$\mathcal{T}$	Contraintes temporelles dans un problème d'optimisation linéaire	(p. 128)
$\text{Tot}(\Pi)$	Ensemble de tous les ordres sur Π qui satisfont (P1) et (P2)	(p. 128)
t_π	Variable temporelle	(p. 127)

Notations spécifiques au chapitre 11

$1_A(t) = 1$	Fonction de test	(p. 159)
$\mathcal{BP}_{R,R'}$	Ensemble des périodes chargées pour la trajectoire (R, R')	(p. 164)
$\mathcal{F}, \mathcal{G}, \mathcal{F}_0$	Ensembles de fonctions	(p. 159)
$R \prec \alpha$	R admet α comme courbe d'arrivée	(p. 160)
$\mathbb{R}_{\geq 0}$	Réels positifs ou nuls	(p. 159)
$S \trianglerighteq \beta$, $S \triangleright \beta$	Le serveur S admet β comme courbe de service simple / strict	(p. 162)
$S \circ S'$	Mise en séquence de deux serveurs	(p. 165)
$\mathcal{S}_n$	Ensembles des serveurs à n entrées et n sorties	(p. 167)
$\mathbf{v}$	Vecteur v	(p. 159)
$[x]^+$	$\max(x, 0)$	(p. 159)

Première partie :

Introduction

CHAPITRE 1

NETWORK CALCULUS : RAPIDE PANORAMA

Origines

Le Network Calculus (NC) est une théorie qui a été développée pour calculer des bornes déterministes (ou pire-cas) sur les délais dans les réseaux de communication.

C'est Cruz qui a écrit les articles fondateurs de cette théorie : il a traité d'abord des calculs de délais et de charge dans les éléments de réseaux pris isolément [Cruz 1991a] ; puis de la mise en réseau des différents éléments [Cruz 1991b]. De nombreuses questions concernant la modélisation du trafic, la topologie du réseau ou la gestion de la taille des paquets de données sont déjà soulevées, mais le formalisme du Network Calculus n'est pas encore présent. Les applications initiales étaient le dimensionnement de réseaux de type IP (réseaux informatiques) ou ATM (réseaux téléphoniques commutés), ainsi que la maîtrise de la qualité de service.

Dans les années 2000, deux ouvrages sont publiés qui définissent le Network Calculus :

Le livre [Chang 2000] donne une présentation très formelle des opérateurs du NC et de leur utilisation dans le cas où le temps et les données sont des quantités discrètes. Une seconde partie est consacrée au Network Calculus stochastique.

Et [Le Boudec 2001] qui décrit tous les outils propres au Network Calculus ainsi que les conditions d'application et donne de nombreux exemples d'applications sur des problèmes très concrets tels que la qualité de service pour internet ou l'amélioration des applications multimédia. Ce livre est une ressource majeure pour quiconque s'intéresse au Network Calculus.

Une étude récente et très détaillée des dernières avancées du Network Calculus (ainsi que de sa version stochastique) a été publiée dans [Fidler 2010].

Principe

Avec des méthodes permettant de calculer des bornes sur les délais, la charge ainsi que d'autres paramètres de qualité de service, son but est d'analyser les comportements critiques et le NC s'intéresse pour cela principalement aux performances pire-cas : soit des performances locales (taille maximale d'un *buffer* au niveau d'un noeud du réseau), soit des performances de bout en bout (temps de trajet de bout en bout pire-cas). Les informations sur le système sont modélisées grâce à des fonctions, telles que les *courbes d'arrivée* qui modélisent le trafic ou les *courbes de service* qui quantifient le service garanti en chaque noeud du réseau.

Applications

Parmi les applications du Network Calculus, on trouve la qualité de service sur internet [Firoiu 2002], la certification des réseaux avioniques [Grieu 2004] l'analyse de *System on Chip* [Chakraborty 2003], les réseaux ethernets industriels [Skeie 2006], ou encore les réseaux critiques embarqués [Boyer 2008]. C'est une alternative à d'autres approches d'étude de performance pire-cas comme : les méthodes holistiques [Tindell 1994], l'approche par trajectoire [Martin 2005] ou le *model checking* [Hendriks 2006]. Il a *a priori* des avantages en termes de modularité et de passage à l'échelle qui permettent des études de réseaux complexes [Perathoner 2007].

Formalisme

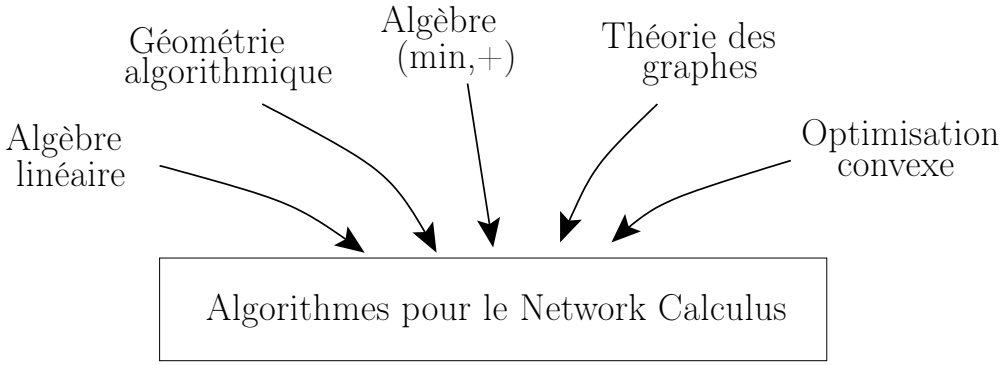
Dès le commencement (voir [Chang 1997, Cruz 1991a, Cruz 1991b]), les méthodes du Network Calculus ont mis en avant l'utilisation des algèbres tropicales $(\min, +)$ et $(\max, +)$, connues pour leurs applications aux *systèmes à événements discrets* [Baccelli 1992]. Les aspects max-plus du Network Calculus sont aussi abordés dans [Daniel-Cavalcante 2006, Daniel-Cavalcante 2008].

Les définitions de *courbe d'arrivée* et de *courbe de service simple* (minimum) peuvent s'exprimer simplement à l'aide de la convolution dans $(\min, +)$. Le NC permet ensuite soit de propager ces contraintes dans le réseau afin de calculer de proche en proche des bornes sur les performances, soit d'exprimer le comportement du réseau à l'aide d'un ensemble d'équations fonctionnelles dans $(\min, +)$, qu'il faut résoudre pour obtenir les bornes.

La convolution dans $(\min, +)$ capture de manière élégante le phénomène *Pay Burst Only Once* (PBOO, cf. [Fidler 2003]) pour une configuration de serveurs dite en tandem (les rafales sont amorties sur l'ensemble des serveurs). Cependant, dès que la configuration du réseau étudié fait intervenir plusieurs flux agrégés, une étude exacte (*tight*) des performances devient beaucoup plus difficile.

Stabilité et bornes

Les différentes utilisations du NC sont généralement classées en fonction de la topologie du réseau, de la politique de service, et du type de service garanti à chaque serveur. L'étude d'un réseau dans le cas général, où les flux peuvent avoir des dépendances cycliques, est un problème ouvert : une question apparemment simple comme décider de la *stabilité* du réseau (c'est-à-dire savoir si la charge globale ou les délais bout en bout restent bornés) n'est pas résolue pour de nombreuses politiques de service. On trouvera des résultats concernant la *stabilité* dans la littérature d'*Adversarial Queuing*, qui étudie le Permanent Sessions Model (proche des modèles étudiés en Network Calculus) [Borodin 2001]. Une condition nécessaire classique pour la stabilité d'un réseau est un facteur d'utilisation inférieur à 1 à chaque noeud. Cette condition est aussi suffisante pour les réseaux acycliques [Cruz 1991b], ou les anneaux unidirectionnels [Tassiulas 1996]. Mais cette condition n'est pas suffisante en général ou dans tous les cas pour la politique de service FIFO : il existe des réseaux instables pour des facteurs d'utilisation arbitrairement petits [Andrews 2007]. Dans le cas général pour les réseaux FIFO, les meilleures conditions suffisantes ainsi que les bornes et délais associés, se trouvent actuellement dans [Rizzo 2007, Rizzo 2008a, Rizzo 2008b] même si elles ne sont généralement pas *tight*. De plus, pour des réseaux acycliques simples comme les réseaux en tandem FIFO l'article [Bisti 2008], qui améliore les bornes connues, a montré que ces bornes n'étaient pas *tight* excepté pour les réseaux tandem *sink tree*.



.

CHAPITRE 2

DÉFINITIONS IMPORTANTES EN NETWORK CALCULUS

2.1. Cadre mathématique

2.1.1. Fonctions manipulées en Network Calculus

Principe

L'objectif premier du Network Calculus est l'évaluation de performance dans les réseaux de communication.

Nous allons nous intéresser aux mouvements des données dans le réseau, mais il est important de distinguer deux types d'objets :

- le mouvement réel des données dans le réseau ;
- les contraintes que ces mouvements vérifient (voir section 2.1.5).

Le mouvement réel des données est modélisé par des *fonctions cumulées* $t \to f(t)$, où $f(t)$ représente la quantité de données totale qui vérifie une condition donnée jusqu'au

temps t (par exemple, la quantité totale de données qui a traversé un élément du réseau), voir figure 2.1.

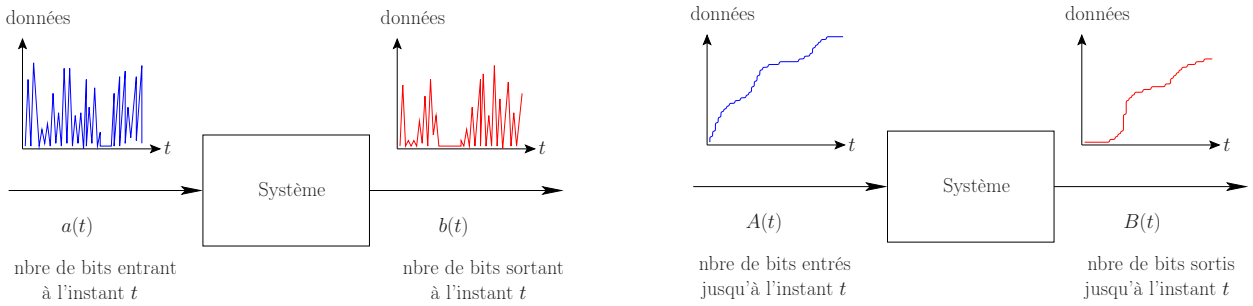


FIGURE 2.1 : Entrée/sortie : instantanées (à gauche), et cumulées (à droite). On a la relation $A(t) = \sum_{s=0}^{t} a(s)$ et idem pour B

Granularité

Les quantités de données présentes en différents points du réseau sont donc modélisées par des fonctions croissantes $t \mapsto f(t)$ où t est le *temps* et $f(t)$ une quantité de *données*. Il existe différents modèles selon que t (resp. $f(t)$) prend des valeurs discrètes (dans $\mathbb{N}$) ou continues (dans $\mathbb{R}_+$). Le temps est soit discret, soit continu. Pour les données, il existe trois granularités différentes :

- fluide (ou infinitésimale) :
  les données sont divisées en morceaux arbitrairement petits ($\mathbb{R}_+$) ;
- discrète (ou unitaire) :
  les données sont regroupées dans des paquets de même taille ($\mathbb{N}$) ;
- multi-échelle :
  les données peuvent être regroupées dans des paquets de taille variable ($\mathbb{N}$, ou $\mathbb{R}_+$).

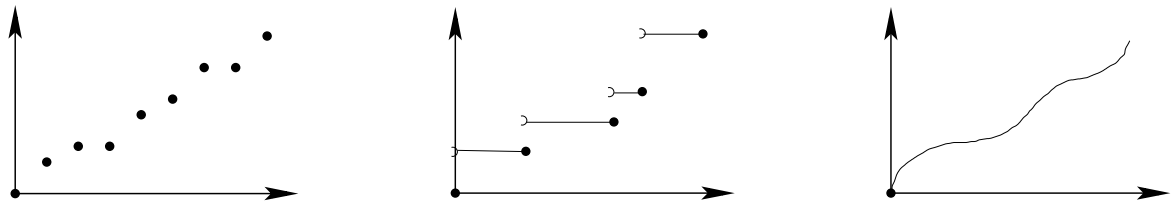


FIGURE 2.2 : Représentation graphique de fonctions : $\mathbb{N} \to \mathbb{N}$ (discret, à gauche), $\mathbb{R} \to \mathbb{N}$ (discret, au milieu), $\mathbb{R} \to \mathbb{R}$ (continue, à droite)

Dans tous les cas, on utilisera le terme « bit » comme unité de mesure.

Remarque 2.1. *Dans [Le Boudec 2001], le modèle* fluide *impose que les fonctions considérées soient continues. Ce n'est pas le cas ici.*

Continuité à gauche

Sauf mention contraire, *nous supposerons que les fonctions cumulées sont continues à gauche*. Ceci n'est pas très restrictif en termes de modélisation : par exemple, soit $f(t)$ la quantité totale de données entrée dans le système jusqu'au temps t, en cas de rafale instantanée de b bits au temps t_0 alors que a bits sont déjà arrivés, il suffit de poser $f(t_0) = a$ et $f(t_0+) = a+b$.

Prendre en compte cette hypothèse est néanmoins important en NC (en particulier lorsqu'on définira le début (ou *start*) des périodes chargées (section 2.1.5) avec des courbes de service strict comme au théorème 7.3). Au contraire, aucune contrainte de continuité n'est imposée pour les enveloppes.

Remarque 2.2. *Dans [Cruz 1991a], la question de la continuité à gauche ou à droite est abordée, en se demandant si l'arrivée d'un paquet — un ensemble de bits d'une certaine longueur — a lieu à l'instant où le premier bit du paquet commence à arriver, ou lorsque le dernier bit est lui aussi arrivée ; de même pour les départs. Cruz étudie aussi la possibilité de prendre en compte l'arrivée de chacun des bits du paquet dans le temps, ce qui ressemble au modèle fluide.*

Ensembles de fonctions

Nous noterons $\mathcal{F}$ (resp. $\mathcal{D}$) l'ensemble des fonctions — pas nécessairement croissantes — de $\mathbb{R}_+$ (resp $\mathbb{N}$) dans $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty; +\infty\}$:

$$\mathcal{F} = \left\{ f : \mathbb{R}_+ \to \overline{\mathbb{R}} \right\},$$

$$\mathcal{D} = \left\{ f : \mathbb{N} \to \overline{\mathbb{R}} \right\}.$$

Nous utilisons le *modèle fluide*, dans lequel le temps est continu, et les données peuvent être subdivisées en morceaux arbitrairement petits (les mesures de temps et de données se font dans $\mathbb{R}_+$), et les fonctions cumulées appartiennent à l'ensemble

$$\mathcal{F}_{\nearrow} = \{ f : \mathbb{R}_+ \to \mathbb{R}_+ \mid f \text{ croissante, continue à gauche, } f(0) = 0 \}.$$

Fonctions concaves, convexes, étoilées

Les définitions suivantes sont tirées de [Le Boudec 2001, p. 110,116], mais adapté pour les fonction suradditives.

Définition 2.1 (Concavité, convexité)**.** *Une fonction* $f : \mathrm{I} \to \mathbb{R}$ *est* convexe *si* $\forall x, y \in \mathrm{I}, \forall t \in [\mathrm{O}; 1]$, *on a*

$$f(t.x + (1-t)y) \leq t.f(x) + (1-t).f(y).$$

On dit qu'une fonction f est concave lorsque $-f$ est convexe.

Définition 2.2 (Fonction étoilée)**.** *Une fonction $f \in \mathcal{F}$ (croissante et nulle sur les négatifs) est* étoilée *si $t \to f(t)/t$ est monotone (au sens large) pour $t > 0$.*

Définition 2.3 (Fonction suradditive, sousadditive)**.** *Soit fonction $f \in \mathcal{F}$. On dit que f est suradditive ssi*

$$f(t+s) \geq f(t) + f(s), \forall s, t \geq 0$$

et elle est sousadditive ssi

$$f(t+s) \leq f(t) + f(s), \forall s, t \geq 0.$$

Théorème 2.1 (adapté de [Le Boudec 2001, 3.1.9])**.** *Une fonction étoilée de $\mathcal{F}_0$ (nulle en zéro) est suradditive si $t \to \frac{f(t)}{t}$ est croissante, et sousadditive si elle est décroissante.*

2.1.2. Dioïde min-plus

Mathématiquement, les fonctions considérées en Network Calculus seront manipulées grâce à des opérateurs qui trouvent leur origine dans l'étude des systèmes à événements discrets [Gaubert 1992, Gaubert 1999, Baccelli 1992, 'Max Plus']. On pourra se référer à [Gaubert 1999, p.14] pour les définitions, ou à [Gondran 2002] pour les propriétés des dioïdes.

Définition 2.4 (Semi-anneau)**.** *Un* semi-anneau *est un ensemble* E *muni de deux lois, la* somme $\oplus$ *et le* produit $\otimes$*, et qui contient deux éléments distincts notés* 0 *et* 1*, tel que :*

- $(\mathrm{E}, \oplus, 0)$ *est un monoïde (loi de composition interne associative munie d'un élément neutre) commutatif ;*
- $(\mathrm{E}, \otimes, 1)$ *est un monoïde ;*
- $\otimes$ *est distributif par rapport à* $\oplus$ *:* $\forall a, b, c \in \mathrm{E}, (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$ *(et idem à gauche) ;*
- 0 *est un élément absorbant pour* $\otimes$ *:* $\forall a \in \mathrm{E}, a \otimes 0 = 0 \otimes a = 0$.

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

Définition 2.5 (Dioïde $(\mathrm{D}, \oplus, \otimes, 0, 1)$)**.** *Un* dioïde *est un semi-anneau idempotent, c'est-à-dire que la loi $\oplus$ vérifie :*

$$\forall a \in \mathrm{D},\ a \oplus a = a.$$

Les notions d'anneau et de dioïde sont exclusives l'une de l'autre [Gaubert 1999, 1.2.4.15]. Supposons l'existence d'un dioïde dont chaque élément a admet un opposé $\overline{a}$. On a alors pour tout $a \in \mathrm{E}$, en utilisant l'associativité de $\oplus$, l'égalité des deux quantités suivantes :

- $(a \oplus a) \oplus \overline{a} = a \oplus \overline{a} = 0$ en utilisant l'idempotence et le symétrique ;
- $a \oplus (a \oplus \overline{a}) = a \oplus 0 = a$ en utilisant le symétrique et l'élément neutre de $\oplus$.

Le dioïde devrait donc être réduit au seul élément 0, alors qu'il doit comporter au moins deux éléments distincts : 0 et 1.

Utilisation en Network Calculus : le dioïde $(min, +)$

La structure $(\mathbb{R} \cup \{+\infty\}, min, +, +\infty, 0)$, souvent appelée « algèbre $(min, +)$ » est un dioïde. C'est sur le dioïde $(min, +)$ que l'on définit les opérations du Network Calculus. Il est commutatif

$$\forall a, b \in \mathrm{E},\ a \otimes b = b \otimes a$$

et sélectif

$$\forall a, b \in \mathrm{E},\ a \oplus b = a \text{ ou } b.$$

Remarque 2.3. *On remarquera en particulier que le neutre pour la loi min est $+\infty$, et qu'il est impossible de définir un symétrique (on est dans un dioïde, pas dans un anneau).*

Remarque 2.4. *Un dioïde est naturellement muni d'une relation d'ordre définie par*

$$\forall a, b \in \mathrm{D},\ a \preceq b \Leftrightarrow a \oplus b = b.$$

Notons que dans le dioïde $(min, +)$, on a donc $a \preceq b \Leftrightarrow min(a, b) = b$, et donc

$$a \preceq b \Leftrightarrow a \geq b.$$

Dans la suite nous travaillerons avec l'ordre usuel.

On pourra consulter [Le Corronc 2010, Le Corronc 2011] sur le dioïde $(min, +)$ et le Network Calculus.

2.1.3. Opérateurs

Le Network Calculus utilise abondamment plusieurs opérateurs classiquement employés dans le dioïde (min, +) [Baccelli 1992]. Les fonctions étudiées sont souvent supposées croissantes, mais par souci de généralité nous accepterons des fonctions qui ne sont pas nécessairement croissantes et qui prennent leurs valeurs dans $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$.

Définition 2.6. *Soit $\mathrm{X} = \mathbb{N}$ ou $\mathbb{R}_+$ et f, g deux fonctions de X dans $\overline{\mathbb{R}}$. Le NC utilise fréquemment les opérations suivantes :*

- *Minimum :*

$$\forall t \in \mathrm{X},\ \min(f, g)(t) = \min(f(t), g(t)).$$

Nous utiliserons aussi la notation infixe $\oplus$: $f \oplus g = \min(f, g)$;

- *Addition :*
$$\forall t \in \mathrm{X},\ (f+g)(t) = f(t) + g(t);$$
- *(Inf-)convolution (ou convolée) :*
$$\forall t \in \mathrm{X},\ (f * g)(t) = \inf_{0 \le s \le t} (f(s) + g(t-s));$$
- *(Sup-)déconvolution (ou déconvolée) :*
$$\forall t \in \mathrm{X},\ (f \oslash g)(t) = \sup_{u \ge 0} (f(t+u) - g(u));$$
- *Clôture sous-additive :*
$$\forall t \in \mathrm{X},\ f^*(t) = \inf_{n \ge 0} f^{(n)}(t)$$
où $f^{(n)}(t) = \underbrace{(f * \cdots * f)}_{n\ fois}(t)$ *pour* $n \ge 1$, *et* $f^{(0)}(t) = 0$ *si* $t = 0$ *et* $+\infty$ *si* $t > 0$.

Les propriétés algébriques de ces opérateurs sont étudiées en détail dans [Baccelli 1992, Le Boudec 2001] : avec ces définitions, la structure $(\mathcal{F}, min, *)$ est aussi un dioïde.

Remarque 2.5. *La déconvolution est un opérateur très particulier, et souvent peu intuitif. En effet, soient deux fonctions* $f, g \in \mathcal{F}$ *: les valeurs* $(f \oslash g)(x)$ *pour de « petites » valeurs de* x *dépendent directement du comportement, parfois difficile à appréhender, de* f *et* g *en* $+\infty$.

D'autres opérations apparaissent parfois en NC :
- Sup-convolution : $(f \overline{*} g)(t) = \sup_{0 \le s \le t}(f(s) + g(t-s))$.
- Inf-déconvolution : $(f \overline{\oslash} g)(t) = \inf_{u \ge 0}(f(t+u) - g(u))$.
- Partie positive : $f_+(t) = \max(f(t), 0)$.
- Clôture positive croissante : $f_\uparrow(t) = \max(\sup_{0 \le s \le t} f(s), 0)$.
- Clôture suradditive : $f^{\overline{*}}(t) = \sup_{n \ge 0} f^{(n)}(t)$

On peut définir de même le maximum (max) et la soustraction (−) de deux fonctions.

Valeurs infinies et ensembles de définition

Soit $f \in \mathrm{X} = \mathcal{D}$ ou $\mathcal{F}$. L'ensemble $Supp(f) = \{t \in \mathrm{X} \mid |f(t)| < +\infty\}$ est appelé le *support* de f. Tout d'abord, remarquons que les résultats obtenus par application des opérations ci-dessus sont bien définis à moins qu'une valeur infinie n'intervienne. Nous considérons en effet que $(+\infty) + (-\infty)$, $(+\infty) - (+\infty)$ et $(-\infty) - (-\infty)$ ne sont pas définis, et toute application d'une opération sur deux fonctions dont la définition fait intervenir un de ces cas donnera un résultat non défini. Vérifier si une combinaison de fonctions et d'opérations est définie ou non est facile (d'un point de vue mathématique comme algorithmique).

Pour tout $f, g \in \mathcal{D}$ ou $\mathcal{F}$, $\min(f, g)$ et $\max(f, g)$ sont toujours définis ; $f + g$ est indéfini si $\exists t$, $f(t) = +\infty$ et $g(t) = -\infty$ (ou le contraire) ; $f - g$ est indéfini si $\exists t$, $f(t) = g(t) = +\infty$

(ou $-\infty$) ; $f * g$ est indéfini si $\exists t_1, t_2,\ f(t_1) = +\infty$ et $g(t_2) = -\infty$ (ou le contraire) ; $f \oslash g$ est indéfini si $\exists t_1 \leq t_2,\ f(t_2) = g(t_1) = +\infty$ (ou $-\infty$) ; f^* est indéfini si $\exists t_1, t_2,\ f(t_1) = +\infty$ et $f(t_2) = -\infty$. Dans la suite de cette thèse, lorsque nous écrirons une formule, nous supposerons que les conditions pour que cette formule soit bien définie sont vérifiées.

Clôture sous-additive : définitions équivalentes

Si $f(0) < 0$, alors $\forall t \in \mathrm{X},\ f^*(t) = -\infty$.

Lorsque $f \in \mathcal{D}$ ou $\mathcal{F}$, et $f(0) \geq 0$, la clôture sous-additive peut, de manière équivalente, être définie par $f^*(0) = 0$ et, pour tout $t > 0$,

$$f^*(t) = \inf_{k \in \mathbb{N},\ t_1,\ldots,t_k > 0,\ t_1+\cdots+t_k = t} (f(t_1) + \cdots + f(t_k)). \tag{2.1}$$

Lorsque $f \in \mathcal{D}$ (cas discret) et $f(0) \geq 0$, la clôture sous-additive peut aussi être définie récursivement par : $f^*(0) = 0$ et, pour tout $t > 0$,

$$f^*(t) = \min[f(t), \min_{0<s<t}(f^*(s) + f^*(t-s))]$$

(voir [Chang 2000]).

Interprétation géométrique de la convolution

La convolution $f * g$ est définie comme

$$\forall t \in \mathrm{X},\ (f * g)(t) = \inf_{0 \leq s \leq t} (f(s) + g(t-s)).$$

La courbe représentative de $f * g$ est l'enveloppe inférieure des « courbes » $t \to f(s) + g(t-s)$ pour s fixé [Le Boudec 2001, p19]. Cela revient à fixer la courbe de f, à faire « glisser » celle de g dessus, puis à prendre l'enveloppe inférieure. Plus précisément, pour tout point $\mathrm{P}(x, f(x))$ de C_f, la courbe représentative de f, on considère la courbe $\mathrm{C}_{g,x}$ qui est la courbe C_g à laquelle on a appliqué une translation $\overrightarrow{\mathrm{OP}}$. Puis on prend l'enveloppe inférieure de cet ensemble de points. On pourra trouver une illustration — dans le cas où les deux fonctions sont croissantes — figure 4.9.

Lien avec la théorie de la résiduation

Il est intéressant de remarquer que les opérateur de *déconvolution* et de *clôture sous-additive* peuvent être définis à partir de la convolution, comme des solutions d'(in)équations fonctionnelles :

Ainsi, la déconvolution $f \oslash g$ est la plus grande solution, dans le dioïde $(\mathcal{F}, min, *)$, de

l'inéquation

$$x * g \leq f$$

où $\leq$ est l'ordre du dioïde, cf 2.4.

De même, la clôture sous-additive f^* est la plus petite solution de l'équation

$$x = min(0, f * x)$$

où 0 est la fonction nulle (avec les notations du dioïde : $x = 0 \oplus (f \otimes x)$).

D'autres résultats intéressants, ainsi qu'une présentation très claire du Network Calculus d'un point de vue de la théorie de la résiduation peuvent être trouvés dans [Le Corronc 2011, 3.2, et p. 57].

2.1.4. Modéliser un réseau : système entrée/sortie

Un réseau $\mathcal{N}$ est généralement modélisé par un graphe dans lequel les serveurs sont représentés par les sommets du graphe, et les flux de données doivent suivre les arêtes du graphe (un flux traverse généralement plusieurs sommets et arêtes). La modélisation d'un réseau de communication en NC consiste donc généralement en :

- la partition du réseau en sous-systèmes (souvent appelée *noeuds*) qui peuvent correspondre à plusieurs échelles (du petit élément matériel comme un processeur, jusqu'à un grand sous-réseau) ;
- la description des *flux* de données, où chaque flux suit un chemin prédéterminé à travers une suite de *noeuds* et/ou chaque flux est contraint par une *courbe d'arrivée* juste avant d'entrer dans le réseau ;
- la description du comportement de chaque noeud : d'une part des courbes de service qui donnent des bornes sur les performances du noeud ; d'autre part la description de la politique de service en cas de multiplexage (lorsque plusieurs flux se partagent le service offert par un noeud).

Remarque 2.6. *Chaque système ou sous-système est défini comme un système entrée/sortie où le nombre d'entrées est le même que le nombre de sorties.*

Définition 2.7 (Trajectoire acceptable — plusieurs flux)**.** *Une* trajectoire (acceptable) *pour un système traversé par p flux est l'ensemble* $((A_k)_{1\leq k\leq p}, (B_k)_{1\leq k\leq p})$ *où les* A_k *et les* B_k *appartiennent à* $\mathcal{F}_{\nearrow}$*, et correspondent respectivement aux fonctions cumulées pour le flux k à l'entrée et à la sortie du système.*

Ainsi, dans le cas général, un *système* $\mathcal{S}$ de p flux sera défini par l'ensemble de toutes ses trajectoires acceptables, et on aura $\mathcal{S} \subseteq \mathcal{F}_{\nearrow}^p \times \mathcal{F}_{\nearrow}^p$. Voir un système comme une boîte noire est habituel en théorie du filtrage classique et permet de manipuler des systèmes d'une échelle quelconque.

Cette définition permet d'étudier des *dynamiques déterministes* mais aussi des *dynamiques non déterministes*.

La définition suivante, limitée *a priori* à un seul flux, permettra aussi de travailler sur des systèmes traversés par *plusieurs* flux agrégés.

Définition 2.8 (Système entrée/sortie — un flux)**.** *Un* système entrée/sortie *est un sous-ensemble* S *de* $\{(F^{in}, F^{out}) \in \mathcal{F}_{\nearrow} \times \mathcal{F}_{\nearrow} \mid F^{in} \geq F^{out}\}$.

Un système entrée/sortie modélise un flux traversant un système où F^{in} (resp. F^{out}) est la fonction cumulée à l'entrée (resp. la sortie) du système et $F^{in} \geq F^{out}$ indique que le système fait uniquement de la transmission de données. Le système peut aller d'un simple serveur à un grand réseau avec du trafic transverse. Une *trajectoire* du système S est un élément (F^{in}, F^{out}) de S.

2.1.5. Enveloppes : courbe d'arrivée et courbe de service

Les contraintes sur les flux dans le réseau sont modélisées par des enveloppes :

- les courbes d'arrivée, qui permettent de modeler le trafic en entrée du système étudié ;
- les courbes de service, qui décrivent le comportement du système ou du serveur.

Courbes d'arrivée : une seule définition

Étant donné un flux traversant un réseau, soit $A \in \mathcal{F}_{\nearrow}$ sa *fonction cumulée* en un certain point du réseau, c'est-à-dire $A(t)$ représente le nombre de bits qui ont traversé ce point jusqu'au temps t, avec $A(0) = 0$. On dit qu'une fonction $\alpha \in \mathcal{F}$ est une *courbe d'arrivée* pour A si : $\forall\, s, t \in \mathbb{R}_+, 0 \leq s \leq t$,

$$A(t) - A(s) \leq \alpha(t - s).$$

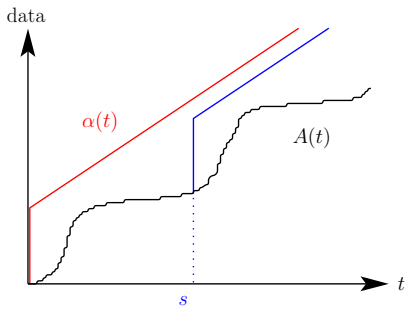


FIGURE 2.3 : Pour tout intervalle de temps $[s;t], s \leq t$, la quantité de données arrivées entre s et t est bornée par $\alpha(t-s)$

L'ensemble de toutes les courbes d'arrivée pour $A \in \mathcal{F}_{\nearrow}$ admet un minimum qui est

aussi une courbe d'arrivée pour A. On peut la définir [Le Boudec 2001, p. 14] par :

$$\alpha = A \oslash A,$$

que nous appellerons la *courbe d'arrivée canonique* pour A.

Exemple de courbe d'arrivée

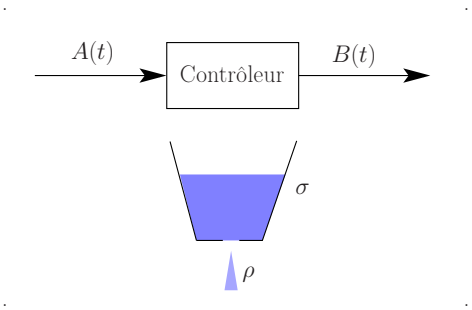


FIGURE 2.4 : Seau percé, ou *leaky-bucket*

Un exemple typique de courbe d'arrivée est la fonction affine :

$$\alpha_{\sigma,\rho}(t) = \sigma + \rho t, \text{ avec } \sigma, \rho \in \mathbb{R}_+$$

(parfois appelée seau percé ou *leaky-bucket*, voir figure 2.4), où σ représente le nombre maximal de bits qui peuvent arriver simultanément (la rafale maximale), et ρ est le débit maximal des arrivées sur le long terme.

Courbes de service : plusieurs définitions

Les courbes de service que nous considérerons auront généralement les caractéristiques suivantes :

- *courbes de service minimum* qui sont des bornes inférieures sur le service fourni dans le système (elles permettent d'obtenir des bornes supérieures sur les performances pire-cas) ;
- *systèmes à un seul flux* $\mathcal{S}$, ou avec plusieurs flux agrégés, et donc $\mathcal{S} \subseteq \mathcal{F}_{\nearrow} \times \mathcal{F}_{\nearrow}$.

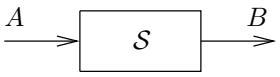


FIGURE 2.5 : Un système entrée/sortie à un flux.

Il existe plusieurs définitions de courbes de service, qui ne sont pas équivalentes.

Pour chaque type $\mathcal{T}$ de courbes de service, nous définissons pour tout $\beta \in \mathcal{F}$ et toute trajectoire entrée/sortie $(A, B) \in \mathcal{F}_{\nearrow} \times \mathcal{F}_{\nearrow}$ les conditions telles que β est une $\mathcal{T}$-courbe

de service (A, B) (nous dirons aussi que (A, B) admet β comme $\mathcal{T}$-courbe de service, ou que (A, B) est une trajectoire acceptable). Nous définissons ensuite pour tout $\beta \in \mathcal{F}$, $\mathcal{S}_{\mathcal{T}}(\beta)$ l'ensemble de toutes les trajectoires qui admettent β comme $\mathcal{T}$-courbe de service. Nous disons alors qu'un système $\mathcal{S}$ admet β comme $\mathcal{T}$-courbe de service si c'est le cas pour toutes ses trajectoires, c'est-à-dire $\mathcal{S} \subseteq \mathcal{S}_{\mathcal{T}}(\beta)$. Les cinq premières définitions sont adaptées de [Le Boudec 2001, Schmitt 2006a] ; et celle de *courbe de service suffisamment strict* de [Schmitt 2011].

- Courbe de service simple (ou *courbe de service*) :

$$\mathcal{S}_{simple}(\beta) = \{(A, B) \in \mathcal{F}_{\nearrow} \times \mathcal{F}_{\nearrow} \mid A \geq B \geq A * \beta\}.$$

- Courbe de service strict :

$$\mathcal{S}_{strict}(\beta) = \{(A, B) \in \mathcal{F}_{\nearrow} \times \mathcal{F}_{\nearrow} \mid A \geq B, \text{ et } \forall \mathcal{BP}]s, t], \ s \leq t, \ B(t) - B(s) \geq \beta(t - s)\}.$$

  où $\mathcal{BP}$ est une *période chargée*, c'est-à-dire un intervalle $I \subseteq \mathbb{R}_+$ tel que

$$\forall u \in I, \ A(u) - B(u) > 0.$$

  Pour tout $t \in \mathbb{R}_+$, on appellera *début* de la période chargée de t, noté $start(t)$, l'instant

$$start(t) = \sup\{u \leq t \mid A(u) = B(u)\}.$$

  Comme A et B sont continues à gauche, on a : $A(start(t)) = B(start(t))$. Si $A(t) = B(t)$, alors $start(t) = t$. Par exemple, pour tout $t \in \mathbb{R}_+$, $]start(t), t[$ est une période chargée, de même que $]start(t), t]$ si $A(t) - B(t) > 0$.
- Courbe de service strict (au sens faible) :

$$\mathcal{S}_{wstrict}(\beta) = \{(A, B) \in \mathcal{F}_{\nearrow} \times \mathcal{F}_{\nearrow} \mid A \geq B, \text{ et } \forall t \geq 0, \ B(t) - B(start(t)) \geq \beta(t - start(t))\}.$$

- Variable Capacity Node (VCN) (définition différente de la présentation originale, cf. [Le Boudec 2001, p. 22]) :

$$\mathcal{S}_{vcn}(\beta) = \{(A, B) \in \mathcal{F}_{\nearrow} \times \mathcal{F}_{\nearrow} \mid \exists C \in \mathcal{F}_{\nearrow}, \forall t \geq 0,$$

$$B(t) = \inf_{0 \leq s \leq t} \left[A(s) + C(t) - C(s)\right] \text{ et } \forall 0 \leq s \leq t, \ C(t) - C(s) \geq \beta(t - s)\}.$$

- Courbe de service adaptatif (d'abord définie pour deux courbes β et β', puis spécialisé

pour une seule courbe, voir [Le Boudec 2001]) :

$$\mathcal{S}_{asc}(\beta) = \{(A, B) \in \mathcal{F}_{\nearrow} \times \mathcal{F}_{\nearrow} \mid A \geq B \text{ et}$$

$$\forall t \in \mathbb{R}_+, \forall s \leq t, B(t) \geq \min[B(s) + \beta(t-s), \inf_{s \leq u \leq t} A(u) + \beta(t-u)]\}$$

- Courbe de service suffisamment strict — S3C [1] :

$$\mathcal{S}_{S3C}(\beta)\{(A, B) \in \mathcal{F} \times \mathcal{F} \mid \forall t \in \mathbb{R}_+, A(t) \geq B(t) \geq A(t - D(t)) + \beta(D(t))\}$$

avec $D(t)$, la *maximum achievable dwell period* [2] au temps t, définie par $D(t) = t - t_0(t)$ où $t_0(t)$ est l'instant d'arrivée du plus ancien bit parmi tous les bits dans le système à l'instant t, pour tous les ordres de traitement possibles.

Définition 2.9 (Courbe de service universelle [Chang 2000])**.** *De plus, on dit qu'une courbe de service est* universelle *si elle est une courbe de service pour tout flux entrant. Autrement dit si elle est indépendant de* α*.*

Remarquons que toutes les courbes de services définies ci-dessus sont des courbes de service minimal. Rien n'empêche les serveurs de se comporter comme un « fil », et de transmettre toutes les données qui arrivent instantanément.

Définition 2.10 (Serveur exact (simple))**.** *On parlera de service exact, ou plutôt de* serveur exact *lorsque le serveur fournit exactement son service simple minimum :*

$$B = A * \beta$$

On n'a pas abordé ici la question des courbes de service *maximum* ([Le Boudec 2001], ou [Wandeler 2006a] dans le cadre du Real-Time Calculus), car elles ne sont pas étudiées dans la suite de ce manuscrit. Néanmoins nous sommes conscient qu'elles peuvent être d'un grand intérêt, tant théorique que pratique.

Remarque 2.7. *Il est tout à fait possible d'utiliser des courbes de service qui ne sont pas dans* $\mathcal{F}_{\nearrow}$*, par exemple des fonctions qui prennent des valeurs négatives, qui sont décroissantes ou qui ont des discontinuités. Néanmoins il est pratiquement nécessaire d'imposer* $\beta(0) \leq 0$*, sans quoi on aurait* $\mathcal{S}_{simple}(\beta) = \mathcal{S}_{wstrict}(\beta) = \mathcal{S}_{strict}(\beta) = \mathcal{S}_{vcn}(\beta) = \emptyset$*.*

Remarquons que dans les définitions précédentes, c'est le *service* qui est simple ou strict, et pas la courbe β. Le type de service décrit comment se comporte le serveur, et la courbe

1. Les notations *SC* pour *service curve* (simple), *S2C* pour *strict service curve* et *S3C* pour *sufficiently strict service curve* sont utilisées dans [Schmitt 2011].
2. que l'on pourrait traduire par « période d'occupation maximale »

permet de décrire les contraintes pour ce comportement. La courbe β n'est pas tantôt stricte, tantôt simple : elle ne change pas. Par abus de langage, il nous arrivera de dire « β est une courbe stricte pour le serveur » pour dire « le serveur fournit un service strict de paramètre la courbe β ».

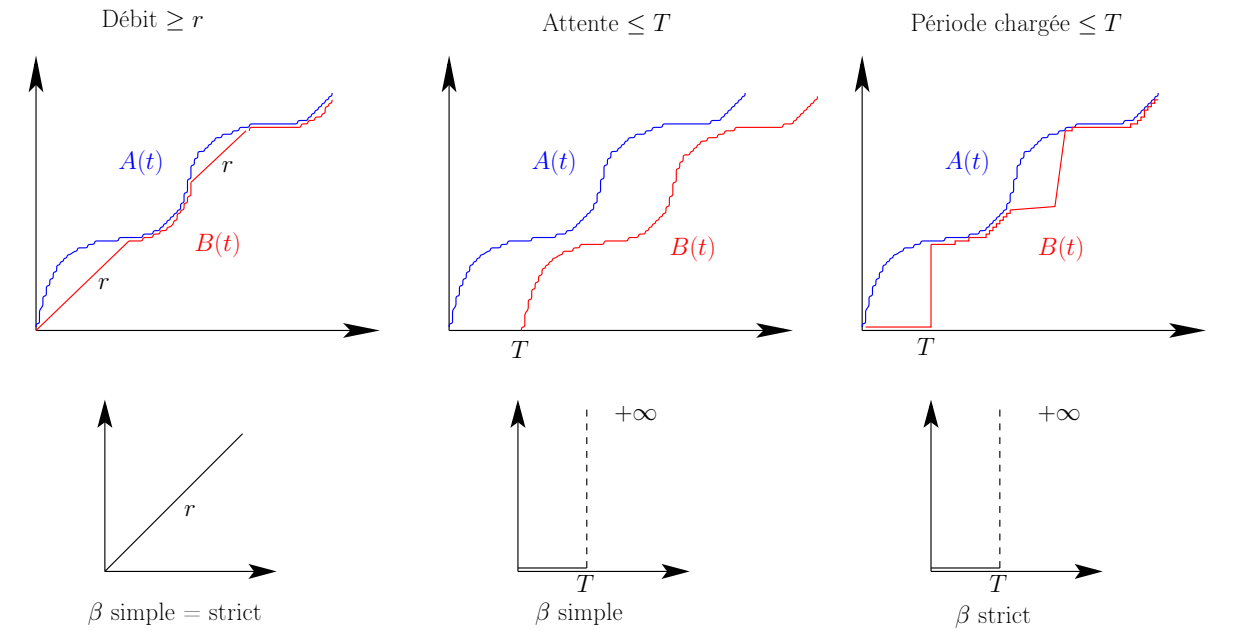


FIGURE 2.6 : Attention, la même courbe δ_T (au milieu et à droite) n'accepte pas les mêmes trajectoires selon le type de service considéré

Exemples de fonctions fréquemment utilisées comme courbes de service

Attention, les fonctions ci-dessous ne permettent pas à elles seules de définir le comportement du serveur. C'est avant tout le type de service qui le définit (voir figure 2.6).

Fonctions classiques souvent utilisées comme courbes de service :

- *délai pur* $T \in \mathbb{R}_+ \cup \{+\infty\} : \delta_T(t) = 0$ si $t \leq T$, $= +\infty$ sinon ;
- *débit constant* $r \in \mathbb{R}_+ \cup \{+\infty\} : \lambda_r(t) = rt$ (si $r = +\infty$, on définit $\lambda_r = \delta_0$) ;
- *latence-débit* : $\beta_{R,T}(t) = R(t-T)_+$ où $R, T \in \mathbb{R}_+$ et a_+ représente $\max(a, 0)$.

2.1.6. Mesures de performance

L'objectif du Network Calculus est d'obtenir des bornes pire-cas sur les performances du réseau étudié.

Une des premières questions qui peut se poser est la question de la *stabilité* du réseau, à savoir si tous les délais subis par les données dans le réseau sont finis, et si les files d'attente sont suffisamment dimensionnées.

Performances caractéristiques et bornes : délai et charge

Dans un système entrée/sortie, des bornes sur la charge pire-cas et sur le délai pire-cas peuvent être déterminées simplement à partir des courbes d'arrivée et de service.

Étant donné un flux traversant un réseau modélisé par un système entrée/sortie S, notons (F^{in}, F^{out}) sa trajectoire [3] pour S. La *charge* pour ce flux au temps t est donnée par

$$b(t) = F^{in}(t) - F^{out}(t),$$

le *délai* subi par les données arrivées au temps t (avec une politique de service FIFO) est

$$\begin{aligned} d(t) &= \inf\{s \geq 0 \mid F^{in}(t) \leq F^{out}(t+s)\} \\ &= \sup\{s \geq 0 \mid F^{in}(t) > F^{out}(t+s)\}. \end{aligned}$$

Remarquons que $d(t) \neq \sup\{s \geq 0 \mid F^{in}(t) \geq F^{out}(t+s)\}$. Par exemple, considérons l'instant $t = 1$ pour la trajectoire $F^{in}(t) = t$, et $F^{out}(t) = 0$ sur $[0,2]$ et $= 1$ sur $]2,4]$ et $= t$ sur $]4,+\infty[$.

Pour cette trajectoire, la *charge pire-cas* est

$$B_{\max} = \sup_{t \geq 0} \left(F^{in}(t) - F^{out}(t) \right)$$

et le *délai pire-cas* est

$$D_{\max} = \sup_{t \geq 0} d(t) = \sup\{t - s \mid 0 \leq s \leq t \text{ et } F^{in}(s) > F^{out}(t)\}.$$

Pour le système S, la *charge pire-cas* (resp. *délai*) est le supremum sur toutes les trajectoires.

Le théorème suivant montre comment l'on peut obtenir des bornes sur les performances du réseau, et comment les contraintes de trafic peuvent être propagées dans le réseau.

Théorème 2.2 ([Chang 2000, Le Boudec 2001]). *Soit* S *un système entrée/sortie fournissant une courbe de service simple* β *et soit* (F^{in}, F^{out}) *une trajectoire telle que* α *est une courbe d'arrivée pour* F^{in}. *Alors on a :*

1. $B_{\max} \leq \sup\{\alpha(t) - \beta(t) \mid t \geq 0\} = v(\alpha, \beta)$ *(distance verticale) ;*
2. $D_{\max} \leq \inf\{d \geq 0 \mid \forall t \geq 0,\ \alpha(t) \leq \beta(t+d)\} = h(\alpha, \beta)$ *(distance horizontale) ;*
3. $\alpha \oslash \beta$ *est une courbe d'arrivée pour* F^{out}.

La figure 2.7 illustre les bornes B_{max} et D_{max}. La figure 2.8 illustre le dernier point, à savoir la propagation de la courbe d'arrivée en sortie du serveur.

3. Pour les trajectoires, nous utilisons la notation (A, B) pour insister sur les opérateurs utilisés, et la notation (F^{in}, F^{out}) pour insister sur le caractère entrée/sortie du système.

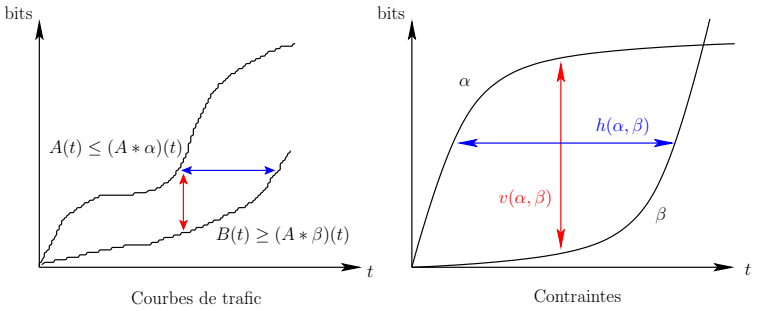


FIGURE 2.7 : Les contraintes sur les courbes de trafic sont modélisées par des enveloppes α et β qui permettent de trouver des bornes sur le délai et la charge maximum

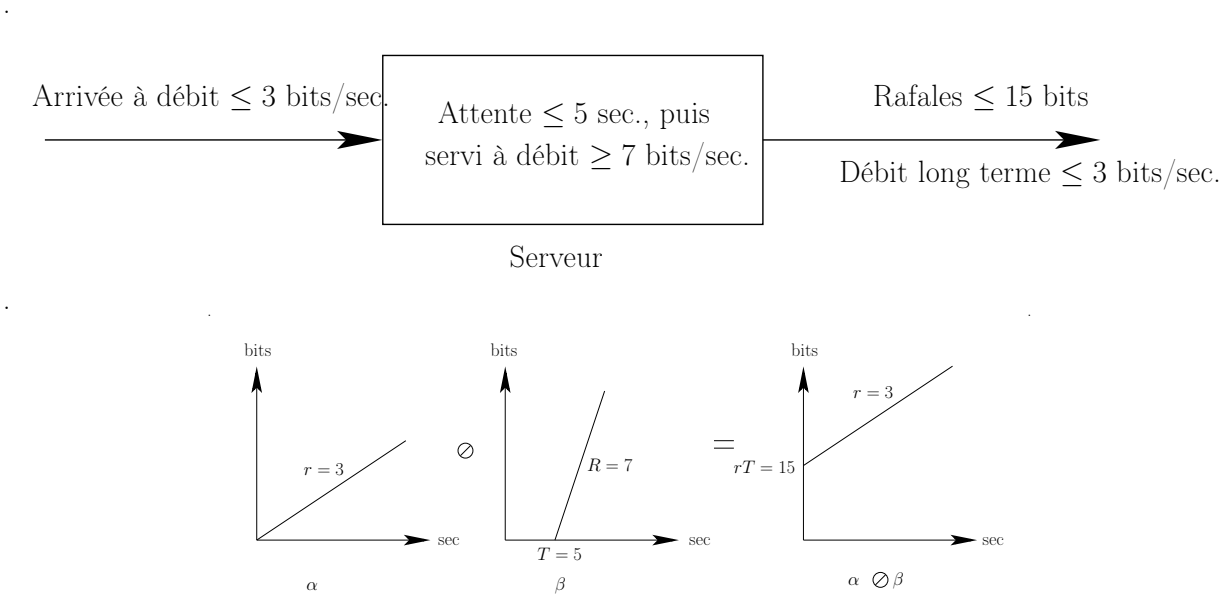


FIGURE 2.8 : Un système entrée/sortie et les contraintes en termes de débit (en haut) ; les contraintes exprimées en termes de fonctions en NC, et le calcul de la nouvelle courbe d'arrivée propagée en sortie du système (en bas)

Bit/flux observé

On souhaite généralement donner des bornes pire-cas globales sur le réseaux. Cependant, il arrive fréquemment de s'intéresser à un flux en particulier on parlera alors du *flux observé*), voire à un bit en particulier (on parlera du *bit observé*).

Bornes « tight »

Comme vu précédemment, le Network Calculus permet d'obtenir des bornes supérieures sur certaines caractéristiques du réseau, comme les délais subis par les données ou la charge au niveau de chaque serveur. Il est évidemment intéressant d'obtenir les bornes les plus précises possibles : le terme « tight » est traditionnellement utilisé. Nous avons choisi de garder ce terme anglais dans ce tapuscrit. En effet, il revêt plusieurs sens, et plusieurs traductions sont donc possibles. Une borne tight peut ainsi être dite :

- serrée : c'est la traduction littérale, mais elle n'est pas assez précise ;
- juste, exacte : cela sous-entendrait que c'est *la* meilleure borne possible, et qu'il n'y en a pas d'autre. Mais les bornes calculées dépendent aussi du modèle utilisé pour les obtenir ;
- atteignable : ce n'est pas toujours vrai, ou alors difficilement démontrable dans certains cas. La plupart des résultats obtenus ne nous permettent pas d'affirmer qu'il existe une trajectoire qui réalise la borne pire-cas ;
- non améliorable : c'est finalement la traduction qui serait la plus précise...

Dans la littérature, le terme « tight » est utilisé selon les contextes dans différents sens. Souvent dans le sens « non améliorable » pour un modèle, ou dans le sens « atteignable » ; et il est généralement précisé lorsque les bornes sont effectivement atteignables.

On pourra se référer à la section 11.3 pour comprendre la distinction entre bornes *atteintes* et *atteignables*.

2.2. Caractéristiques d'un réseau

2.2.1. Topologies

La topologie est la définition de l'architecture du réseau, c'est-à-dire l'ensemble des restrictions qui s'appliquent soit au graphe sous-jacent, soit à la disposition des flux dans le réseau. Étudier un réseau dans le cas général est souvent très compliqué. La plupart des résultats de Network Calculus se limitent donc à des topologies plus simples. On pourra se référer à [Lenzini 2007, Lenzini 2008].

Réseaux « acycliques » / sans dépendances cycliques : (ou *feed-forward*) On peut numéroter les serveurs dans $\mathbb{N}$ de telle sorte que lorsqu'il existe un flux qui va du noeud i au noeud j, alors on a $i < j$;

Réseau en tandem : (ou *serveurs mis en tandem*) Réseau $\mathcal{N}$ dont le graphe orienté induit est un chemin orienté sans raccourcis (donc sur une ligne). On observe généralement un flux principal qui peut être rejoint puis quitté (une seule fois) par des flux transverses.

Réseaux sink-tree : On observe un flux principal qui traverse des serveurs numérotés $1, 2, .., n$. Des flux transverses peuvent rejoindre le flux principal, mais restent agrégés avec lui dans tous les serveurs jusqu'au serveur n ;

Nested flow : (flux imbriqués) On observe encore un flux principal sur une ligne, qui peut être rejoint puis quitté par des flux transverses sur un sous-chemin, mais si deux flux transverses interfèrent avec le flux principal respectivement sur les noeuds $e_1, \ldots, s_1$ et $e_2, \ldots, s_2$, on doit avoir : $e_1 \leq e_2 < s_2 \leq s_1$ ou $e_2 \leq e_1 < s_1 \leq s_2$;

Réseau en diamant : Un exemple très simple de réseau sans dépendance cyclique, à quatre serveurs et deux flux (figure 9.2).

Cas général : étudier un réseau qui peut comporter des dépendances cycliques est à l'heure actuelle un problème ouvert, sauf dans certains cas très précis. Il peut donc être intéressant d'essayer de se ramener au cas acyclique, quitte à « supprimer » certains liens dans le réseau. C'est le but par exemple des études sur la *turn prohibition* [Fidler 2003].

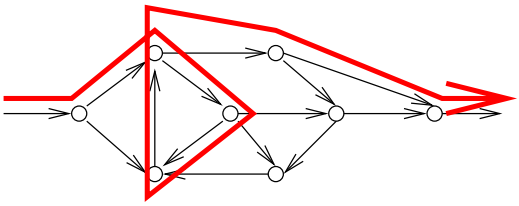


FIGURE 2.9 : Lorsqu'il y a des dépendances cycliques, étudier un réseau est encore plus compliqué

2.2.2. Politiques de service

La politique de service, la plupart du temps au niveau d'un serveur, détermine l'ordre dans lequel les paquets — ou les bits d'information — présents dans la file d'attente du serveur vont être servis.

On parle de multiplexage lorsque plusieurs flux entrent dans un seul serveur et qu'il sont agrégés : ils sont « mélangés » et traités comme un seul flux, mais il faut tout de même déterminer un ordre pour les différents paquets des flux de départ.

Les différentes « politiques de service » ci-dessous n'ont pas toutes le même statut : certaines déterminent quel est le prochain paquet à traiter, d'autres donnent simplement

des contraintes sur ce que doit être l'ordre de traitement des paquets. La dernière — multiplexage aveugle ou *blind multiplexing* — ne donne pas d'information sur l'ordre de service des paquets, mais sur la façon dont on pourra étudier la politique de service elle-même.

FIFO : (*F*irst In First Out) Premier arrivé, premier servi. Tous les paquets sont servis dans leur ordre de leur arrivée au niveau du serveur.

FIFO par flux : À l'intérieur d'un même flux, les paquets sont servis dans l'ordre d'arrivée. Mais si plusieurs flux sont traités par le serveur au même moment, il n'y a aucune garantie sur le flux qui sera traité en premier. Le serveur peut utiliser une autre politique de service que FIFO pour ordonner des paquets de flux différents.

Fixed Priority : (ou *static priority*) Les flux ont chacun une priorité, et c'est le flux le plus prioritaire qui est servi en premier. Dans le cas où l'on considère que l'arrivée des paquets n'est pas instantanée, ce type d'ordonnancement peut être préemptif (pour une étude dans le cadre du NC en non-préemptif, voir [Sofack 2011]).

Fair Queuing : Un serveur de débit maximum R qui doit servir n flux va fournir à chaque flux un débit R/n, par exemple en les servant alternativement.

Weighted Fair Queuing : Comme en *fair qeueing*, mais les différents flux se voient assigner des poids w_i, et le service qu'ils reçoivent les uns les autres est proportionnel à ce poids.

FIFO* C'est une contrainte globale sur l'ensemble d'un système : les bits doivent sortir du sytème dans l'ordre de leur arrivée dans le système. À l'intérieur du système, les serveurs ne servent pas obligatoirement en FIFO.

TDMA (*Time Division Multiple Access*) Les flux, ou les paquets, ne peuvent être servis que dans des intervalles de temps déterminés à l'avance. Le service du serveur est donc « découpé » entre les différents flux. Il existe d'autres technique de partage du medium, comme par exemple par fréquences distinctes (FDMA) ou *Code Division Multiple Access*.

Multiplexage aveugle (multiplexage arbitraire, *blind multiplexing*) On suppose qu'on n'a pas d'information sur la politique de service. Il peut y en avoir une définie de manière très précise, mais on ne la connaît pas. Il est souvent fait l'hypothèse que le service est FIFO par flux.

Dans la suite de ce manuscrit, nous ferons souvent l'hypothèse de *blind multiplexing* avec FIFO par flux.

Remarque 2.8. *Comme souligné dans [Rizzo 2005], l'hypothèse FIFO par flux est souvent faite, même si elle est parfois implicite.*

2.3. Résultats classiques

Ci-dessous, deux théorèmes classiques du Network Calculus sur la mise en tandem (ou concaténation) de deux serveurs, et sur le service résiduel (appelé « Blind multiplexing » dans [Le Boudec 2001, théorème 6.2.1]), c'est-à-dire le service disponible pour l'un des flux lorsque deux flux agrégés traversent un serveur (on ne fait pas d'hypothèse sur la politique de service). Dans les deux cas, il est intéressant d'étudier ce que ces théorèmes peuvent nous apporter sur le calcul des bornes de performance.

Théorème 2.3 (Tandem [Le Boudec 2001, Chang 2000])**.** *Soit un flux traversant deux serveurs en tandem de courbes de service respectives β_1 et β_2. Alors la concaténation des deux serveurs offre un service simple minimal de courbe $\beta_1 * \beta_2$ à ce flux.*

Définition 2.11 (Pay Bursts Only Once)**.** *(PBOO, [Le Boudec 2001, p28]) Lorsqu'on étudie le délai pire-cas pour un flux traversant un système formé par la concaténation de deux serveurs, on peu utiliser au moins deux méthodes différentes :*

1. *utiliser directement la courbe de service du système (théorème 2.3) ;*
2. *calculer itérativement les bornes au niveau de chaque serveur (théorème 2.2).*

Avec la deuxième méthode, les rafales du flux apparaissent plusieurs fois dans l'expression du délai alors qu'elles n'apparaissent qu'une fois dans l'expression de la première méthode. Ce qui conduit potentiellement à des résultats plus pessimistes. C'est le phénomène PBOO.

Théorème 2.4 (Service résiduel [Le Boudec 2001, Chang 2000])**.** *Soit un noeud offrant une courbe de service strict β, et deux flux (agrégés) entrant dans ce serveur, avec des courbes d'arrivée respectives α_1 et α_2. Alors une courbe de service simple pour le flux 1 est $\beta_1 = (\beta - \alpha_2)_+$.*

Définition 2.12. *Pay Multiplexing Only Once ((PMOO), [Fidler 2003, Schmitt 2006a]) Par analogie avec PBOO, [Fidler 2003] étudie des systèmes où plusieurs fux agrégés traversent un serveur, et où il y a du trafic transverse. Lorsqu'on étudie un flux en particulier, on peut calculer une courbe de service résiduel pour ce flux, puis utiliser cette courbe de service pour calculer les bornes sur les performances. Procéder ainsi permet de ne « payer » qu'une seule fois pour les rafales du trafic transverse.*

Des algorithmes sont donnés dans [Schmitt 2006a] pour les deux types de méthodes vues à la définition 2.11, et on trouvera un exemple à la section 9.4 :

- *total flow analysis* (TFA) qui consiste à calculer — séparément — une borne supérieure sur le délai pour chaque serveur traversé par le flux observé puis faire la somme ;
- *separated flow analysis* (SFA) qui consiste à calculer une courbe de service résiduelle pour chaque serveur sur le chemin du flux observé, puis calculer la convolution de ces courbes de service pour obtenir une courbe de service résiduelle pour le chemin complet, et finalement calculer une borne de bout en bout sur le délai en utilisant le théorème 2.2.

2.4. Avantages et limites du NC

Le Network Calculus, fondé sur l'utilisation d'enveloppes pour modéliser les contraintes du réseau et l'utilisation du dioïde $(min, +)$ pour combiner ces contraintes, a intrinsèquement certains avantages et inconvénients par rapport à d'autres méthodes d'étude de performances d'un réseau. Les mêmes avantages et inconvénients se retrouveront donc aussi dans des modèles comme le Real-Time Calculus par exemple. Ces caractéristiques sont :

- un formalisme mathématique précis (l'utilisation de $(min, +)$) ; ce qui permet *a priori*, par rapport à des modèles basés sur l'interprétation au cas par cas des mouvements de données dans le réseau, d'éviter les erreurs d'interprétation lors de l'application à des exemples pratiques.
- la possibilité d'utiliser des types d'enveloppes plus ou moins précises pour les contraintes. Par exemple, pour accélérer les calculs, on peut décider d'utiliser des courbes d'arrivée / service de type concave / convexe : certes les bornes seront moins précises, mais on est assuré d'obtenir des bornes qui sont toujours valides.
- la possibilité, en théorie, d'étudier des réseaux à n'importe quelle échelle. En effet, les notions de trajectoire entrée / sortie et de système ne font pas d'hypothèse sur l'objet étudié, et la combinaison de plusieurs sous-systèmes pour former un systèmes est aussi tout à fait possible. Dans la pratique, il est cependant difficile d'obtenir des bornes précises.
- le pessimisme des bornes, qu'il est difficile de contrôler. C'est justement l'objet des recherches sur le Network Calculus...

On pourra trouver des comparaisons plus précises sur les avantages et inconvénients d'autres modèles dans, par exemple, [Boyer 2010b].

CHAPITRE 3

ÉTAT DE L'ART

3.1. Applications du Network Calculus

Le Network Calculus trouve des applications en *qualité de service*, dans la certification des réseaux — en particulier avioniques —, et son utilisation est à l'étude pour les réseaux embarqués.

Qualité de service

Par *qualité de Service* (QoS) dans un réseau, on entend à la fois la garantie d'une certaine performance dans le service des données, et l'ensemble des mécanismes qui permettent de rendre certaines applications prioritaires par rapport à d'autres. Fournir ces mécanismes ou garantir une certaine qualité relève de l'*ingénierie de trafic*.

On peut par exemple vouloir garantir des bornes sur :

- le débit
- les délais
- la charge au niveau des serveurs
- la jigue (*jitter*, ou plus précisément *packet delay variation*) : c'est la variation sur le délai de bout en bout entre des paquets envoyés à intervalle régulier.

L'*Internet Engineering Task Force* (IETF) a, entre autres, pour mission de définir des architectures réseau pour internet qui permettent de donner la priorité à certains flux et contrôler le trafic. Parmi ces modèles, on peut citer :

- Integrated Services (IntServ) qui réserve des ressources à l'avance pour garantir un service. Mais maintenir à jour l'ensemble des ressources réservées passe difficilement à l'échelle dans certains grands réseaux ([IntServ , Le Boudec 1998] et [Le Boudec 2001, p. 75]).
- Differentiated services (DiffServ) qui utilise un code présent dans les paquets IP, ce qui permet un meilleur passage à l'échelle.

Certification et systèmes embarqués

Switched ethernet : réseaux de type ethernet, qui s'organisent autour de commutateurs [Georges 2002] (comme les réseaux de type ATM).

AFDX (Avionics Full DupleX switched ethernet) : le NC a été utilisé pour garantir les délais dans le réseau AFDX de l'Airbus A380 [Grieu 2004, Frances 2009, Charara 2005]. La thèse de J. Grieu a de plus grandement amélioré certaines bornes obtenues en Network Calculus en prenant en compte des propriétés physiques du réseau comme le fait que les liaisons entre deux serveurs ont un débit maximal.

FlexRay : système de communication en développement dans l'industrie automobile basé sur TDMA [FlexRay 2005].

3.2. Domaines et modèles proches du NC

Certains domaines répondent aux mêmes problématiques que le Network Calculus (ou à des problématiques très proches) mais ont des approches différentes.

Le Network Calculus stochastique est une branche probabiliste du NC dont les applications sont très proches, mais qui ne vise pas à donner des bornes déterministes. C'est-à-dire que les contraintes sur les arrivées ou les services ne sont pas respectées, avec une certaine probabilité. Certaines des méthodes introduites dans [Diaz 2004] sont par exemple utilisées dans [Scharbarg 2009] pour étudier les réseaux de type AFDX. Des détails concernant l'aspect probabiliste peuvent être trouvés dans [Fidler 2006a] ou [Ciucu 2005]. On pourra aussi consulter une étude très détaillée de l'évolution du Stochastic Network Calculus dans [Fidler 2010].

Des formalismes proches du NC comme le Sensor Calculus [Schmitt 2005] (enveloppes proches des courbes de service simple), ou le Real-Time Calculus (enveloppes proches des courbes de service strict) étudient des réseaux spécifiques. Nous aborderons plus précisément RTC dans le chapitre 5.

L'approche par trajectoire de Steven Martin permet de calculer des délais de bout en bout pour des réseaux avec priorité fixe [Bauer 2010, Martin 2005]. La méthode consiste à construire directement la trajectoire (ici, l'ensemble des mouvements

de paquets) qui produit le pire cas pour la mesure de performance considérée (typiquement le délai).

La stabilité des réseaux FIFO est plus particulièrement étudiée dans le cadre du Permanent Sessions Model dans [Andrews 1996, Andrews 2000, Charny 2000, Andrews 2001, Mavronicolas 2001, Andrews 2007, Cholvi 2007]. Il y est montré qu'il existe des réseaux FIFO instables pour des charges de réseau arbitrairement petites.

Les systèmes à événements discrets (SED) , bien qu'utilisant une approche différente de celle du Network Calculus [Gaubert 1992, Baccelli 1992, Le Corronc 2011, Le Corronc 2012a], , ont des problématiques très proches. La théorie de la résiduation permet d'étudier des systèmes dans le dioïde $(min, +)$, et la plupart des opérateurs utilisés en NC trouvent leur origine dans les SED. Récemment, le Network Calculus a été utilisé pour étudier des systèmes de manière plus générale [Boudec 2012], avec des techniques très proches des SED.

3.3. Logiciels NC

Plusieurs logiciels ont été développés par différentes équipes dans le cadre du NC ou des domaines proches du NC. Ils s'agit d'outils spécifiques aux différents modèles (NC, RTC...), d'applications qui sont utilisées à l'échelle industrielle, ou encore de librairies de calcul spécialisées. La liste suivante n'est pas exhaustive, certains articles citant d'autres prototypes n'étant pas forcément accessibles.

CyNC (Cyclic Network Calculus) [CyNC , Schioler 2006]

- Auteurs : H. Schioler, J. Jensen, J. D. Nielsen, K. G. Larsen ;
- Soutiens : Center for Embedded SW Systems et Aalborg University, Denmark ;
- Modèles : RTC (malgré la dénomination NC) ; topologies = générales (même avec dépendances cycliques) ; politiques de service = comme RTC (Fixed Priority et FIFO) ; fonctions = temps discret, fonctions ultimement affines ;
- Implémentation : librairie Matlab, couplée avec la plateforme de simulation SimuLink qui permet de spécifier des réseaux de communication (aussi sous Matlab).

COINC (COmputational Issues in Network Calculus) [COINC , Bouillard 2009a]

- Auteurs : S. Lagrange, A. Bouillard, B. Gaujal, E. Thierry ;
- Soutiens : ARC INRIA, INRIA Rhone-Alpes, LIP et ISTIA Angers ;
- Modèles : librairie de calcul des opérations du NC pour les fonctions affines par morceaux ultimement pseudo-périodiques (temps continu ou discret) ;
- Implémentation : librairie C++ interfacée à SciLab.

DISCO Network Calculator [DISCO]

- Auteurs : J. B. Schmitt, F. A. Zdarsky, I. Martinovic, N. Gollan ;
- Soutien : Distributed Computer Systems Lab, University of Kaiserslautern, Germany ;
- Modèles : NC ; topologies = tandem et trafic transverse sur sous-chemins ; politiques de service = FIFO ou *blind multiplexing* ; fonctions = temps continu, fonctions affines par morceaux ultimement affines ;
- Implémentation : librairie Java couplée avec GraphML pour la topologie des réseaux de comm.

RTC-Toolbox [Thiele 2006, Wandeler 2006b]

- Auteurs : E. Wandeler, L. Thiele ;
- Soutien : Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland ;
- Modèles : RTC = Variability Characterization Curves (VCC), opérateurs min-plus et max-plus, fonctions affines par morceaux, continues à gauche ou à droite, ultimement pseudo-périodiques. Vue l'implémentation, on doit pouvoir étendre son application à de nombreux modèles ;
- Implémentation : Noyau Java (pour développer de nouvelles fonctionnalités), et ensemble de librairies Matlab (pour utiliser les outils RTC).

MinMaxGD [Cottenceau 2000], puis [Cottenceau 2001, Le Corronc 2012a], qui complète la librairie MaxPlus de Scilab [Gaubert]

- Auteurs : L. Hardouin, M. Lhommeau, B. Cottenceau, E. Le Corronc (pour l'extension ContainerMinMaxGD) ;
- Soutien : ISTIA, Université d'Angers ;
- Modèles : librairie de calcul des opérations (min,+) et (max,+) pour les fonctions de $\mathbb{N}$ dans $\mathbb{N}$ mises sous forme de séries formelles ;
- Implémentation : librairie C++ interfacée à Scilab.

RTaW-Pegase [Boyer 2010b]

- Auteurs : M. Boyer, N. Navet, J. Migge ;
- Soutiens : ANR Pegase, RTaW (RealTime-at-Work) ;
- Modèles : NC, topologies = acycliques, politiques de service = FIFO et Fixed Priority, fonctions = temps continu, fonctions affines par morceaux ultimement pseudo-périodiques ;
- Implémentation : logiciel en Java avec interface utilisateur, dont éditeur de réseau de communication.

DEBORAH (DElay BOund Rating AlgoritHm) [Bisti 2010]

- Auteurs : L. Bisti, L. Lenzini, E. Mingozzi, G. Stea ;
- Soutien : Computer Networking Group (CNG), University of Pisa, Italy ;
- Modèles : NC, topologies = tandem avec trafic transverse sur des sous-chemins, po-

litique de service FIFO, fonctions = temps continu, fonctions affines par morceaux ultimement affines ;
- Implémentation : logiciel en C++.

NC-maude [Boyer 2010a]
- Auteur : M. Boyer ;
- Soutien : ONERA, Toulouse ;
- Modèles : librarie de calcul par réécriture manipulant les opérations standard (min,+) et les règles de calcul qui peuvent être fournies par des théorèmes de Network Calculus ;
- Implémentation : basée sur le logiciel de réécriture MAUDE.

ConfGen [Charara 2005]
- Auteurs : outil non public, inspiré des travaux de thèse de J. Grieu ;
- Soutien : développé par Rockwell Collins ;
- Modèles : *a priori* basé sur les réseaux AFDX de l'A380 ;
- Implémentation : outil non public.

3.4. Contributions

L'objectif premier de ce travail de thèse était de développer de nouveaux algorithmes dans le cadre du Network Calculus. Le cadre général du Network Calculus a été présenté en partie I. Une liste des publications auxquelles a conduit ce travail est disponible page 45.

Nous nous sommes tout d'abord intéressé à l'aspect modélisation (partie II), et aux liens entre les différents modèles qui n'avaient pas été étudiés de manière systématique. Nous nous sommes demandé :

- quels sont les liens entre les différents modèles présents dans la littérature. Nous avons démontré l'équivalence d'expressivité entre Greedy Processor en Real-Time Calculus et Variable Capacity Node en Network Calculus (chapitre 5, rapport de recherche [6], et présentation à [7]) ;
- quelles classes de fonctions sont utilisables en pratique pour exprimer les contraintes du NC. Nous avons étendu les travaux de [Bouillard 2007c, Bouillard 2008c] sur les classes de fonctions stables par les opérateurs du Network Calculus (chapitre 4, en préparation [1]) ;
- quels sont les liens entre les différentes définitions de courbe de service. Nous avons établi une hiérarchie entre (certaines) définitions de courbe de service (chapitre 6, WODES'2010 [1], et présentation à [7]) ;

1. Ce travail constitue le prolongement de la « Toolbox » [Bouillard 2008c] et reprend la plupart des définitions de [Bouillard 2007c]

- s'il était possible de définir un nouveau type de service intermédiaire entre service simple et service strict, qui soit préservé par la mise en tandem de serveurs (chapitre 7, rapport de recherche [6]).

Nous avons aussi cherché à fournir des outils variés (partie III) pour étudier les bornes pire-cas en Network Calculus :

- un nouvel opérateur $(min, +)$ — la convolution multidimensionnelle —, qui généralise la convolution $(min, +)$ et permet d'étudier certaines configurations en temps polynomial lorsque les courbes d'arrivée et de service sont respectivement concaves et convexes (chapitre 8, VALUETOOLS'2008 [2]) ;
- un algorithme (de complexité exponentielle), utilisant l'optimisation linéaire, qui permet d'obtenir les meilleures bornes de performance possibles en multiplexage aveugle, pour tout réseau sans dépendances cycliques. Jusqu'alors aucun algorithme n'existait pour donner ces bornes dans le cas général. Nous fournissons aussi une heuristique qui donne des bornes moins précises mais a une complexité plus raisonnable (chapitre 9, INFOCOM'2010 [3]).

Tout au long de cette thèse, l'accent a été mis sur l'importance d'un formalisme clair (partie IV) ; c'est dans ce sens que nous proposons des notations pour les différents concepts de Network Calculus. Cette étude — ce ne sont que des propositions — nous a surtout permis de mettre à jour certaines difficultés de mise en œuvre du NC qui sont liées au formalisme (chapitre 11, MSR'09 [4]).

Nous souhaitons enfin faire quelques remarques sur l'utilisation des preuves en Network Calculus (chapitre 10), ainsi que sur les perspectives (chapitre 12).

Bibliographie personnelle

Conférences internationales avec comité de lecture

[1] Anne Bouillard, Laurent Jouhet, and Éric Thierry. Comparison of different classes of service curves in network calculus. Proceeding of the 10th International Workshop on Discrete Event Systems (WODES'2010), août 2010. 43

[2] Anne Bouillard, Laurent Jouhet, and Éric Thierry. Computation of a $(min, +)$ mutlidimensional convolution for end-to-end performance analysis. Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'2008), octobre 2008. 44

[3] Anne Bouillard, Laurent Jouhet, and Éric Thierry. Tight performance bounds in the worst-case analysis of feed forward networks. Proceedings of the 29th IEEE Conference on Computer Communications (INFOCOM'2010), mars 2010. 44

Conférence nationale avec comité de lecture

[4] Anne Bouillard, Marc Boyer, and Laurent Jouhet. Notations pour le calcul réseau. Actes du 7ème colloque francophone sur la Modélisation des Systèmes Réactifs (MSR'09), novembre 2009. 44

Rapports de recherche

[5] Anne Bouillard, Laurent Jouhet, and Éric Thierry. End-to-end performance guarantees for multipath flows. Rapport HAL hal-00289106, juin 2008.

[6] Anne Bouillard, Laurent Jouhet, and Éric Thierry. Service curves in network calculus : dos and don'ts. Rapport HAL inria-00431674, novembre 2009. 43, 44

Exposé invité

[7] Laurent Jouhet. Classification de modèles du calcul réseau. Journée WEED du 05/02/2010 à l'université d'Orsay Paris 11. 43

Deuxième partie :

Modèles et classes de fonctions

CHAPITRE 4

QUELLES CLASSES DE FONCTIONS UTILISER EN PRATIQUE

Dans ce chapitre, nous étudions différentes classes de fonctions, et leur stabilité pour les opérateurs du Network Calculus :

- dans la continuité des travaux de [Bouillard 2008c] et [Bouillard 2007c], nous exposons plusieurs classes de fonctions, et certaines de leurs propriétés algébriques qui peuvent être d'un intérêt pratique, tant pour simplifier des preuves que pour implémenter efficacement certaines opérations (Section 4.1) ;
- nous donnons deux cas particuliers de fonctions qu'il est intéressant d'étudier car elles fournissent des (contre-)exemples (Section 4.2) ;
- nous nous intéressons à la clôture de classes de fonctions affines par morceaux pour certains opérateurs : ce point est primordial pour être sûr que l'on saura exploiter les résultats d'un calcul dans d'autres calculs (Section 4.3) ;

- nous étudions la possibilité d'exprimer toute fonction affine par morceaux ultimement pseudo-périodique à l'aide de fonctions affines et des opérateurs du Network Calculus. Ce résultat — théorique — a pour but de préciser s'il y a égalité (ou pas) entre la classe des fonctions affines par morceaux ultimement pseudo-périodiques, et la clôture des fonctions affines par les opérateurs du NC (Section 4.4) ;
- nous étudions les fonctions suradditives, et essayons de déterminer s'il y a équivalence entre deux propriétés sur cette classe de fonctions (Section 4.5).

Les définitions et théorèmes des sections 4.1, 4.2, 4.3 proviennent de [Bouillard 2007c].

Ces résultats peuvent être vus comme un guide pour quiconque souhaiterait implémenter les opérations du Network Calculus.

Certains outils implémentent les opérateurs NC pour plusieurs classes de fonctions [Boyer 2010b, Boyer 2011], ce qui permet en fonction du type de fonctions, soit de calculer des bornes plus rapidement, soit d'assurer que les résultats seront en précision exacte.

4.1. Introduction

4.1.1. Classes de fonctions

Stabilité des classes de fonctions

Une classe C de fonctions est *stable* (ou *close*) par un ensemble E d'opérateurs si, lorsqu'on applique des opérateurs de E à une des fonctions de C, le résultat (lorsqu'il est défini) appartient encore à C.

La *clôture* d'une classe C de fonctions par un ensemble E d'opérateurs est la classe de toutes les fonctions obtenues en appliquant un nombre fini de fois des opérateurs de E à des éléments de C. C'est aussi la plus petite classe de fonctions qui contient C, et qui soit stable par E.

Comportements asymptotiques : classes de fonctions

Définition 4.1. *Soit f une fonction de* X *dans* $\overline{\mathbb{R}}$ *où* $X = \mathbb{N}$ *ou* $\mathbb{R}_+$. *On a alors, avec* $X^* = X \setminus \{0\}$, *f est :*

- *affine si*
  $\exists \sigma, \rho \in \mathbb{R}, \forall t \in X, f(t) = \rho t + \sigma$ *ou* $\forall t \in X, f(t) = +\infty$ *(resp.* $-\infty$*)* ;
- *ultimement affine si*
  $\exists T \in X, \exists \sigma, \rho \in \mathbb{R}, \forall t > T, f(t) = \rho t + \sigma$ *ou* $\forall t > T, f(t) = +\infty$ *(resp.* $-\infty$*)* ;
- *pseudo-périodique si*
  $\exists (c, d) \in \mathbb{R} \times X^*, \forall t \in X, f(t + d) = f(t) + c$;

- *ultimement pseudo-périodique si*
  $\exists T \in X, \exists (c,d) \in \mathbb{R} \times X^*, \forall t > T, f(t+d) = f(t) + c$;
- *ultimement franche si*
  $\exists T \in X, \forall t > T, f(t) \in \mathbb{R}$, *ou* $\forall t > T, f(t) = +\infty$, *ou* $\forall t > T, f(t) = -\infty$;
- *franche si*
  elle est ultimement franche comme ci-dessus, et $\forall\ 0 \leq t < T, f(t) \in \mathbb{R}$. *De plus, il faut* $f(T) \in \mathbb{R}$ *ou* $f(T) = +\infty$ *(resp.* $-\infty$*) dans le cas où* $\forall t > T, f(t) = +\infty$ *(resp.* $-\infty$*).*

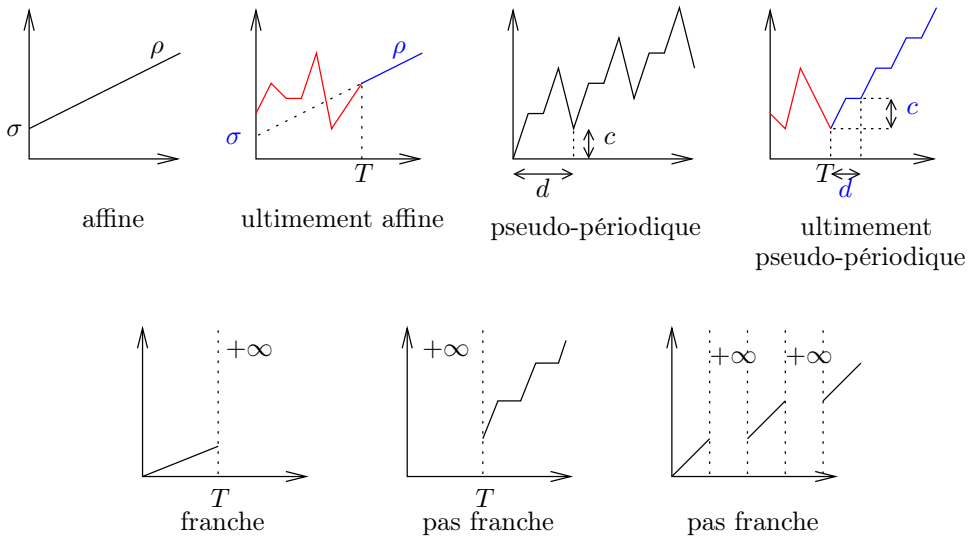


Figure 4.1 : Illustration graphique des classes de fonctions définies en 4.1

Pour les fonctions affines et ultimement affines, ρ est le *taux d'accroissement*.

Pour une fonction f pseudo-périodique, d est appelée une *période* de f, c est l'*incrément* associé à d, et *la période* de f est la plus petites de ses périodes (non nulles).

Pour une fonction ultimement affine (resp. ultimement pseudo-périodique), nous dirons parfois que la fonction est ultimement affine *à partir de* T, et appellerons T *un rang* de la fonction.

Étant donnée une fonction ultimement pseudo-périodique, il existe un plus petit rang de pseudo-périodicité, que nous appellerons *le rang* de la fonction.

Plus généralement, pour $f, g \in \mathcal{F}$, nous dirons qu'*ultimement* $f = g$ si

$$\exists T \in X, \forall t > T, f(t) = g(t)$$

Remarque 4.1. *Être franche (dans le cas réel) est équivalent à avoir un support égal à* $[0, T]$ *ou* $[0, T[$, *avec* $T \in \mathbb{R} \cup \{+\infty\}$.

Une fonction croissante définie sur $X = \mathbb{N}$ ou $X = \mathbb{R}_+$ est toujours ultimement franche et, si $f(0) \in \mathbb{R}$, elle est franche.

Remarque 4.2. *Une fonction ultimement affine est ultimement franche et pseudo-périodique ; elle admet tout $\varepsilon > 0$ comme période.*

Fonctions affines par morceaux

Définition 4.2. *Une fonction $f \in \mathcal{F}$ est* affine par morceaux *s'il existe une suite croissante $(a_i)_{i\in\mathbb{N}}$ qui tend vers $+\infty$, avec $a_0 = 0$ et $\forall i \geq 0$, f est affine sur $]a_i, a_{i+1}[$, i.e. $\forall t \in]a_i, a_{i+1}[$, $f(t) = +\infty$ ou $\forall t \in]a_i, a_{i+1}[$, $f(t) = -\infty$ ou $\exists \sigma_i, \rho_i \in \mathbb{R}$, $\forall t \in]a_i, a_{i+1}[$, $f(t) = \sigma_i + \rho_i t$. Si cette suite est minimale au sens de l'inclusion, les (a_i) sont appelés* discontinuités.

Soit $f \in \mathcal{F}$ une fonction affine par morceaux et $a \in \mathbb{R}_+$, la limite à droite *de f en a est définie par $f(a+) = \lim_{t\to a, t>a} f(t)$ et la* limite à gauche *de f en a est définie par $f(a-) = \lim_{t\to a, t<a} f(t)$. Ces limites existent.*

Soit $X \subseteq \mathbb{R}_+$ et $Y \subseteq \mathbb{R}$, nous appelons $\mathcal{F}[X, Y]$ l'ensemble des fonctions affines par morceaux de $\mathcal{F}$ telles qu'il existe une suite $(a_i)_{i\in\mathbb{N}}$ avec les propriétés ci-dessus et qui vérifient $\forall i \geq 0$, $a_i \in X$ et $f(a_i), f(a_i+), f(a_i-) \in Y \cup \{-\infty, +\infty\}$.

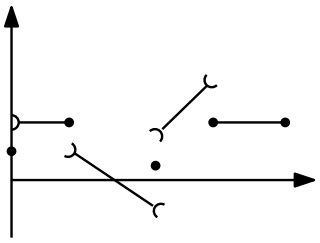


Figure 4.2 : Une fonction affine par morceaux correspondant à la définition 4.2.

De telles fonctions sont continues à gauche (resp. à droite) si $\forall i \geq 0$, $f(a_i) = f(a_i-)$ (resp. $f(a_i) = f(a_i+)$).

Nous étudierons principalement $\mathcal{F}[\mathbb{N}, \mathbb{R}]$, $\mathcal{F}[\mathbb{Q}_+, \mathbb{R}]$, $\mathcal{F}[\mathbb{R}_+, \mathbb{R}]$ ou $\mathcal{F}[\mathbb{Q}_+, \mathbb{Q}]$.

Une fonction affine par morceaux jusqu'en $T + d$, puis (ultimement) pseudo-périodique de période d à partir de T est affine par morceaux. En effet, la partie définie sur $[T; T + d]$, qui est affine par morceaux, sera « recopiée » (avec un incrément).

4.1.2. Formules algébriques

Distributivité

La proposition suivante n'est pas nouvelle, mais nous soulignons le fait qu'elle s'applique à des familles infinies de fonctions, ce que nous utiliserons plus loin.

Proposition 4.1 (Distributivité)**.** *Soit $(f_i)_{i\in I}$ et $(g_j)_{j\in J}$ deux familles de fonctions, toutes les deux dans $\mathcal{D}$ (ou toutes les deux dans $\mathcal{F}$) où I et J sont deux ensembles quelconques (potentiellement*

infinis). Alors, lorsque les résultats sont définis, on a

$$(\sup_{i\in I} f_i) \oslash (\inf_{j\in J} g_j) = \sup_{i\in I, j\in J} (f_i \oslash g_j) \tag{4.1}$$

$$(\inf_{i\in I} f_i) * (\inf_{j\in J} g_j) = \inf_{i\in I, j\in J} (f_i * g_j) \tag{4.2}$$

Multiplication par une constante

La proposition suivante concerne la multiplication par une constante positive, et est une simple conséquence de la définition des opérateurs. Nous n'appelons pas cette propriété « linéarité » car dans la suite nous utiliserons parfois la linéarité dans $(+,\times)$, parfois dans $(\min,+)$.

Attention : multiplier par une constante négative échange min et max (de même que inf et sup) dans la proposition suivante.

Par abus de langage, nous notons $\alpha f(t)$ la fonction $t \mapsto \alpha f(t)$.

Proposition 4.2 (Multiplication par une constante positive)**.** *Multiplier une fonction de $\mathcal{F}$ par une constante* positive *donnée commute avec les opérateurs du NC. Plus précisément, soit $f, g \in \mathcal{D}$ ou $\mathcal{F}$, et $\alpha \in \mathbb{R}_+$. Alors :*

- $\alpha f(t) + \alpha g(t) = \alpha(f+g)(t)$;
- $\alpha f(t) - \alpha g(t) = \alpha(f-g)(t)$;
- $\min(\alpha f(t), \alpha g(t)) = \alpha \min(f,g)(t)$;
- $\max(\alpha f(t), \alpha g(t)) = \alpha \max(f,g)(t)$;
- $(\alpha f(t)) * (\alpha g(t)) = \alpha(f*g)(t)$;
- $(\alpha f(t)) \oslash (\alpha g(t)) = \alpha(f \oslash g)(t)$;
- $(\alpha f(t))^* = \alpha f^*(t)$.

Relèvement

Les propositions suivantes montrent que transformer les fonctions de $\mathcal{F}$ en ajoutant un terme linéaire $t \mapsto \lambda t$ ou un terme constant $t \mapsto \mu$ ne complique généralement pas le calcul.

Par abus de langage, nous notons $f(t) + \lambda t + \mu$ la fonction $t \mapsto f(t) + \lambda t + \mu$.

Proposition 4.3 (Relèvement)**.** *Soit $f, g \in \mathcal{D}$ ou $\mathcal{F}$, et $\lambda, \mu \in \mathbb{R}$. Alors :*

- $(f(t) + \lambda t + \mu) + (g(t) + \lambda t + \mu) = (f+g)(t) + 2\lambda t + 2\mu$;
- $(f(t) + \lambda t + \mu) - (g(t) + \lambda t + \mu) = (f-g)(t)$;
- $\min((f(t) + \lambda t + \mu), (g(t) + \lambda t + \mu)) = \min(f,g)(t) + \lambda t + \mu$;
- $\max((f(t) + \lambda t + \mu), (g(t) + \lambda t + \mu)) = \max(f,g)(t) + \lambda t + \mu$;
- $(f(t) + \lambda t + \mu) * (g(t) + \lambda t + \mu) = (f*g)(t) + \lambda t + 2\mu$;
- $(f(t) + \lambda t + \mu) \oslash (g(t) + \lambda t + \mu) = (f \oslash g)(t) + \lambda t$;
- $(f(t) + \lambda t)^* = f^*(t) + \lambda t$ *(ajouter μ peut être beaucoup plus difficile).*

Ce sont ici aussi des conséquences directes des définitions des opérateurs. La transformation $f(t) \mapsto f(t) + \lambda t$ peut être vue comme un « relèvement », en particulier on peut transformer tout fonction $f \in \mathcal{D}$ ou $\mathcal{F}$ telle que $\exists p \in \mathbb{R}_+, \forall t < t', f(t') - f(t) \geq -p \cdot (t' - t)$ en une fonction croissante en prenant $\lambda = p$.

Cependant, cette proposition nous montre aussi qu'il est peu probable de pouvoir tirer parti de la monotonie des fonctions pour construire des algorithmes plus rapides que dans le cas général.

4.2. Deux exemples utiles : fonction indicatrice et distance

Soit $S \subseteq \mathbb{R}_+$ (ou $\mathbb{N}$), nous notons $\mathbb{1}_S$ la fonction indicatrice de S, définie sur $\mathbb{R}_+$ (ou $\mathbb{N}$) par :

$$\mathbb{1}_S(t) = \begin{cases} 1 & \text{si } t \in S \\ 0 & \text{sinon} \end{cases}$$

Avec cette définition, remarquons que $\mathbb{1}_S \in \mathcal{F}[\mathbb{R}_+, \mathbb{R}]$ si et seulement si la frontière de S (adhérence de S moins intérieur de S) n'admet pas de point d'accumulation (c'est-à-dire un réel obtenu par limite d'une suite extraite de S).

Nous notons $dist(t, S)$ la distance de t à S, c'est-à-dire

$$dist(t, S) = \inf\{|t - s|,\ s \in S\}.$$

Et nous utiliserons aussi le PGCD(a, b) de deux nombres, noté $a \wedge b$:

$$a \wedge b = \text{PGCD}(a, b).$$

Soit $\alpha, \beta \in \mathbb{R}_+^*$, nous utiliserons les notations $\mathbb{N}\alpha = \{s \in \mathbb{R} \mid \exists n \in \mathbb{N},\ s = n\alpha\}$, $\mathbb{Z}\alpha = \{s \in \mathbb{R} \mid \exists n \in \mathbb{Z},\ s = n\alpha\}$ et $\mathbb{N}\alpha - \mathbb{N}\beta = \{s \in \mathbb{R} \mid \exists p, q \in \mathbb{N},\ s = p\alpha - q\beta\}$. De même $\mathbb{N}\alpha + \mathbb{N}\beta = \{s \in \mathbb{R} \mid \exists p, q \in \mathbb{N},\ s = p\alpha + q\beta\}$. Remarquons que, si $\alpha/\beta \in \mathbb{Q}$ alors $\mathbb{N}\alpha - \mathbb{N}\beta = \mathbb{Z}(\alpha \wedge \beta)$, et si $\alpha/\beta \notin \mathbb{Q}$ alors $\mathbb{N}\alpha - \mathbb{N}\beta$ est dense dans $\mathbb{R}$.

Proposition 4.4 (Convolution). *Soit $\alpha, \beta \in \mathbb{R}_+^*$ (resp. $\mathbb{N}^*$) et $f, g \in \mathcal{F}$ (resp. $\mathcal{D}$), alors :*

(i) Soit $f = \mathbb{1}_{\overline{\mathbb{N}\alpha}}$ et $g = \mathbb{1}_{\overline{\mathbb{N}\beta}}$, alors $f * g = \mathbb{1}_{\overline{\mathbb{N}\alpha + \mathbb{N}\beta}}$.

(ii) Soit $f(t) = dist(t, \mathbb{N}\alpha)$ et $g(t) = dist(t, \mathbb{N}\beta)$, alors $(f * g)(t) = dist(t, \mathbb{N}\alpha + \mathbb{N}\beta)$.

(iii) Dans les deux cas précédents, si $f, g \in \mathcal{F}$ et $\beta/\alpha \notin \mathbb{Q}$, alors $f * g$ n'est pas ultimement pseudo-périodique.

Preuve.

(i) Conséquence de la définition de $*$.

(ii) Soit $t_1, t_2 \in \mathbb{R}_+$ tel que $t_1 + t_2 = t$, alors $\exists s_1 \in \mathbb{N}\alpha$ (resp. $s_2 \in \mathbb{N}\beta$) tel que $f(t_1) = dist(t_1, \mathbb{N}\alpha) = |t_1 - s_1|$ (resp. $g(t_2) = dist(t_2, \mathbb{N}\beta) = |t_2 - s_2|$). On a $f(t_1) + g(t_2) = |t_1 - s_1| + |t_2 - s_2| \geq |t - (s_1 + s_2)| \geq dist(t, \mathbb{N}\alpha + \mathbb{N}\beta)$, et donc

$$(f * g)(t) \geq dist(t, \mathbb{N}\alpha + \mathbb{N}\beta).$$

D'autre part, soit $t \geq 0$, on a $\exists s_1 \in \mathbb{N}\alpha$, $s_2 \in \mathbb{N}\beta$ tel que $dist(t, \mathbb{N}\alpha + \mathbb{N}\beta) = |t - (s_1 + s_2)|$.

- Si $s_1 = s_2 = 0$, alors $(f * g)(t) \leq dist(t, \mathbb{N}\alpha + \mathbb{N}\beta)$ car en effet $(f * g)(t) \leq |t|$ pour tout t.
- Sinon $s_1 > 0$ ou $s_2 > 0$. Supposons que $s_2 > 0$. Choisissons $t_1 = s_1$ et $t_2 = t - s_1$, on a alors $t_1 + t_2 = t$ et $t_2 \geq 0$ car si on avait $t_2 < 0$, on aurait aussi $|t - s_1| = |t_2| < |t_2 - s_2| = dist(t, \mathbb{N}\alpha + \mathbb{N}\beta)$, ce qui est impossible. On a alors $f(t_1) = 0$ et $g(t_2) = dist(t_2, \mathbb{N}\beta) = dist(t - s_1, \mathbb{N}\beta) = dist(t, s1 + \mathbb{N}\beta) = dist(t, \mathbb{N}\alpha + \mathbb{N}\beta)$ par définition de s_1 qui réalise la distance minimale. Et donc $(f * g)(t) \leq f(t_1) + g(t_2) = dist(t, \mathbb{N}\alpha + \mathbb{N}\beta)$.

Ce qui termine la preuve de $(f * g)(t) = dist(t, \mathbb{N}\alpha + \mathbb{N}\beta)$.

(iii) La non-ultime pseudo-périodicité dans le cas où $\beta/\alpha \notin \mathbb{Q}$ est due au fait que dans ce cas $\mathbb{N}\beta - \mathbb{N}\alpha$ est dense dans $\mathbb{R}$. Plus précisément (dans les deux cas) : Raisonnons par l'absurde et supposons que $f*g$ soit ultimement pseudo-périodique de pseudo-période T à partir d'un certain rang r. Sur l'intervalle $[r; r + \mathrm{T}]$, le nombre de zéros de $f * g$ est fini (borné grossièrement par $\left\lceil \frac{r+\mathrm{T}}{\alpha} \right\rceil \times \left\lceil \frac{r+\mathrm{T}}{\beta} \right\rceil$). Il existe donc une constante $\varepsilon > 0$ telle que, sur $[r; r + \mathrm{T}]$, la distance entre deux zéros est bornée inférieurement strictement par ε. Et de même sur tout intervalle $[t_0; t_0 + \mathrm{T}], t_0 > r$ (avec le même ε). Comme $\mathbb{N}\beta - \mathbb{N}\alpha$ est dense dans $\mathbb{R}$, $\exists n, m \in \mathbb{N}, m \times \beta - n \times \alpha \leq \varepsilon$. Prenons $t_1 = n_1 \times \alpha + m_1 \times \beta$ un zéro de $f * g$ avec $n_1 \in \mathbb{N}^*, m_1 \in \mathbb{N}$ (on peut choisir $n_1 \neq 0$ sinon on aurait $\mathbb{N}\alpha + \mathbb{N}\beta = \mathbb{N}\beta$, or $\alpha \neq 0$). On pose alors $t_0 = n \times t_1$. On a alors $t_1 = (n \times n_1) \times \alpha + (n \times m_1) \times \beta$ et $t_1 + \varepsilon = (n \times (n_1 - 1)) \times \alpha + (n \times m_1 + m) \times \beta$ deux zéros de $f * g$ dont la distance n'est pas bornée inférieurement strictement par ε. D'où une contradiction. Par l'absurde, $f * g$ n'est pas pseudo périodique.

■

Proposition 4.5 (Déconvolution)**.** *Soit $\alpha, \beta \in \mathbb{R}_+^*$ (resp. $\alpha \in 2\mathbb{N}$, $\beta \in \mathbb{N}$) et $f, g \in \mathcal{F}$ (resp. $\mathcal{D}$), alors :*

(i) *Soit $f = \mathbb{1}_{\mathbb{N}\alpha}$ et $g = 1 - \mathbb{1}_{\mathbb{N}\beta} = \mathbb{1}_{\overline{\mathbb{N}\beta}}$, alors $f \oslash g = \mathbb{1}_{(\mathbb{N}\alpha - \mathbb{N}\beta) \cap \mathbb{R}_+}$. Si $\beta/\alpha \in \mathbb{Q}$, alors $f \oslash g = \mathbb{1}_{\mathbb{Z}(\alpha \wedge \beta) \cap \mathbb{R}_+}$. Si $\beta/\alpha \notin \mathbb{Q}$, alors $f \oslash g \notin \mathcal{F}[\mathbb{R}_+, \mathbb{R}]$.*

(ii) *Soit $f(t) = dist(t, \mathbb{N}\alpha)$ et $g(t) = dist(t, \mathbb{N}\beta)$. Si $\beta/\alpha \in \mathbb{Q}$, alors $(f \oslash g)(t) = \frac{\alpha}{2} - dist(t, \mathbb{Z}(\alpha \wedge \beta) + \frac{\alpha}{2})$. Si $\beta/\alpha \notin \mathbb{Q}$, alors $(f \oslash g)(t) = \frac{\alpha}{2}$.*

(iii) *Soit $f(t) = t.dist(t, \mathbb{N}\alpha)$ et $g(t) = t.dist(t, \mathbb{N}\beta)$. Si $\beta/\alpha \in \mathbb{N}$, alors $(f \oslash g)(t) = t\frac{\alpha}{2}$ si $t \in \mathbb{N}\alpha$ et $= +\infty$ si $t \notin \mathbb{N}\alpha$. Si $\beta/\alpha \notin \mathbb{N}$, alors $f \oslash g = +\infty$ sur $\mathbb{R}_+$.*

Preuve.

(*i*) D'après les définitions, on a $\forall t \in \mathbb{R}_+$ (resp. $\mathbb{N}$), $(f \oslash g)(t) \in \{-1, 0, +1\}$. En fait $(f \oslash g)(t) = \sup_{s\geq 0}(f(t+s)-g(s)) \geq 0$ en choisissant $s \in \mathbb{N}\beta$. Donc $(f \oslash g)(t) = \sup_{s\geq 0}(f(t+s)-g(s)) = 1$ si et seulement si $\exists s \geq 0$, $f(t+s) = 1$ et $g(s) = 0$, i.e. $t+s \in \mathbb{N}\alpha$ et $s \in \mathbb{N}\beta$. Ce qui peut se produire si et seulement si $t \in \mathbb{N}\alpha - \mathbb{N}\beta$. Donc $f \oslash g = \mathbb{1}_{(\mathbb{N}\alpha-\mathbb{N}\beta)\cap\mathbb{R}_+}$. La fin de la proposition est la conséquence de la forme de $\mathbb{N}\alpha - \mathbb{N}\beta$, selon que $\alpha/\beta \in \mathbb{Q}$ ou pas. Cependant, dans les deux cas, $f \oslash g$ a bien la même période α que f.

(*ii*) Soit $t \in \mathbb{R}_+$ (resp. $\mathbb{N}$). La déconvolution en t est $(f \oslash g)(t) = \sup_{s\geq 0}(dist(t+s, \mathbb{N}\alpha) - dist(s, \mathbb{N}\beta)) = \sup_{s\geq 0}(dist(s, \mathbb{N}\alpha - t) - dist(s, \mathbb{N}\beta))$. On a $(f \oslash g)(t) \leq \frac{\alpha}{2}$ et en prenant $s \in \mathbb{N}\beta$, on a aussi $(f \oslash g)(t) \geq 0$. Montrons d'abord que le supremum dans $(f \oslash g)(t)$ peut être choisi pour $s \in \mathbb{N}\beta$ au lieu de $s \geq 0$ sans changer sa valeur. Soit $s \geq 0$, prenons $a \in \mathbb{N}\alpha - t$ (resp. $b \in \mathbb{N}\beta$) tel que $|s-a| = dist(s, \mathbb{N}\alpha - t)$ (resp. $|s-b| = dist(s, \mathbb{N}\beta)$). Ces trois nombre s, a, b peuvent être ordonnés de plusieurs manières :

- Supposons que $s \leq a < b$ ou $b < a \leq s$, on a $dist(s, \mathbb{N}\alpha - t) - dist(s, \mathbb{N}\beta) = dist(s, a) - dist(s, b) < 0$ alors que $(f \oslash g)(t) \geq 0$, donc ce s n'est pas nécessaire au calcul du supremum.
- Supposons que $s \leq b \leq a$ ou $a \leq b \leq s$, alors faire varier s vers b ne change pas la valeur $dist(s, a) - dist(s, b)$, et a et b restent ses points les plus proches, respectivement dans $\mathbb{N}\alpha - t$ et $\mathbb{N}\beta$. Autrement dit, en choisissant $s' = b \in \mathbb{N}\beta$, on a $dist(s, \mathbb{N}\alpha - t) - dist(s, \mathbb{N}\beta) = dist(s, a) - dist(s, b) = dist(s', a) - dist(s', b) = dist(s', \mathbb{N}\alpha - t) - dist(s', \mathbb{N}\beta)$.
- Supposons que $a \leq s \leq b$. Par définition de a, on a $a \leq s \leq a + \frac{\alpha}{2} \leq b$. Soit on a $a \leq s \leq a + \frac{\alpha}{2} \leq a + \alpha < b$ et alors s n'est pas nécessaire pour le calcul du supremum car $dist(s, a) - dist(s, b) = dist(s, \mathbb{N}\alpha - t) - dist(s, \mathbb{N}\beta) \leq 0$, soit $a \leq s \leq a + \frac{\alpha}{2} \leq b \leq a + \alpha$. Dans ce dernier cas, en faisant varier s vers $a + \frac{\alpha}{2}$, $dist(s, a) - dist(s, b)$ augmente, et a et b restent les points les plus proches de s dans $\mathbb{N}\alpha - t$ et $\mathbb{N}\beta$. Ainsi avec $s' = a + \frac{\alpha}{2}$, on a $dist(s, \mathbb{N}\alpha - t) - dist(s, \mathbb{N}\beta) = dist(s, a) - dist(s, b) \leq dist(s', a) - dist(s', b) = dist(s', \mathbb{N}\alpha - t) - dist(s', \mathbb{N}\beta)$. Alors en faisant varier s' vers b, les points les plus proches respectivement dans $\mathbb{N}\alpha - t$ et $\mathbb{N}\beta$ deviennent $a + \alpha$ et b, et la valeur $dist(s', \mathbb{N}\alpha - t) - dist(s', \mathbb{N}\beta) = dist(s', a + \alpha) - dist(s', b)$ ne change pas. Ainsi pour $s'' = b \in \mathbb{N}\beta$, on a $dist(s'', \mathbb{N}\alpha - t) - dist(s'', \mathbb{N}\beta) = dist(s', \mathbb{N}\alpha - t) - dist(s', \mathbb{N}\beta) \geq dist(s, \mathbb{N}\alpha - t) - dist(s, \mathbb{N}\beta)$.
- Supposons que $b \leq s \leq a$. C'est le symétrique du cas précédent mais il faut faire attention à un détail. Comme ci-dessus, on peut se restreindre au cas $a - \alpha \leq b \leq a - \frac{\alpha}{2} \leq s \leq a$. Faire varier s vers $s' = a - \frac{\alpha}{2}$ augmente $dist(s, \mathbb{N}\alpha - t) - dist(s, \mathbb{N}\beta)$, et il suffit de vérifier que $s' \geq 0$: ce qui est vrai car $b \in \mathbb{N}\beta$ et donc $0 \leq b \leq a - \frac{\alpha}{2}$. Alors en faisant varier $s' = a - \frac{\alpha}{2}$ vers $s'' = b$, en raisonnant comme dans le cas précédent, il faut vérifier que le nouveau point potentiel le plus proche $a - \alpha$ appartient bien à $\mathbb{N}\alpha - t$: ce qui est vrai car $a \in \mathbb{N}\alpha - t$, $t \geq 0$ et $0 \leq s \leq a$ implique que $a \in \mathbb{N}^* - t$.

Nous avons montré que

$$(f \oslash g)(t) = \sup_{s\geq 0}(dist(s, \mathbb{N}\alpha - t) - dist(s, \mathbb{N}\beta)) = \sup_{s\in\mathbb{N}\beta}(dist(s, \mathbb{N}\alpha - t) - dist(s, \mathbb{N}\beta))$$

Et on a donc :

$$\begin{aligned}(f \oslash g)(t) &= \sup_{s\in\mathbb{N}\beta} dist(s, \mathbb{N}\alpha - t)\\ &= \frac{\alpha}{2} - inf_{s\in\mathbb{N}\beta} dist(s, \mathbb{N}\alpha + \frac{\alpha}{2} - t)\\ &= \frac{\alpha}{2} - dist(\mathbb{N}\beta, \mathbb{N}\alpha + \frac{\alpha}{2} - t)\\ &= \frac{\alpha}{2} - dist(\mathbb{N}\alpha - \mathbb{N}\beta, t - \frac{\alpha}{2})\end{aligned}$$

Si $\beta/\alpha \in \mathbb{Q}$, on a $\mathbb{N}\alpha - \mathbb{N}\beta = \mathbb{Z}(\alpha \wedge \beta)$ et $(f \oslash g)(t) = \frac{\alpha}{2} - dist(t, \mathbb{Z}(\alpha \wedge \beta) + \frac{\alpha}{2})$. Si $\beta/\alpha \notin \mathbb{Q}$, l'ensemble $\mathbb{N}\alpha - \mathbb{N}\beta$ est dense dans $\mathbb{R}$, et il existe donc une suite $a_n\alpha - b_n\beta$, $a_n, b_n \in \mathbb{N}$, qui tend vers $t - \frac{\alpha}{2}$. Ce qui signifie $dist(\mathbb{N}\alpha - \mathbb{N}\beta, t - \frac{\alpha}{2}) = 0$ et $(f \oslash g)(t) = \frac{\alpha}{2}$.

(iii) Soit $t \in \mathbb{R}_+$, on a

$$\begin{aligned}(f \oslash g)(t) &= \sup_{s\geq 0}((t+s).dist(t+s, \mathbb{N}\alpha) - s.dist(s, \mathbb{N}\beta))\\ &= t.\sup_{s\geq 0} dist(t+s, \mathbb{N}\alpha) + \sup_{s\geq 0}(s.dist(t+s, \mathbb{N}\alpha) - s.dist(s, \mathbb{N}\beta))\\ &= t\frac{\alpha}{2} + \sup_{s\geq 0}(s.dist(t+s, \mathbb{N}\alpha) - s.dist(s, \mathbb{N}\beta))\end{aligned}$$

L'analyse du second terme nous amène à étudier plusieurs cas :

- Supposons que $\beta/\alpha \notin \mathbb{Q}$. Le second terme est plus grand que $\sup_{s\geq\mathbb{N}\beta}(s.dist(t+s, \mathbb{N}\alpha) - s.dist(s, \mathbb{N}\beta)) = \sup_{s\in\mathbb{N}\beta} s.dist(s, \mathbb{N}\alpha - t)$. Comme $\mathbb{N}\beta - \mathbb{N}\alpha$ est dense dans $\mathbb{R}$, on sait qu'il existe une suite $b_n\beta - a_n\alpha$, $a_n, b_n \in \mathbb{N}$, qui tend vers $\frac{\alpha}{2} - t$ sans atteindre cette valeur. Ce qui implique que l'ensemble $\{b_n,\ n \in \mathbb{N}\}$ est infini et on peut extraire une sous-suite $b_{\phi(n)}$ croissante. On a $\sup_{s\in\mathbb{N}\beta} s.dist(s, \mathbb{N}\alpha - t) \geq \sup_{n\in\mathbb{N}} b_{\phi(b)}\beta.dist(b_{\phi(b)}\beta, \mathbb{N}\alpha - t) = +\infty$ car $b_{\phi(n)} \to +\infty$ et $dist(b_{\phi(b)}\beta, \mathbb{N}\alpha - t) = dist(b_{\phi(b)}\beta - a_{\phi(n)}\alpha, \mathbb{N}\alpha - t) \to dist(\frac{\alpha}{2} - t, \mathbb{N}\alpha - t) = \frac{\alpha}{2}$. Donc $(f \oslash g)(t) = +\infty$.
- Si $\beta/\alpha \in \mathbb{Q}$, on sait d'après le cas (ii) que $h(t) = \sup_{s\geq 0}(dist(t+s, \mathbb{N}\alpha) - dist(s, \mathbb{N}\beta)) = \frac{\alpha}{2} - dist(t, \mathbb{Z}(\alpha \wedge \beta) + \frac{\alpha}{2})$. Ce supremum $h(t)$ est égal à 0 si et seulement si $\frac{\alpha\wedge\beta}{2} = \frac{\alpha}{2}$ et $t \in \mathbb{Z}(\alpha\wedge\beta)$, i.e. $\beta/\alpha \in \mathbb{N}$ et $t \in \mathbb{N}\alpha$ (car $t \geq 0$). Dans ce cas, on a $\sup_{s\geq 0}(s.dist(t+s, \mathbb{N}\alpha) - s.dist(s, \mathbb{N}\beta)) = \sup_{s\geq 0} s.(dist(s, \mathbb{N}\alpha) - dist(s, \mathbb{N}\beta)) = 0$ et enfin $(f \oslash g)(t) = t\frac{\alpha}{2}$. Sinon $h(t)$ est strictement positive, et il existe un $s_0 \geq 0$ tel que $dist(t+s_0, \mathbb{N}\alpha) - dist(s_0, \mathbb{N}\beta) = c > 0$. Dans ce dernier cas, prenons la suite $s_n = s_0 + n(\alpha \vee \beta)$, $n \in \mathbb{N}$. On a $dist(t+s_n, \mathbb{N}\alpha) - dist(s_n, \mathbb{N}\beta) = c$ pour tout $n \in \mathbb{N}$. Ainsi $\sup_{n\in\mathbb{N}}(s_n.dist(t+s_n, \mathbb{N}\alpha) - s_n.dist(s_n, \mathbb{N}\beta)) = +\infty$ et enfin $(f \oslash g)(t) = +\infty$.

■

Remarque 4.3. *Notons que l'exemple $g(t) = t.dist(t, 2\mathbb{N})$ qui donne $(g \oslash g)(t) = t$ si t est pair et $= +\infty$ partout ailleurs, était déjà étudié dans la proposition 2 de [Bouillard 2007b]. La fonction g y était décrite comme l'interpolation linéaire de la fonction $f \in \mathrm{D}$ définie par $f(t) = t$ si t est impair et $= 0$ sinon.*

Proposition 4.6. *Soit $\mathrm{S} \subseteq \mathbb{R}_+$ (resp. $\mathbb{N}$) et $f \in \mathcal{F}$ (resp. $\mathcal{D}$). Alors :*

(i) Si $f(t) = \mathbb{1}_{\overline{\mathrm{S}}}(t)$, alors $f^*(t) = \mathbb{1}_{\overline{\mathbb{N}\mathrm{S}}}(t)$.

(ii) Si $f(t) = dist(t, \mathrm{S})$, alors $f^*(t) = dist(t, \mathbb{N}^*\mathrm{S})$ pour tout $t > 0$ (et $f^*(0) = 0$).

Preuve. Dans les deux cas, nous utilisons l'équation 2.1 qui définit la clôture sous-additive où, sans perte de généralité, on admet que $t_i \geq 0$ (au lieu de $t_i > 0$).

Pour la proposition (i), on a $f^*(t) = 0$ si et seulement si $t \in \mathbb{N}\mathrm{S}$; sinon $1 \leq f^*(t) \geq f(t) = 1$.

Pour la proposition (ii), soit $\varepsilon \in \mathbb{R}_+^*$, soit $t_1, \ldots, t_k > 0$ tel que $t = t_1 + \cdots + t_k$; alors $\forall 1 \leq i \leq k$, $\exists s_i \in \mathrm{S}$ tel que $|t_i - s_i| - \frac{\varepsilon}{k} \leq f(t_i) = dist(t_i, \mathrm{S}) \leq |t_i - s_i|$ (si S n'a pas de point d'accumulation en dehors de S, on a en fait l'égalité $dist(t_i, \mathrm{S}) = |t_i - s_i|$). Alors $f(t_1) + \cdots + f(t_k) \geq \sum_{1 \leq i \leq k} |t_i - s_i| - \frac{\varepsilon}{k} \geq |t - (s_1 + \cdots + s_k)| - \varepsilon \geq dist(t, \mathbb{N}\mathrm{S}) - \varepsilon$. C'est vrai pour tout $\varepsilon > 0$, ce qui implique $f^*(t) \geq dist(t, \mathbb{N}\mathrm{S})$. Soit alors $\varepsilon > 0$, $t > 0$ et prenons le plus petit entier $k \geq 1$ tel que $\exists s_1, \ldots, s_k \in \mathrm{S}$, $dist(t, \mathbb{N}\mathrm{S}) \leq |t - (s_1 + \cdots + s_k)| \leq dist(t, \mathbb{N}\mathrm{S}) + \varepsilon$ (si l'infimum est atteint pour un élément de $\mathbb{N}\mathrm{S}$, posons $\varepsilon = 0$ et manipulons une égalité ; c'est le cas par exemple dans le modèle discret si S est fini). Prenons $t_1 = s_1$, ..., $t_{k-1} = s_{k-1}$, $t_k = t - (s_1 + \cdots + s_{k-1})$; alors $t_1 + \cdots + t_{k-1} + t_k = t$ et $t_k \geq 0$ car, soit $k = 1$ et $t_k = t \geq 0$, soit $k \geq 2$ et $t_k \geq 0$ parce que $t_k < 0$ impliquerait que $|t_k| \leq |t_k - s_k|$ ce qui contredit la minimalité de k. Alors $\forall 1 \leq i \leq k-1$, $f(t_i) = 0$ et $f(t_k) = dist(t_k, \mathrm{S}) \leq dist(t_k, s_k) = |t_k - s_k| = |t - (s_1 + \cdots + s_k)|$. Ainsi $f^*(t) \leq f(t_1) + \cdots + f(t_k) \leq dist(t, \mathbb{N}\mathrm{S}) + \varepsilon$, pour tout $\varepsilon > 0$. Ce qui termine la preuve de $f^*(t) = dist(t, \mathbb{N}\mathrm{S})$. ■

Remarquons que dans le cas $f(t) = dist(t, \mathrm{S})$, il est possible de « relever » cette fonction pour obtenir une fonction *croissante* en prenant $f(t) + t$ dont la clôture sous-additive est $f^*(t) + t$ (la croissance marche aussi pour $f(t) = \mathbb{1}_{\overline{\mathrm{S}}}(t)$ mais seulement dans le modèle discret). Pour obtenir une clôture sous-additive continue, il faut que $0 \in \mathrm{S}$ et alors on a $f^*(t) = dist(t, \mathbb{N}\mathrm{S})$ pour tout $t \geq 0$.

4.3. Problèmes de clôture

4.3.1. Fonctions affines par morceaux

La classe des fonctions affines par morceaux est plutôt robuste vis-à-vis des principaux opérateurs du Network Calculus, même sans hypothèse d'ultime pseudo-périodicité. Ceci est illustré dans la proposition suivante, qui résume les résultats de [Bouillard 2007b].

Proposition 4.7 ([Bouillard 2007b]). *Les classes $\mathcal{F}[\mathbb{R}_+,\mathbb{R}]$ et $\mathcal{F}[\mathbb{Q}_+,\mathbb{Q}]$ sont closes pour les opérations $+,-,*$ et pour la clôture sous-additive. La classe des fonctions ultimement pseudo-périodiques dans $\mathcal{F}[\mathbb{Q}_+,\mathbb{Q}]$ est close pour $\oslash$.*

La classe $\mathcal{F}[\mathbb{R}_+,\mathbb{R}]$ n'est pas close pour la déconvolution $\oslash$. La proposition 4.5 donne des contre-exemples, même avec des entrées ultimement pseudo-périodiques : en utilisant les points (i) et (ii) de la proposition, avec des fonctions de périodes β et α telles que $\beta/\alpha \notin \mathbb{Q}$. De tels exemples, qui utilisent la structure de $\mathbb{N}\beta - \mathbb{N}\alpha$ n'apparaissent pas dans $\mathcal{F}[\mathbb{Q}_+,\mathbb{Q}]$. La proposition 4.5 (ii) introduit un autre type de contre-exemple qui s'applique à $\mathcal{F}[\mathbb{Q}_+,\mathbb{Q}]$ où la propriété « franche » est perdue. Cependant, les fonctions de ce dernier exemple ne peuvent pas être relevées en fonctions croissantes (avec la proposition 4.3), et sont donc assez inhabituelles en NC. On peut se demander si la classe des fonctions croissantes dans $\mathcal{F}[\mathbb{Q}_+,\mathbb{Q}]$ est close pour $\oslash$. La prochaine proposition montre que sans ultime pseudo-périodicité, le caractère affine par morceau peut être perdu.

Proposition 4.8. *Soit h une fonction convexe C^1 sur $[0,1]$. Alors il existe $f,g \in \mathcal{F}[\mathbb{N},\mathbb{R}]$ tel que $h = f \oslash g$ sur $[0,1]$.*

Pour exhiber de telles fonctions, nous utilisons les lemmes suivants.

Lemme 4.9. *Soit ϕ une application de $\mathbb{N}$ dans $[0,1]$ tel que $\phi(\mathbb{N})$ est dense dans $[0,1]$ (par exemple, une surjection $\mathbb{N} \to \mathbb{Q} \cap [0,1]$). Soit p_x l'équation de la tangente à h en x (i.e. $p_x(t) = \sigma_x + \rho_x \cdot t = h'(x)\cdot(t-x)+h(x)$). Alors $h = \sup_{n\in\mathbb{N}} p_{\phi(n)}$.*

Nous ne donnons pas ici la preuve de ce résultat classique qui utilise le fait que le graphe d'une fonction convexe est toujours au-dessus de ses tangentes.

Lemme 4.10. *Pour tout $n \geq 0$, prenons les fonctions définies sur $\mathbb{R}$ par*

$$f_n(t) = \begin{cases} p_{\phi(n)}(t-2n) & \text{sur } [2n,2n+1], \\ -\infty & \text{sinon.} \end{cases}$$

$$g_n(t) = \begin{cases} \alpha_n \cdot (t-2n) & \text{sur } [2n,2n+1], \\ +\infty & \text{sinon.} \end{cases}$$

où α_n est une constante quelconque plus grande que $\rho_{\phi(n)} = h'(\phi(n))$.
Alors

$$\forall i \in \mathbb{N},\ f_i \oslash g_i = p_{\phi(i)} \text{ on } [0,1],$$

$$\forall i \neq j \in \mathbb{N},\ f_i \oslash g_j = -\infty \text{ on } [0,1].$$

Ce lemme est une application directe du lemme 8 de [Bouillard 2007b] qui décrit la déconvolution de deux segments.

Preuve de la proposition 4.8. Soit h une fonction convexe C^1 sur $[0,1]$, et prenons les fonctions f_n et g_n définies au lemme 4.10 ; combinons ces segments en calculant

$$f = \sup_{i\in\mathbb{N}} f_i \text{ et } g = \inf_{j\in\mathbb{N}} g_j$$

La distributivité établie dans la proposition 4.1, ainsi que les lemmes 4.9 et 4.10 impliquent que :

$$\begin{aligned} f \oslash g &= \sup_{i,j} f_i \oslash g_j \\ &= \sup_{i} f_i \oslash g_i \quad \text{sur } [0,1] \\ &= \sup_{i} p_{\phi(i)} \quad \text{sur } [0,1] \\ &= h \quad \text{sur } [0,1]. \end{aligned}$$

■

4.3.2. Généralisation

Les deux fonctions f et g ainsi construites peuvent sembler pathologiques et éloignées des fonctions que l'on peut rencontrer en Network Calculus (elles ne sont pas croissantes et leur support est $\cup_{n\in\mathbb{N}}[2n, 2n+1]$). Cependant, on peut partir de cet exemple pour généraliser.

Proposition 4.11. *Soit h une fonction convexe C^1 sur $[0,1]$ telle que $h'(0) > 0$, donc* strictement croissante. *Alors il existe des fonctions* croissantes continues $f, g \in \mathcal{F}[\mathbb{N}, \mathbb{R}]$ *telles que $h = f \oslash g$ sur $[0,1]$.*

Preuve. Le principe reste le même mais on fera attention aux segments sur les intervalles $[2n, 2n+1]$.

Nous définissons f et g comme des fonctions croissantes continues telles que pour tout $n \in \mathbb{N}$,

$$f = f_n \text{ sur } [2n, 2n+1],\ f = \tilde{f}_n \text{ sur } [2n+1, 2n+2],$$

$$g = g_n \text{ sur } [2n, 2n+1],\ g = \tilde{g}_n \text{ sur } [2n+1, 2n+2],$$

où :

- f_n est un segment sur $[2n, 2n+1]$ et égale à $-\infty$ partout ailleurs ;
- $\tilde{f}_n$ est un segment sur $[2n+1, 2n+2]$ et égale à $-\infty$ ailleurs ;
- g_n est un segment sur $[2n, 2n+1]$ et égale à $+\infty$ ailleurs ;
- $\tilde{g}_n$ est un segment sur $[2n+1, 2n+2]$ et égale à $+\infty$ ailleurs.

Les fonctions f_n et g_n correspondent aux fonctions introduites dans la preuve de la

proposition 4.8, et les fonctions $\tilde{f_n}$ et $\tilde{g_n}$ sont des fonctions d'ajustement choisies pour assurer la forme du résultat, mais sans changer la déconvolution sur $[0,1]$.

Nous définissons tout d'abord f_n et g_n sur $[2n, 2n+1]$ par :

$$\begin{aligned} f_n(t) &= h'(\phi(n)) \cdot (t-2n) + f(2n) \\ g_n(t) &= h'(\phi(n)) \cdot (t-2n) + g(2n). \end{aligned}$$

Donc f_n et g_n sont des segments parallèles et nous définirons $f(2n)$ et $g(2n)$ de telle sorte que $f(2n) - g(2n) = \sigma_{\phi(n)}$.

Pour l'intervalle $[2n+1, 2n+2]$, nous considérons deux cas. Nous savons que $f(2n+1) - g(2n+1) = \sigma_{\phi(n)}$ (car f_n et g_n sont parallèles et $f(2n) - g(2n) = \sigma_{\phi(n)}$), donc :

- si $\sigma_{\phi(n+1)} \geq \sigma_{\phi(n)}$,

  $\tilde{f_n} =$ segment tel que $\begin{cases} \tilde{f_n}(2n+1) = f_n(2n+1) \\ \tilde{f_n}(2n+2) = f_n(2n+1) + \sigma_{\phi(n+1)} - \sigma_{\phi(n)} \end{cases}$ et

  $\tilde{g_n} =$ segment tel que $\tilde{g_n}(2n+2) = \tilde{g_n}(2n+1) = g_n(2n+1)$.

- $\sigma_{\phi(n+1)} < \sigma_{\phi(n)}$,

  $\tilde{f_n} =$ segment tel que $\tilde{f_n}(2n+2) = \tilde{f_n}(2n+1) = f_n(2n+1)$ et

  $\tilde{g_n} =$ segment tel que $\begin{cases} \tilde{g_n}(2n+1) = g_n(2n+1) \\ \tilde{g_n}(2n+2) = g_n(2n+1) + \sigma_{\phi(n)} - \sigma_{\phi(n+1)}. \end{cases}$

Dans les deux cas, nous garantissons $f(2n+2) - g(2n+2) = \tilde{f_n}(2n+2) - \tilde{g_n}(2n+2) = \sigma_{\phi(n+1)}$; et f et g sont continues et croissantes.

La distributivité de la déconvolution implique que, sur $[0,1]$,

$$f \oslash g = \sup_i (f|_{[i,i+1]} \oslash g|_{[i,i+1]}) \vee \sup_i (f|_{[i+1,i+2]} \oslash g|_{[i,i+1]})$$

Cette équation peut donc se réécrire sur $[0,1]$,

$$f \oslash g = \sup_{n\in\mathbb{N}} (f_n \oslash g_n) \vee \sup_{n\in\mathbb{N}} (\tilde{f_n} \oslash \tilde{g_n}) \vee \sup_{n\in\mathbb{N}} (\tilde{f_n} \oslash g_n) \vee \sup_{n\in\mathbb{N}} (f_{n+1} \oslash \tilde{g_n}). \tag{4.3}$$

On choisit ϕ de telle sorte que les pentes des segments d'ajustement des fonctions f et g sur $\cup_{n\in\mathbb{N}}[2n+1, 2n+2]$ soient inférieures ou égales aux pentes sur $\cup_{n\in\mathbb{N}}[2n, 2n+1]$.

Les pentes sur $\cup_{n\in\mathbb{N}}[2n, 2n+1]$ sont toutes $\geq h'(0)$. Soit α quelconque tel que $0 < \alpha \leq h'(0)$ (ce qui est possible car $h'(0) > 0$). L'application $x \mapsto \sigma_x = h(x) - xh'(x)$ est continue sur le compact $[0,1]$, et donc uniformément continue, et $\exists \varepsilon > 0$, $\forall x, y \in [0,1]$, $|x-y| \leq \varepsilon \implies |\sigma_x - \sigma_y| \leq \alpha$.

La pente de $\tilde{f_n}$ ou $\tilde{g_n}$ est toujours 0 ou $|\sigma_{\phi(n+1)} - \sigma_{\phi(n)}|$, et il suffit de choisir ϕ tel que $\forall n \in \mathbb{N}$, $|\phi(n+1) - \phi(n)| \leq \varepsilon$. Il faut aussi que $\phi(\mathbb{N})$ soit dense dans $[0,1]$ pour assurer $h = \sup_{n\in\mathbb{N}} p_{\phi(n)}$.

Une manière de remplir les deux contraintes sur ϕ est de fixer une constante k telle que

$\frac{1}{2^k} \leq \varepsilon$, et

$$\forall i,\ 0 \leq i < 2^k,\ \phi(i) = \frac{i}{2^k}$$
$$\forall i,\ 0 \leq i < 2^{k+1},\ \phi(i + 2^k) = 1 - \frac{i}{2^{k+1}}$$
$$\forall i,\ 0 \leq i < 2^{k+2},\ \phi(i + 2^k + 2^{k+1}) = \frac{i}{2^{k+2}}$$
$$\dots$$

Ainsi pour k fixé, et n parcourant $\mathbb{N}$, les valeurs successives $\phi(n)$ font des aller-retours sur $[0,1]$, en raffinant la subdivision à chaque nouvelle passe (voir fugure 4.3).

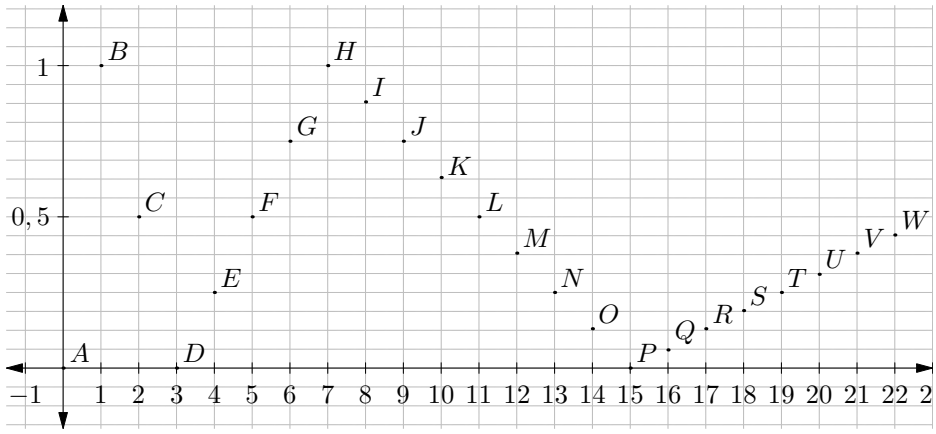
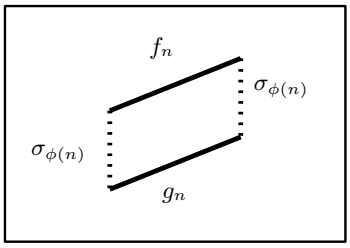


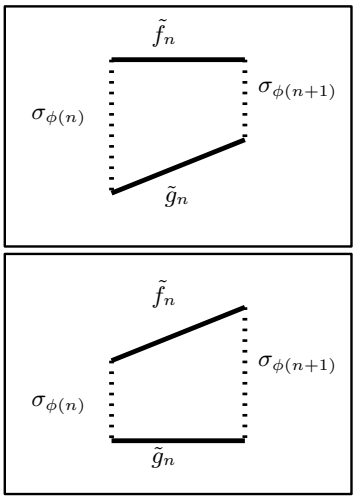
Figure 4.3 : Exemple de fonction ϕ, pour $k = 1$

Nous analysons maintenant les termes de l'équation 4.3. Nous donnons directement la sortie sur $[0,1]$ de la déconvolution des segments, en utilisant le lemme 8 de [Bouillard 2007b].

Pour tout $n \in \mathbb{N}$, les prochaines égalités sont vraies sur $[0,1]$.

- $f_n \oslash g_n = p_{\phi(n)}$

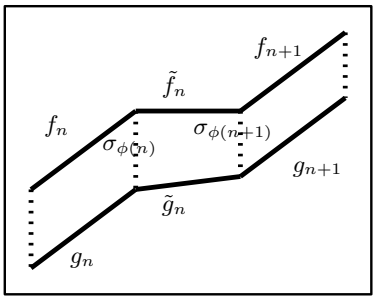




- $\tilde{f}_n \oslash \tilde{g}_n(t) =$ constante
  $= \sigma_{\phi(n)}$ si $\sigma_{\phi(n)} > \sigma_{\phi(n+1)}$
  $\leq \sigma_{\phi(n)} + \rho_{\phi(n)} t$
  $\leq f_n \oslash g_n(t)$.

  $= \sigma_{\phi(n+1)}$ si $\sigma_{\phi(n)} \leq \sigma_{\phi(n+1)}$
  $\leq \sigma_{\phi(n+1)} + \rho_{\phi(n+1)} t$
  $\leq f_{n+1} \oslash g_{n+1}(t)$,

donc les termes de la forme $\tilde{f}_n \oslash \tilde{g}_n$ n'apparaissent pas dans $f \oslash g$.



- $\tilde{f}_n \oslash g_n(t) =$ (pente de $\tilde{f}_n$) $\times t +$ valeur en 0
  $=$ (pente de $\tilde{f}_n$) $\times t + \sigma_{\phi(n)}$
  $\leq \rho_{\phi(n)} t + \sigma_{\phi(n)}$
  $\leq f_n \oslash g_n(t)$.

- $f_{n+1} \oslash \tilde{g}_n(t) =$ (pente de $\tilde{g}_n$) $\times t +$ valeur en 0
  $=$ (pente de $\tilde{g}_n$) $\times t + \sigma_{\phi(n+1)}$
  $\leq \rho_{\phi(n+1)} t + \sigma_{\phi(n+1)}$
  $\leq f_{n+1} \oslash g_{n+1}(t)$,

donc ces termes n'apparaissent pas dans $f \oslash g$.

Au final, sur $[0,1]$, on a $f \oslash g = \sup_{n\in\mathbb{N}} f_n \oslash g_n = \sup_{n\in\mathbb{N}} p_{\phi(n)} = h$. ■

Remarque 4.4.

- *Si h est une fonction rationnelle (quotient de polynômes) avec des coefficients dans $\mathbb{Q}$, alors on a de plus $f, g \in \mathcal{F}[\mathbb{N}, \mathbb{Q}]$.*
- *Ces constructions restent valides sur $]0,1[$ où h est C^1 est convexe (par exemple $t \mapsto 1/(1-t)$). Pour construire la fonction ϕ qui doit balayer $]0,1[$, il faudra faire des allers-retours sur des intervalles fermés $[\ell_n, r_n]$ tels que la suite ℓ_n (resp. r_n) soit décroissante (resp. croissante) et tende vers 0 (resp. 1) ; et à chaque intervalle $[\ell_n, r_n]$ on prendra ε_n (et pas ε) pour utiliser la continuité uniforme.*

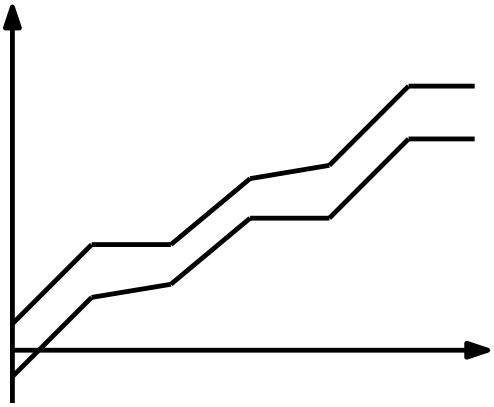


FIGURE 4.4 : Illustration de la forme de f et g. On remarquera que les deux courbes peuvent se croiser, mais comme l'incrément de $\sigma_{\phi(n)}$ est borné par α, l'évolution de la distance entre les deux courbes est bornée

4.3.3. Résultats de stabilité : résumé en image

La figure 4.5, tirée de l'article de Bouillard et Thierry [Bouillard 2007b] résume de nombreux résultats de stabilités de classes de fonctions. Chaque flèche indique dans quelle classe de fonction est le résultat lorsqu'on applique une des opérations qui étiquettent cette flèche. Lorsqu'une flèche pointe en dehors de son point de départ, cela signifie qu'il existe des exemples de fonctions dont les images par les opérateurs n'appartiennent plus à la classe initiale.

Les preuves peuvent être trouvées dans [Bouillard 2007b]. Les opérateurs pris en compte sont les opérateurs classiques du Network calculus.

Ces résultats sont d'un intérêt pratique évident : dès lors que l'on souhaite implémenter les opérateurs du Network Calculus, il est nécessaire de savoir dans quelle classe de fonction sont les résultats pour deux raisons :

- il est très fréquent de *composer* ces opérateurs, et il est donc nécessaire que la composition d'opérateurs soit bien définie pour pouvoir appliquer un autre opérateur à un résultat obtenu par calcul ;
- de même, les bornes cherchées étant déterministes et conservatives, on ne peut pas accepter la possibilité d'erreurs de calculs, dues à des arrondis par exemple.
  - soit on accepte les erreurs d'arrondi, qui peuvent se propager de manière importante par composition, et les bornes obtenues ne sont plus sûres et peuvent devenir fausses,
  - soit on impose des calculs exacts (par exemple avec des valeurs rationnelles) mais les calculs peuvent se révéler coûteux en espace et en temps,
  - soit on décide d'accepter d'éventuels arrondis mais en gardant des bornes justes, et il faut alors accepter que les bornes calculées ne soient pas *tight* même dans le cas

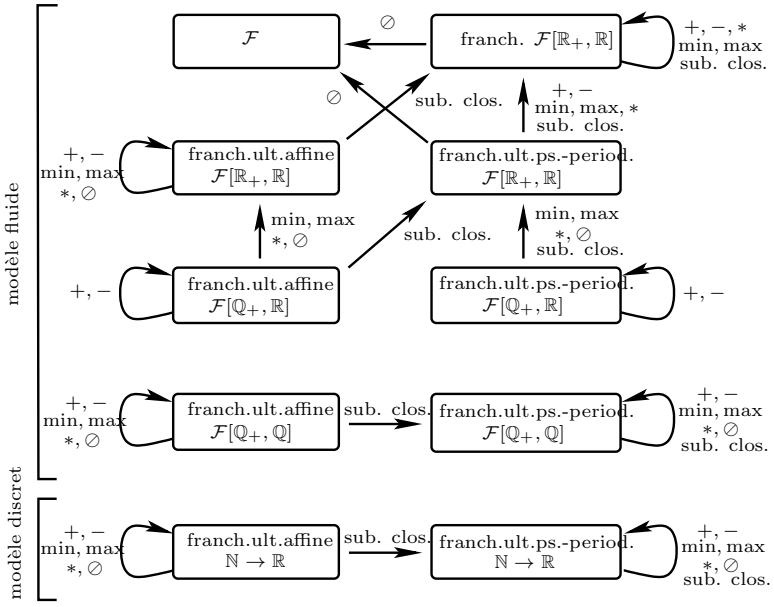


FIGURE 4.5 : Stabilité des classes de fonctions [Bouillard 2007b]

où elles le seraient en théorie. Cette dernière possibilité demanderait de nouveaux résultats théoriques pour calculer les arrondis de manière efficace et sûre.

En pratique les résultats de la figure 4.5 nous apprennent en particulier que :

- dans le cas discret comme continu, la clôture sousadditive (*sub. clos.*) nécessite de pouvoir travailler avec des fonctions ultimement pseudo périodiques ;
- dès lors que l'on souhaite travailler avec un modèle de données *fluide* ($\mathcal{F}[\mathbb{Q}_+,\mathbb{R}]$), il faut aussi travailler avec un *temps continu*, et en fait dans la classe de fonctions $\mathcal{F}$;
- il est cependant possible de travailler avec des *données* et un *temps* dans $\mathbb{Q}$ car la classe des fonctions franches ultimement pseudo périodiques et dans $\mathcal{F}[\mathbb{Q}_+,\mathbb{Q}]$ est stable pour les opérateurs considérés.

On pourra par exemple se référer à l'implémentation de [Boyer 2011] qui laisse le choix de plusieurs classes de fonctions (valeurs rationnelles ou valeurs réelles) selon que l'on souhaite des résultats exacts (vis-à-vis de la théorie) ou des calculs plus rapides.

4.4. Reconstruire une fonction à l'aide des opérateurs du Network Calculus

Nous nous demandons maintenant s'il est possible, étant donné une fonction affine par morceaux ultimement pseudo-périodique, de la construire à partir d'un ensemble de fonctions affines et des opérateurs classiques du NC. Nous souhaitons ainsi pouvoir exprimer toute fonction affine par morceau ultimement pseudo-périodique à l'aide d'une

expression de taille finie ne faisant intervenir que des fonctions affines et des opérateurs du NC.

Nous donnons cette construction lorsque les fonctions considérées sont *continues*, et discutons du cas discontinu.

Dans cette section, toutes les fonctions considérées sont affines par morceaux, et définies sur $\mathbb{R}$ ou $\mathbb{R}_+$.

Notations

Droite de pente p passant par un point A

Dans la suite on utilisera souvent, sans détailler les calculs, le résultat (de lycée) suivant. La droite $\mathrm{D}_{p,\mathrm{A}}$ de pente p et passant par le point $\mathrm{A}(x_\mathrm{A}, y_\mathrm{A})$ est de la forme : $\mathrm{D}_{p,\mathrm{A}} : y = a \times x + b$, avec $a = p$, et vérifie $y_\mathrm{A} = a \times x_\mathrm{A} + b$. Donc la droite $\mathrm{D}_{p,\mathrm{A}}$ est la courbe représentative de la fonction :

$$f_{p,\mathrm{A}} : x \to p \times x + (y_\mathrm{A} - p \times x_\mathrm{A})$$

Fonction à reconstruire

Notons f_{depart} la fonction à reconstruire et $f_n, n \in \mathbb{N}$ les étapes de la fonction que nous construisons de proche en proche (elle sera modifiée au fur et à mesure), ainsi que C_{f_n} la courbe représentative à chaque étape.

On suppose que la fonction à reconstruire f_{depart} est définie sur $\mathbb{R}_+$, et ultimement pseudo-périodique, de rang T, de période d, et d'incrément c (cf. définition 4.1).

$$\exists \mathrm{T} \in \mathbb{R}_+, \exists (c,d) \in \mathbb{R} \times \mathbb{R}_+^*, \forall t > \mathrm{T}, f_{depart}(t+d) = f_{depart}(t) + c$$

Nous allons construire notre fonction de proche en proche, en rajoutant des segments de « gauche à droite » (abscisses croissantes). Nous gardons en permanence en mémoire la pente la plus petite p_{min} et la pente la plus grande p_{max} rencontrées jusqu'ici ($p_{min} \leq p_{max}$).

La pente du dernier segment ajouté sera notée p_{last}.

4.4.1. Partie transitoire d'une fonction continue

Nous considérons seulement la partie transitoire, donc le nombre de segments est fini. Au lieu d'ajouter un segment à chaque étape, nous ajoutons une demi-droite qui coïncide avec le segment voulu. Cette demi-droite sera ensuite en partie supprimée à l'étape suivante lors de l'ajout du « segment » suivant.

Commençons avec la fonction f_0 constante, égale à $f_{depart}(0)$ ($p_{min} = p_{max} = 0$). Les fonctions f_0 et f_{depart} coïncident donc sur l'« intervalle » $[0,0] = 0$.

« Concaténation » d'un nouveau segment [AB], de pente p

Supposons qu'à l'étape n, la fonction objectif soit reconstruite jusqu'au point A, c'est-à-dire

$$f_n(x) = \begin{cases} f_{depart}(x) & \text{pour } 0 \leq x \leq x_{\mathrm{A}} \text{ (partie reconstruite)} \\ f_{p_{last},\mathrm{A}} & \text{pour } x_{\mathrm{A}} \leq x \text{ (demi-droite)} \end{cases}$$

et que nous voulions « rajouter » le segment [AB], de pente p. Ce segment est supporté par la droite $\mathrm{D}_{p,\mathrm{A}}$.

Par construction, C_{f_n} passe par le point A. Nous considérons deux cas :

- Cas $p \leq p_{last}$

$$\text{Si } p \leq p_{min}, \text{ faire } p_{min} := p$$

$$f_{n+1} := min(\underbrace{f_n}_{(1)}, max(\underbrace{f_{p_{min},\mathrm{A}}}_{(2)}, \underbrace{f_{p,\mathrm{A}}}_{(3)}))$$

  On a bien :

$$f_{n+1}(x) = \begin{cases} f_n(x) = f_{depart}(x) & \text{pour } 0 \leq x \leq x_{\mathrm{A}} \text{ car } (2) \geq (3), (1) \leq (2) \\ f_{p,\mathrm{A}}(x) = f_{depart}(x) & \text{pour } x_{\mathrm{A}} \leq x \leq x_{\mathrm{B}} \text{ car } (3) \geq (2), (3) \leq (1) \\ f_{p,\mathrm{A}}(x) & \text{pour } x_{\mathrm{B}} \leq x \text{ car } (3) \geq (2), (3) \leq (1) \end{cases}$$

- Cas $p \geq p_{last}$ (cas symétrique)

$$\text{Si } p \geq p_{max}, \text{ faire } p_{max} := p$$

$$f_{n+1} := max(f_n, min(f_{p_{max},\mathrm{A}}, f_{p,\mathrm{A}}))$$

Et on met à jour $p_{last} := p$. Les $f_{p_{min}}$ et $f_{p_{max}}$ permettent de sélectionner les portions de courbes qui nous intéressent. On notera que c'est à cette étape que l'on utilise la continuité de f_{depart}. La nouvelle fonction obtenue f_{n+1} est reconstruite jusqu'au point B.

On réitère ces étapes pour chacun des segments de f_{depart} (qui sont en nombre fini), et on obtient alors une fonction qui coïncide avec f_{depart} sur sa partie transitoire $[0;\mathrm{T}]$, mais qui termine ensuite par une demi-droite sur $]\mathrm{T};+\infty[$.

Il est important de noter que, les fonctions f_n étant construites par itérations successives, on peut écrire chaque fonction f_n à l'aide d'une seule expression mathématique utilisant uniquement des fonctions affines et des opérateurs du Network Calculus.

4.4.2. Partie pseudo-périodique d'une fonction continue

La partie pseudo-périodique de f_{depart} est de période d, et d'incrément c.

Pour simplifier la suite, on appellera $f_{periode}$ la fonction :

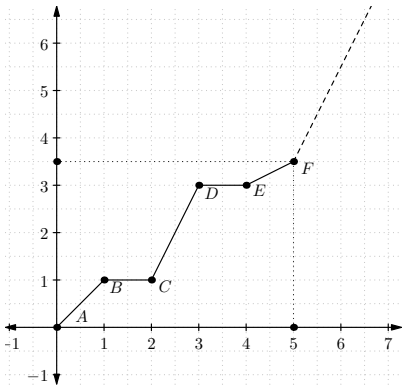


FIGURE 4.6 : Exemple pour $f_{relevement}$: on a relevé $f_{periode}$ pour obtenir une fonction croissante ; la demi-droite finale a la pente la plus grande (la même que [CD])

- reconstruite à partir de f_{depart} ;
- nulle en 0 ;
- dont les variations sur $[0;d]$ sont les mêmes que celles de f_{depart} sur $[\mathrm{T};\mathrm{T}+d]$;
- constituée d'une demi-droite sur $[d;+\infty[$.

Il suffit de commencer la reconstruction vue précédemment à partir du point $(\mathrm{T};f_{depart}(\mathrm{T}))$. En pratique la pente de la demi-droite finale est la même que celle du dernier segment, mais nous ne l'utiliserons pas dans la suite (car elle sera modifiée à l'étape suivante).

Notre but est maintenant de décrire comment obtenir la fonction pseudo-périodique continue de « pseudo-période » $f_{periode}$. Graphiquement, on souhaite donc recoller bout à bout la représentation graphique de $f_{periode}$ une infinité de fois.

Nous allons pour cela utiliser la clôture sous-additive, et la convolution de deux fonctions croissantes ; ainsi que les théorèmes suivants.

Relèvement de $f_{periode}$

La fonction $f_{periode}$ est constituée d'un nombre fini de segments sur $[0,d]$, puis d'une demi-droite.

Soit p_{min} la plus petite valeur parmi les pentes des segments et celle de la demi-droite de $f_{periode}$, et p_{max} la plus grande. On obtient ces valeurs lors de la construction, et on définit $p_{enveloppe} = p_{max} - p_{min}$. Considérons la fonction $f_{relevement}$ définie par

$$f_{relevement}(x) = \begin{cases} f_{periode}(x) - p_{min} \times x & \text{sur } [0;d] \\ f_{periode}(d) - p_{min} \times d + (p_{enveloppe}) \times (x-d) & \text{sur } [d;+\infty[\end{cases}$$

Cette fonction est continue, croissante (au sens large) sur $[0;d]$, puis affine de pente $p_{max} - p_{min}$ $[d;+\infty[$ (voir figure 4.6).

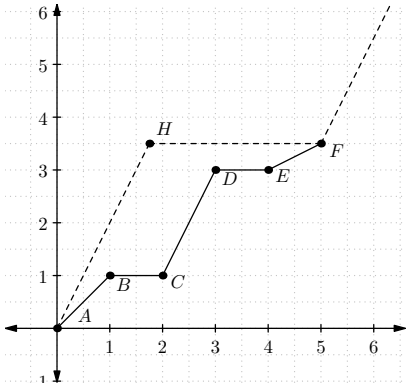


FIGURE 4.7 : Exemple pour g, l'enveloppe de $f_{relevement}$: g est alternativement de pente $p_{enveloppe}$, puis 0, puis $p_{enveloppe}$, où $p_{enveloppe}$ est la pente la plus grande — ici celle de (CD)

Enveloppe de $f_{relevement}$

Nous construisons maintenant la fonction g, définie par :

$$g(x) = \begin{cases} p_{enveloppe} \times x & \text{sur } [0; f_{relevement}(d)/p_{enveloppe}] \\ f_{relevement}(d) & \text{sur } [f_{relevement}(d)/p_{enveloppe}; d] \\ p_{enveloppe} \times x + (f_{relevement}(d) - p_{enveloppe} \times d) & \text{sur } [d; +\infty[\end{cases}$$

Cette fonction g est continue, croissante ; on a $f_{relevement}(d) = g(d)$,

$$\forall x \in [0; d], f_{relevement}(x) \leq g(x)$$

$$\forall x \in [d; +\infty[, f_{relevement}(x) = g(x).$$

Construction de la clôture sous-additive (ou étoile) de g

Pour obtenir la clôture sous-additive, on pourrait calculer $g * g$, puis $g * g * g$ ... cette méthode converge, mais pas en temps fini. Nous utilisons donc évidement directement l'opérateur g^*. Mais cette construction itérative permet de comprendre la forme de g^* : Comme vu page 25, on obtient $g * g$ en faisant « glisser » la courbe de g sur la courbe de g, et on prend l'enveloppe inférieure. On obtient alors deux « marches » au lieu d'une seule. Et ainsi de suite, à chaque étape du calcul de g^*, une marche se construit. La convolution g^* de g est croissante, et pseudo-périodique de période d et d'incrément c (donc on a une infinité de marches). De plus pour tout $x \in [0; d], g^*(x) = f_{relevement}(x)$, voir figure 4.8.

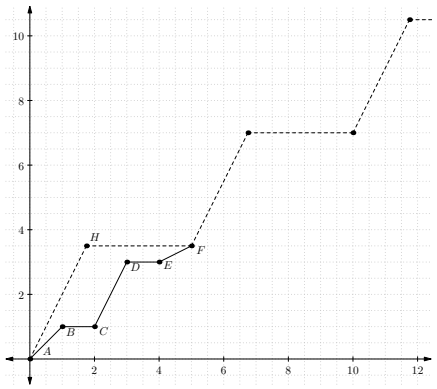


FIGURE 4.8 : Exemple pour g^* : fonction pseudo-périodique, de période d et d'incrément c

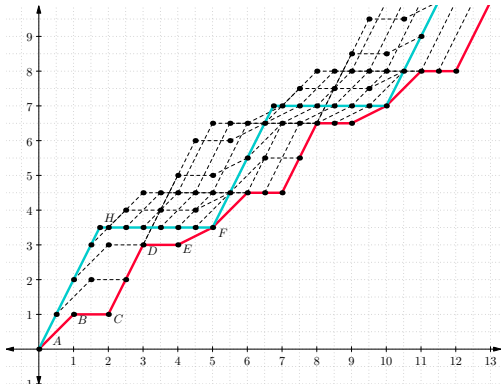


FIGURE 4.9 : Exemple pour $f_{relevement} * g^*$ (en rouge) : fonction pseudo-périodique, de période d et d'incrément c. Sur chaque période, on reconnaît la courbe de $f_{periode}$

Construction de la partie pseudo-périodique de f_{depart} par convolution

De même, on définit (figure 4.9) :

$$f_{relevement,etoile} = f_{relevement} * g^*$$

qui revient à faire « glisser » la courbe de $f_{relevement}$ sur celle de g^*.

Recollement de la partie transitoire et de la partie périodique

Puis, nous décalons $f_{relevement,etoile}$ vers la droite (même principe) :

$$f_{decalage} = f_{relevement,etoile} * max(0, f_{p_{enveloppe},(\mathrm{T},0)}).$$

Cette étape a le même résultat qu'une convolution avec δ_{T}, mais la fonction utilisée est continue.

Cependant la courbe obtenue est toujours relevée, et on a $f_{decalage}(\mathrm{T}) = 0$. On peut

obtenir le résultat voulu avec

$$f_{ok}(x) = f_{decalage}(x) - p_{min} \times (x - \mathrm{T}) + f_{depart}(\mathrm{T}).$$

Remarquons que $\forall x \in [\mathrm{T}; +\infty[, f_{ok}(x) = f_{depart}(\mathrm{T})$.

Il faut alors recoller f_{ok} avec la partie transitoire. Ce qui peut se faire avec la même méthode que pour la construction de la partie transitoire (en adaptant la méthode à f_{ok} qui n'est plus une demi-droite mais une concaténation infinie de segments).

Ceci termine la construction dans le cas continu. Nous pouvons remarquer que :

- le but est de définir chaque étape comme l'application d'opérateurs du NC aux fonctions précédemment obtenus. En particulier, on ne construit jamais les courbes obtenue par convolution itérativement, mais on les définit justement comme des convolution. C'est l'opérateur qui nous intéresse ;
- on utilise la continuité lorsqu'on utilise les p_{min} et les p_{max} pour combiner des morceaux de courbes ;
- on se ramène à des convolutions de fonctions croissantes, quitte à relever les fonctions en ajoutant des termes de la forme $x \to \lambda \times x$, mais on n'utilise pas le théorème de relèvement qui permet de calculer directement la convolution de deux fonctions relevées. Ce résultat pourrait peut-être rendre plus simples certaines étapes de la construction ;
- le recollement de la partie transitoire et de la partie pseudo-périodique nécessite lui aussi une fonction f croissante pour le « décalage » selon l'axe des abscisses.

4.4.3. Conclusion et cas avec discontinuités

La construction vue précédemment permet de montrer que toute fonction *continue* affine par morceaux et pseudo-périodique peut être obtenue à partir des fonctions affines par application des opérateurs du Network Calculus. Vu les résultats présents dans [Bouillard 2007b], il faudrait encore montrer que la classe des fonctions continues est stable par les opérateurs du NC, pour montrer que la classe des fonctions continues, affines par morceaux et pseudo-périodiques est exactement la clôture de la classe des fonctions affines par ces opérateurs.

Jusqu'ici nous avons seulement considéré des fonctions continues. Et toutes les opérations décrites ci-dessus utilisaient uniquement des fonctions affines, et des opérateurs classiques en NC. Il est cependant légitime de se poser la même question pour des fonctions discontinues.

La construction proposée pour la partie transitoire ne fonctionne plus dans ce cas. En effet, elle est principalement basée sur la plus petite ou la plus grande pente d'un ensemble de segments, mais les discontinuités correspondent à des segments de « pente infinie ».

Mais si l'on accepte, pour définir un segment [AB] ($x_A < x_B$), des fonctions de la forme :

$$f_{AB} = \begin{cases} +\infty & \text{sur }]-\infty; x_A[\\ d_{(AB)} & \text{sur } [x_A; x_B] \\ +\infty & \text{sur }]x_B; +\infty[\end{cases}$$

où $d_{A,B}$ est la fonction dont la courbe représentative est la droite (AB), il est très facile de construire la partie transitoire (il suffit de prendre le *min* sur tous les segments considérés).

Cependant la construction de la partie pseudo-périodique nous semble plus difficile car il est compliqué de « relever » cette partie pour obtenir une fonction croissante, dont la convolution avec g^* serait facile à calculer. Ce problème reste donc, pour nous, ouvert.

4.5. Fonctions suradditives : non-équivalence de deux propriétés

Nous nous intéressons ici aux fonctions réelles suradditives (définition 2.3)

$$\forall x, y \in \mathbb{R}, f(x+y) \geq f(x) + f(y)$$

et bornées sur un intervalle fini, comme c'est le cas pour la plupart des courbes de service considérées en Network Calculus.

Le lemme de Fekete [Kuczma 2009, p. 463] repris pour les fonctions suradditives assure que pour toute fonction f suradditive à valeurs bornées sur tout intervalle fini, la limite $lim_{t\to+\infty}(\frac{f(t)}{t})$ existe. Remarquons que cette limite peut être infinie.

Étant donné une fonction suradditive f et bornée sur un intervalle fini, nous nous demandons s'il y a équivalence entre les deux propriétés suivantes :

- $lim_{t\to+\infty}\frac{f(t)}{t} = +\infty$;
- $\forall t > 0, f \oslash f(t) = +\infty$.

Remarque 4.5. *Pour fixer les idées, en termes par exemple de courbe de service simple d'un serveur, la première propriété correspond à un* débit *long terme infini, et la deuxième à des rafales arbitrairement grandes. Nous rappelons que la fonction f est suradditive et donc, en particulier, que f n'est pas une fonction en escalier.*

Nous allons montrer que pour une fonction f suradditive,

$$lim_{t\to+\infty}\frac{f(t)}{t} = +\infty \implies \forall t > 0, f \oslash f(t) = +\infty$$

mais que la réciproque n'est pas vraie.

Montrons que pour f suradditive, $lim_{t\to+\infty}\frac{f(t)}{t} = +\infty \Rightarrow \forall t > 0, f \oslash f(t) = +\infty$

Preuve. Nous allons raisonner par l'absurde : soit une fonction f vérifiant $lim_{t\to+\infty}\frac{f(t)}{t} = +\infty$, et supposons que la propriété $\forall t > 0, f \oslash f(t) = +\infty$ n'est pas vérifiée, c'est-à-dire

$$\exists \mathrm{T}, \mathrm{K} \in \mathbb{R}, f \oslash f(\mathrm{T}) \leq \mathrm{K}, \text{ soit (définition 2.6)}$$

$$\exists \mathrm{T}, \mathrm{K} \in \mathbb{R}, sup_{u\geq 0}(f(\mathrm{T}+u) - f(u)) \leq \mathrm{K}$$

On a donc

$$\forall u \in \mathbb{R}, f(\mathrm{T}+u) - f(u) \leq \mathrm{K}. \tag{4.4}$$

Pour tout $t \in \mathbb{R}_+$, soit r_t le reste de la division euclidienne de t par T, c'est-à-dire

$$t = n \times \mathrm{T} + r_t$$

avec $0 \leq r_t < \mathrm{T}$, et $n := \lfloor \frac{t}{\mathrm{T}} \rfloor$. On a donc $t = \underbrace{\mathrm{T} + \mathrm{T} + \cdots + \mathrm{T}}_{n \text{ fois}} + r$, et on peut écrire

$$f(t) - f(r_t) = f(n\mathrm{T} + r_t) - f((n-1)\mathrm{T} + r_t) + f((n-1)\mathrm{T} + r_t) - \cdots - f(\mathrm{T} + r_t) + f(\mathrm{T} + r_t) - f(r_t)$$

et appliquer n fois l'inégalité 4.4, pour obtenir

$$f(t) - f(r_t) \leq n \times \mathrm{K}.$$

On a alors pour tout $t \in \mathbb{R}_+$,

$$\frac{f(t)}{t} \leq \frac{1}{t} \times \lfloor \frac{t}{\mathrm{T}} \rfloor \times \mathrm{K} + \frac{f(r_t)}{t}$$

On peut alors borner $f(r_t)$ par $\mathrm{M} := max_{x\in[0,\mathrm{T}]} f(x)$ qui ne dépend que de T, qui est lui même fixé indépendamment de t. Nous utilisons ici le fait que f est borné sur tout intervalle fini.

Pour tout $t \geq 1$, on a donc $\frac{f(t)}{t} \leq \frac{\mathrm{K}}{\mathrm{T}} + \mathrm{M}$, on donc on ne peut pas avoir $lim_{t\to+\infty}\frac{f(t)}{t} = +\infty$

■

Montrons que pour f suradditive, $lim_{t\to+\infty}\frac{f(t)}{t} = +\infty \nLeftarrow \forall t > 0, f \oslash f(t) = +\infty$

Nous donnons un contre-exemple en définissant la fonction suivante :

Définition 4.3. *Considérons la fonction définie sur* $\mathbb{R}^+$ *par :*

$$f : t \mapsto \left(\sum_{i=1}^{\lceil \sqrt[3]{t} \rceil} \frac{1}{i^2} \right) \times t$$

Cette fonction est linéaire par morceaux (elle est égale à $t \mapsto \left(\sum_{i=1}^{n} \frac{1}{i^2}\right) \times t$ sur chaque intervalle $]n^3;(n+1)^3]$, $n \in \mathbb{N}$) (voir figure 4.10).

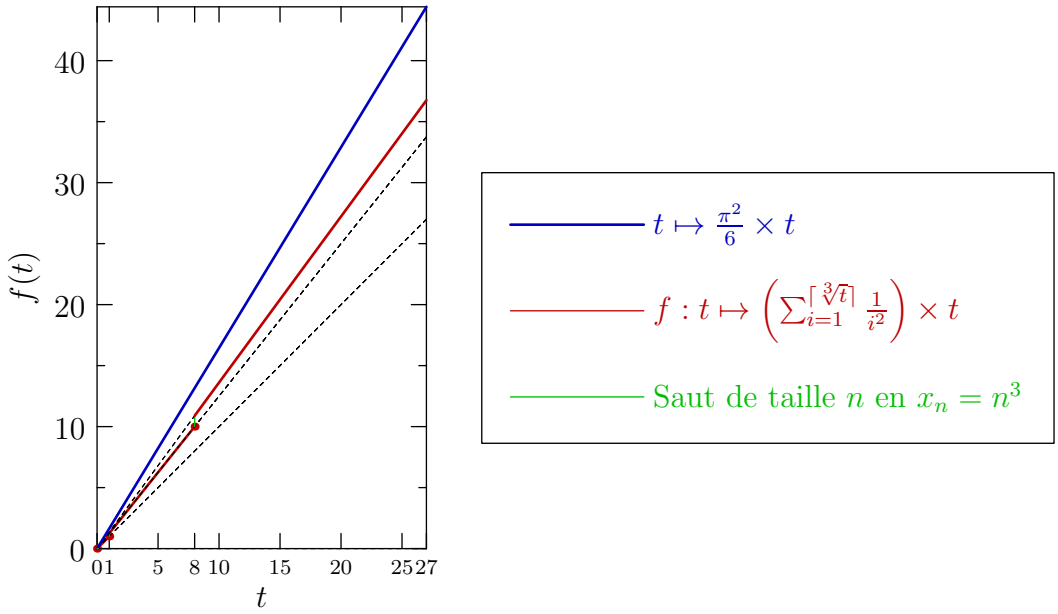


FIGURE 4.10 : Une fonction suradditive qui vérifie $\forall t > 0, f \oslash f(t) = +\infty$ mais pas $\lim_{t\to+\infty} \frac{f(t)}{t} = +\infty$. Discontinuités en $x_1 = 1$, $x_2 = 8$, $x_3 = 27\ldots$

Théorème 4.12. *La fonction* f *de la définition 4.3 a les propriétés suivantes :*

1. *elle est suradditive ;*
2. *elle vérifie :* $\forall t, f \oslash f(t) = +\infty$ *;*
3. *mais elle ne vérifie pas :* $\lim_{t\to+\infty} \frac{f(t)}{t} = +\infty$.

Preuve.

1. La fonction $t \mapsto \frac{f(t)}{t}$ (qui est constante sur les intervalles $]n^3;(n+1)^3]$, $n \in \mathbb{N}$) est croissante sur $\mathbb{R}_*^+$, et $f(0) = 0$. Donc f est étoilée. D'après le théorème 2.1 , elle est suradditive.
2. Pour tout $n \in \mathbb{N}$, f admet une discontinuité en $x_n = n^3$ (de « taille n »). Donc pour tout $t > 0$,

$$f \oslash f(t) = sup_{u\geq 0}(f(t+u) - f(u)) \geq sup_{n\in\mathbb{N}}(f(t+x_n) - f(x_n)) \geq sup_{n\in\mathbb{N}}(n)$$

On a donc $f \oslash f(t) = +\infty, \forall t \in \mathbb{R}_+$.

3. $\forall t \geq 0, f(t) \leq \left(\sum_{i=1}^{\infty} \frac{1}{i^2}\right) \times t = \frac{\pi^2}{6} \times t$.

■

Conclusion

Dans ce chapitre nous nous sommes intéressé aux classes de fonctions utilisables en pratique en Network calculus :

- les trois premières sections donnent des exemples et contre-exemples qui permettent de comprendre les problèmes de stabilité de classes de fonctions en Network Calculus. Le résumé de la figure 4.5 [Bouillard 2007b] permet d'apporter des réponses très concrètes à quiconque voudrait implémenter les opérateurs du Network Calculus. On y voit l'importance de la classe de fonctions $\mathcal{F}$;
- la section 4.4 étudie le problème inverse, à savoir la possibilité de construire n'importe quelle fonction de $\mathcal{F}$ à partir de fonctions affines et des opérateurs du Network Calculus. L'intérêt de cette section est plus théorique, et la réponse apportée est très partielle (elle ne concerne que des fonctions *continues*) ;
- la section 4.5 donne un (contre-)exemple, plus anecdotique, qui montre qu'il est nécessaire de faire attention lorsque l'on interprète des résultats du Network Calculus ; en particulier en ce qui concerne la déconvolution.

CHAPITRE 5

COMPARAISON AVEC RTC

Après une présentation du modèle RTC, ce chapitre fait le lien entre le Greedy Processor de RTC, et le Variable Capacity Node de NC (en se limitant aux courbes de service minimal).

Introduction

Aux côtés du Network Calculus, il existe d'autres théories alternatives qui ont un formalisme très proche (modèles utilisant des enveloppes, opérateurs dans $(\min, +)$) mais qui revendiquent obtenir de meilleures bornes qu'en utilisant le NC, ou être plus appropriées à des applications particulières. On pourra citer le Real-Time Calculus, le Sensor Calculus.

Les similarités/différences avec RTC sont présentées dans le tableau 5.1. La ligne *Applications* donne les applications premières des différentes théories.

On peut se demander si les modèles analysés dans ces théories sont vraiment différents des modèles du Network Calculus. Si c'est le cas, d'où proviennent les différences ? Sont-elles majeures ? Nous essaierons d'étudier en partie ses questions vis-à-vis du RTC.

Nous nous intéressons ici aux modèles non probabilistes et aux modèles dont le but est de calculer des bornes déterministes sur les performances pire-cas.

Dans le cadre probabiliste, plusieurs formalismes existent dans la littérature sous le nom de « Stochastic Network Calculus » [Chang 2000, Ciucu 2005, Fidler 2006b, Jiang 2006, Jiang 2008].

Ils utilisent aussi des contraintes par enveloppes, mais avec des hypothèses probabilistes et ont pour but d'évaluer les performances pire-cas en utilisant des distributions aléatoires.

	Network Calculus (NC)	Real-Time Calculus (RTC)
Math.	Contraintes : enveloppes , algèbre (min, +), fonctions cumulées à une seule variable	Contraintes : enveloppes , algèbre (min, +), fonctions cumulées à deux variables
App.	Réseaux de communication, Internet IntServ & DiffServ	Systèmes embarqués temps-réel
Soft.	DISCO Network Calculator [Schmitt 2006a, Gollan 2008], COINC [COINC], Rockwell Collins ConfGen	RTC Toolbox [Wandeler 2006a, Thiele 2001, Wandeler 2006c, Chakraborty 2003], CyNC [Schioler 2005, Schioler 2007]

Tableau 5.1.: Comparaison de NC et RTC

Nous ne nous intéressons pas ici à ces modèles probabilistes, bien qu'ils semblent très prometteurs.

Complément à propos des Variable Capacity Nodes

Lemme 5.1 (Cohérence, et caractérisation alternative). *Soit* $A, C \in \mathcal{F}_{\nearrow}$ *et* $B \in \mathcal{F}$ *tel que* $\forall t \geq 0$, $B(t) = \inf_{0 \leq s \leq t} \left[A(s) + C(t) - C(s) \right]$. *On a alors* $B \in \mathcal{F}_{\nearrow}$ *et* $\forall t \geq 0$,

$$B(t) = A(start(t)) + C(t) - C(start(t)).$$

Preuve. Comme $A(0) = C(0) = 0$, on a $B(0) = 0$. Et par hypothèse, $\forall t \geq 0$, $B(t) \leq A(t)$ (prendre $s = t$) et $B(t) \leq C(t)$ (prendre $s = 0$).

Comme A et C sont continues à gauche, la définition de B nous assure qu'elle est aussi continue à gauche.

Soit $t < t'$. Pour tout s, on a $A(s) + C(t') - C(s) = (A(s) + C(t) - C(s)) + (C(t') - C(t))$. De plus, puisque A et C sont croissantes, nous avons $C(t') - C(t) \geq 0$, ainsi que $\inf_{t \leq s \leq t'}(A(s)) = A(t)$, et $\inf_{t \leq s \leq t'}(-C(s)) = -C(t')$. D'où :

$$B(t') = \inf\Bigg[\underbrace{\inf_{0 \leq s \leq t} \Big(A(s) + C(t') - C(s) \Big)}_{= B(t) + C(t') - C(t) \geq B(t)}, \underbrace{\inf_{t \leq s \leq t'} \Big(A(s) + C(t') - C(s) \Big)}_{= A(t) + C(t') - C(t') = A(t) \geq B(t)} \Bigg]$$

Ainsi $B(t') \geq B(t)$, et donc B est croissante.

Pour tout $t \geq 0$, $start(t)$ est bien défini grâce à la continuité à gauche de A et B et au fait que $A(0) = B(0) = 0$. Pour tout $t \geq 0$, comme $B(start(t)) = A(start(t))$, et en écrivant

$A(s)+C(t)-C(s) = (A(s)+C(start(t)-C(s))+(C(t)-C(start(t))$, nous avons

$$B(t) = \inf\Bigg[\underbrace{\inf_{0\le s\le start(t)}\Big[A(s)+C(t)-C(s)\Big]}_{\begin{array}{l}= \text{B(start(t))+C(t)-C(start(t))}\\ = \text{A(start(t))+C(t)-C(start(t))}\end{array}} , \inf_{start(t)\le s\le t}\Big[A(s)+C(t)-C(s)\Big]\Bigg]$$

Le premier terme $A(start(t))+C(t)-C(start(t))$ se retrouve dans le second terme lorsque $s = start(t)$. Ainsi, pour tout $t \ge 0$,

$$B(t) = \inf_{start(t)\le s\le t}\Big[A(s)+C(t)-C(s)\Big] \tag{VCN}$$

De plus, si $B(t) = A(t)$, nous avons $start(t) = t$ et ainsi $A(start(t))+C(t)-C(start(t)) = A(t) = B(t)$, ce qui termine la preuve. Sinon, $B(t) < A(t)$ (et donc $start(t) < t$). Dans ce cas,

$$\forall start(t) < s \le t,\ A(s) > \inf_{start(s)\le u\le s}\Big[A(u)+C(s)-C(u)\Big]$$

avec $start(u) = start(t)$ par définition. Ainsi $\forall start(t) < s \le t$,

$$A(s)+C(t)-C(s) > \inf_{start(t)\le u\le s}\Big[A(u)+C(t)-C(u)\Big] \ge \inf_{start(t)\le u\le t}\Big[A(u)+C(t)-C(u)\Big] = B(t).$$

Ainsi l'infimum dans l'équation (VCN) est nécessairement atteint en $s = start(t)$, c'est-à-dire

$$B(t) = A(start(t))+C(t)-C(start(t)).$$

Notons que pour prouver cette formule nous n'avons pas utilisé la croissance des fonctions A, B, C. ■

5.1. Présentation de RTC

Le Real-Time Calculus est généralement présenté comme une alternative au Network Calculus.

Même si une certaine parenté avec le NC est communément admise (enveloppes pour modéliser le trafic entrant et le service), le RTC revendique des caractéristiques spécifiques qui permettraient d'avoir une meilleure modélisation du réseau, ainsi que des bornes plus précises, et adaptées à des systèmes plus complexes.

Dans la littérature ([Wandeler 2006c]), les comparaisons entre RTC et NC sont souvent informelles. Dans ce chapitre nous essayons donc de comparer les modèles RTC et NC : plus précisément, nous comparons les *RTC Greedy Processors (GP)* et les *NC Variable Capacity Nodes*.

Remarque 5.1. *À noter que les éléments de présentation de RTC qui suivent sont tirés de la thèse de doctorat [Wandeler 2006a], très complète et précise sur le sujet du Real-Time Calculus ; en particulier de l'annexe A, page 197. Toutes les notations RTC qui suivent viennent de cette thèse.*

Espace de fonctions étudiées

RTC Greedy Processor	NC Variable Capacity Node
$\mathrm{R}[s,t)$, $s \leq t$, $s,t \in \mathbb{R}$, positive, vérifiant la relation de Chasles : $\forall u \leq v \leq w$, $\mathrm{R}[u,w) = \mathrm{R}[u,v) + \mathrm{R}[v,w)$.	$\mathrm{R}(t)$, $t \in \mathbb{R}$ ou plus souvent $t \in \mathbb{R}_+$, croissante et $\mathrm{R}(0) = 0$.

Ci-dessous, une liste des principales différences trouvées dans la littérature entre les objets manipulé en NC et en RTC [Wandeler 2006c] :

- les variables temporelles en RTC prennent leurs valeurs dans $\mathbb{R}$ (et pas seulement $\mathbb{R}_+$ comme c'est souvent le cas en NC) ;
- en RTC, on n'a pas besoin d'un instant $t = 0$ (comme en NC) où toutes les fonctions considérées sont nulles ;
- les fonctions de RTC ont deux arguments temporels et pas seulement un comme en NC.

Relation de Chasles

Cependant, une autre hypothèse faite en RTC est souvent omise dans les comparaisons RTC/NC : les preuves en RTC utilisent fréquemment la relation de Chasles sans pour autant que celle-ci soit clairement stipulée dans les hypothèses de départ. Bien entendu, la relation de Chasles est tout à fait naturelle étant donné que $\mathrm{R}[s,t)$ représente la quantité de données traitée (ou le service fourni) dans l'intervalle de temps $[s,t[$.

Dans la suite, nous étudions l'expressivité de RTC [Wandeler 2006a, Wandeler 2006c] en prenant en compte la relation de Chasles.

NC vers RTC

Soit $\mathrm{R}(.)$ une fonction définie sur $\mathbb{R}$ (si elle est définie sur $\mathbb{R}_+$, on posera $\mathrm{R}(t) = 0$ pour $t < 0$). Définissons $\hat{\mathrm{R}}[s,t) = \mathrm{R}(t) - \mathrm{R}(s)$ pour tout $s \leq t$. On vérifiera que $\hat{\mathrm{R}}$ vérifie la relation de Chasles et, si R est croissante, alors $\hat{\mathrm{R}}$ est positive.

RTC vers NC

Soit $\mathrm{R}[.,.)$ une fonction définie sur $\{(s,t) \in \mathbb{R}^2 \mid s \leq t\}$. Définissons $\check{\mathrm{R}}(t) = \mathrm{R}[0,t)$ pour $t \geq 0$ et $= -\mathrm{R}[t,0)$ pour $t < 0$. Si R vérifie la relation de Chasles, on peut vérifier que $\check{\mathrm{R}}(0) = 0$ et $\check{\mathrm{R}}(t) - \check{\mathrm{R}}(s) = \mathrm{R}[s,t)$ pour tout $s \leq t$ (c'est-à-dire $\hat{\check{\mathrm{R}}} = \mathrm{R}$). Ainsi, si R est positive, alors $\check{\mathrm{R}}$ est

croissante (on a aussi $\check{\mathrm{R}} = \mathrm{R}[p,t)$ pour tout $p \in \mathbb{R}$, mais alors $\check{\mathrm{R}}(0)$ n'est pas nécessairement nul).

Proposition 5.2. *L'application* $\mathrm{R} \mapsto \check{\mathrm{R}}$ *est une bijection entre les fonctions positives sur* $\{(s,t) \in \mathbb{R}^2 \mid s \leq t\}$ *qui vérifient la relation de Chasles et les fonctions croissantes sur* $\mathbb{R}$ *qui s'annulent en* $t = 0$. *Sa réciproque est l'application* $\mathrm{R} \mapsto \hat{\mathrm{R}}$.

Ainsi, l'utilisation de fonctions à deux arguments en RTC est plus justifiée par la simplicité d'utilisation et d'écriture des équations que par la puissance de modélisation.

Équations RTC

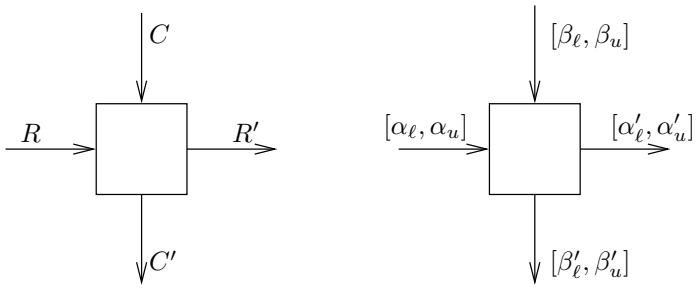


Figure 5.1 : Un système entrée/sortie à un flux en RTC : (a) entrée/sortie ; (b) contraintes

RTC Greedy Processor	**NC Variable Capacity Node**
Temps : $t \in \mathbb{R}$ Entrée : R, C Sortie : R', C' Backlog : B	Temps : $t \in \mathbb{R}_+$ Entrée : R, C Sortie : R', C' Backlog : B
(A7) $\mathrm{R}'[s,t) = \mathrm{C}[s,t) - \mathrm{C}'[s,t)$ (A8) $\mathrm{C}'[s,t) = \sup_{s \leq u \leq t} \left[\mathrm{C}[s,u) - \mathrm{R}(s,u) - \mathrm{B}(s), 0\right]$ (A10) $\mathrm{B}(t) - \mathrm{B}(s) = \mathrm{R}[s,t) - \mathrm{R}'[s,t)$	(B1) $\mathrm{R}'(t) = \inf_{0 \leq u \leq t} \left[\mathrm{C}(t) - \mathrm{C}(u) + \mathrm{R}(u)\right]$ (B2) $\mathrm{C}'(t) = \mathrm{C}(t) - \mathrm{R}'(t)$ (B3) $\mathrm{B}(t) = \mathrm{R}(t) - \mathrm{R}'(t)$

En examinant les équations pour le Variable Capacity Node, on remarque aisément que pour *toutes* fonctions R et C en entrée, il existe des fonctions R', C' en sortie qui vérifient les équations, et qu'elle sont définies de manière unique ((B1) donne R′ à partir de R et C, puis (B2) donne C′ à partir de R′ et C). La charge (backlog) est aussi bien déterminée de manière unique (avec (B3)).

Lors de l'examen des équations RTC, pour des entrées R et C, l'existence de fonctions $\mathrm{R}', \mathrm{C}', \mathrm{B}$ qui satisfont les équations est moins évidente (deux fonctions inconnues par équation).

Lemme 5.3 (Cohérence des équations RTC)**.** *Étant donné* R, C, *soit* $\mathrm{R}', \mathrm{C}', \mathrm{B}$ *un ensemble de solutions (s'il en existe) au sytème d'équations* (A7), (A8), (A10). *Si* R *et* C *vérifient la relation*

de Chasles, alors R′ *et* C′ *aussi. Dans ce cas, si* R *et* C *sont de plus positives, alors* B,R′ *et* C′ *le sont aussi.*

Preuve. Sachant que C vérifie la relation de Chasles, et vue l'équation (A7), il suffit de montrer que soit R′ soit C′ vérifie cette relation pour que les deux la vérifient.

Soit $s \leq u \leq t$, on a alors :

$$\begin{aligned}
C'[s,u)+C'[u,t) &\overset{(A8)}{=} C'[s,u)+\sup_{u\leq w\leq t}\Big[\underbrace{C[u,w)}_{C[s,w)-C[s,u)} - \underbrace{R[u,w)}_{R[s,w)-R[s,u)} - B(u),0\Big] \\
&\overset{(A10)}{=} C'[s,u)+\sup_{u\leq w\leq t}\Big[C[s,w)-R[s,w)+R'[s,u)-C[s,u)-B(s),0\Big] \\
&\overset{(A7)}{=} C'[s,u)+\sup_{u\leq w\leq t}\Big[C[s,w)-R[s,w)-C'[s,u)-B(s),0\Big] \\
&= \sup_{u\leq w\leq t}\Big[C[s,w)-R[s,w)-B(s),C'[s,u)\Big] \\
&= \sup_{u\leq w\leq t}\Big[C[s,w)-R[s,w)-B(s),\sup_{s\leq w\leq u}[C[s,w)-R[s,w)-B(s),0]\Big] \\
&= \sup_{s\leq w\leq t}\Big[C[s,w)-R[s,w)-B(s),0\Big] \\
&= C'[s,t).
\end{aligned}$$

Ainsi, si C′ vérifie la relation de Chasles, alors pour tout $s \in \mathbb{R}$, $C'[s,s) = 0$. Soit $\sup\Big[C[s,s)-R[s,s)-B(s),0\Big] = 0$ et alors $B(s) \geq 0$. De plus, comme C′ est définie comme un $sup[...,0]$, on a $C'[s,t) \geq 0$ pour tout $s \leq t$. Alors, vérifier si $R'[s,t) \geq 0$ revient à vérifier que $C'[s,t) \leq C[s,t)$. Ce qui est vrai car $C'[s,t) = \sup_{s\leq u\leq t}\Big[\underbrace{C[s,u)}_{\leq C[s,t)} - \underbrace{R[s,u)}_{\geq 0} - \underbrace{B(s)}_{\geq 0}, \underbrace{0}_{\leq C[s,t)}\Big]$. ■

Lemme 5.4 (Une équation récursive)**.** *Soit* $A \in \overline{\mathbb{R}}^{\mathbb{R}}$*. Alors les fonctions* $F \in \overline{\mathbb{R}}^{\mathbb{R}}$ *qui vérifient :*

$$\forall s \leq t,\ F(t) = \inf\Big(\inf_{s\leq u\leq t} A(u), F(s)\Big)$$

sont exactement les fonctions de la forme $F(t) = \inf\Big(\inf_{-\infty\leq u\leq t} A(u), a\Big)$ *où* $a \in \overline{\mathbb{R}}$ *est une constante.*

Preuve. Procédons par conditions nécessaires. Toute fonction $F \in \overline{\mathbb{R}}^{\mathbb{R}}$ vérifiant la relation initiale vérifie aussi :

- $\forall s \leq t,\ F(t) \leq F(s)$, ainsi F est décroissante sur $\mathbb{R}$. Elle admet une limite $\lim_{s\to-\infty} F(s) = a \in \overline{\mathbb{R}}$.
- $\forall s \leq t,\ F(t) \leq \inf_{s\leq u\leq t} A(u)$. ainsi $F(t) \leq \inf_{-\infty\leq u\leq t} A(u)$. Et non pouvons donc écrire $F(t) = \inf\Big(\inf_{-\infty\leq u\leq t} A(u), F(s)\Big)$

La dernière expression de F est valable pour tout $s \leq t$. Prenons la limite $s \to -\infty$ et nous obtenons $F(t) = \inf\Big(\inf_{-\infty\leq u\leq t} A(u), a\Big)$.

Dans l'autre sens, pour t fixé :

- soit $F(t) = a$, et alors $\inf_{-\infty \le u \le t} A(u) \ge a$. Pour tout $s \le t$ on a aussi $\inf_{-\infty \le u \le s} A(u) \ge a$ et donc $F(s) = a$. On a alors bien $F(t) = F(s) = \inf\Big(\inf_{s \le u \le t} A(u), F(s)\Big)$
- soit $F(t) < a$. Remarquons que $\forall s \in \mathbb{R}, F(s) \le a$. On peut alors écrire $F(t)$ sous la forme $F(t) = \inf_{-\infty \le u \le t} A(u) = \inf(\inf_{s \le u \le t} A(u), \inf_{-\infty \le u \le s} A(u)) = \inf(\inf_{s \le u \le t} A(u), F(s))$

■

5.2. Théorème d'équivalence

Théorème 5.5 (Équivalence RTC - NC)**.** *Soit* R, C *deux fonctions définies sur* $D = \{(s,t) \in \mathbb{R}^2 \mid s \le t\}$*, positives et vérifiant la relation de Chasles. Soit* $p \in \mathbb{R}$ *un nombre réel quelconque et pour tout* F *définie sur* D*, définissons* $\check{F}(t) = F[p,t)$ *si* $t \ge p$*, et* $= -F[t,p)$ *si* $t < p$*. Soit* $\omega = \inf_{-\infty \le u \le p}\Big[\check{R}(u) - \check{C}(u)\Big]$.

L'ensemble de solutions R', C', B *du système d'équations* (A7),(A8),(A10) *est donné par* $R'[s,t) = \check{R}'(t) - \check{R}'(s)$*,* $s \le t$ *(et de même pour* C'*) et pour tout* $t \in \mathbb{R}$*,*

$$\begin{aligned} \check{R}'(t) &= \check{C}(t) + \inf\Big(\inf_{-\infty \le u \le t}\Big[\check{R}(u) - \check{C}(u) + \sigma\Big], \gamma\Big) \\ \check{C}'(t) &= \check{C}(t) - \check{R}'(t) \\ B(t) &= \check{R}(t) - \check{R}'(t) + \sigma \end{aligned}$$

où $\sigma, \gamma \in \overline{\mathbb{R}}$ *sont des constantes telles que* ($\sigma = -\omega$ *et* $\gamma \ge 0$)*, ou* ($\sigma \ge -\omega$ *et* $\gamma = 0$)*.*

Dans tous les cas on a $B(p) = \sigma$*. Et, si* $B(p) = 0$*, alors pour tout* $t \ge p$ *:*

$$\check{R}'(t) = \inf_{p \le u \le t}\Big[\check{R}(u) + \check{C}(t) - \check{C}(u)\Big]$$

qui est la description d'un Variable Capacity Node avec un point de départ en $t = p$ *(les deux fonctions* $\check{R}$ *et* $\check{C}$ *sont croissantes, égales à 0 en* $t = p$*, et positives sur* $[p, +\infty[$*).*

Preuve. Procédons par conditions nécessaires pour trouver une formule close qui soit solution au système d'équations (A7), (A8), (A10).

Avec les hypothèses sur R et C, d'après le lemme 5.3, R' et C' doivent vérifier la relation de Chasles. *On fixe* $p \in \mathbb{R}$ *un nombre réel quelconque*. D'après la proposition 5.2 et avec la notation $\check{F}(t) = F[p,t)$ si $t \ge p$ et $= -F[p,t)$ si $t < p$, pour toute fonction $F[.,.)$ définie sur $\{(s,t) \in \mathbb{R}^2 \mid s \le t\}$, on sait que R' (resp. C') seront définies de manière unique par les valeurs de $\check{R}'$ (resp. $\check{C}'$). On peut alors réécrire les équations (A7), (A8), (A10) avec les nouvelles inconnues $\check{R}', \check{C}' \in \mathcal{F}_{\nearrow}$ et $B \in \mathcal{F}$. Voici l'ensemble des équations qu'elles doivent vérifier :

Tout d'abord, $\check{R}'(p) = 0$ et $\check{C}'(p) = 0$. Quand toutes les inconnues vérifient la relation de

Chasles, (A7) est équivalente à (A7′) : $\forall t \in \mathbb{R}$, $\check{R}'(t) = \check{C}(t) - \check{C}'(t)$. Comme on a $\check{C}(p) = 0$, $\check{R}'(p) = 0$ implique $\check{C}'(p) = 0$. Et réciproquement.

Avec (A7) et (A8) on a : pour tout $s \le t$,

$$R'[s,t) = C[s,t) - \inf_{s \le u \le t} \Big[C[s,u) - R[s,u) - B(s), 0 \Big]$$

Avec les fonctions à un argument, on peut écrire : pour tout $s \le t$,

$$\check{R}'(t) = \inf_{s \le u \le t} \Big[\check{C}(t) - \check{C}(u) + \check{R}(u) - \check{R}(s) + B(s) + \check{R}'(s), \check{C}(t) - \check{C}(s) + \check{R}'(s) \Big]$$

L'équation (A10) implique en particulier que pour tout $s \ge p$, $B(s) - B(p) = R[p,s) - R'[p,s)$, soit (A10′) : $B(s) - B(p) = \check{R}(s) - \check{R}'(s)$. Quand $s < p$, $B(p) - B(s) = R[s,p) - R'[s,p) = -\check{R}(s) + \check{R}'(s)$. Ainsi (A10′) est vérifiée pour tout $s \in \mathbb{R}$.

L'expression $\check{R}'(t)$ est alors égale à :

$$\check{R}'(t) = \inf_{s \le u \le t} \Big[\check{C}(t) - \check{C}(u) + \check{R}(u) + B(p), \check{C}(t) - \check{C}(s) + \check{R}'(s) \Big]$$

que l'on peut reformuler (A8′) : pour tout $s \le t$,

$$\check{R}'(t) - \check{C}(t) = \inf \Big(\inf_{s \le u \le t} \Big[\check{R}(u) - \check{C}(u) + B(p) \Big], \check{R}'(s) - \check{C}(s) \Big)$$

D'après le lemme 5.4, la fonction $\check{R}'$ est de la forme : pour tout $t \in \mathbb{R}$,

$$\check{R}'(t) - \check{C}(t) = \inf \Big(\inf_{-\infty \le u \le t} \Big[\check{R}(u) - \check{C}(u) + B(p) \Big], \gamma \Big)$$

où $\gamma \in \overline{\mathbb{R}}$ est une constante.

Le nouveau système d'équations (A7′),(A8′),(A10′) et $\check{R}'(p) = 0$ admet comme solutions $\check{R}', \check{C}', B$: quitte à adpater $B(p)$ et γ pour que $\check{R}'(p) = 0$, c'est-à-dire $\inf \Big(\inf_{-\infty \le u \le p} \Big[\check{R}(u) - \check{C}(u) + B(p) \Big], \gamma \Big) = 0$. Notons $\omega = \inf_{-\infty \le u \le p} \Big[\check{R}(u) - \check{C}(u) \Big]$. On a $\omega \le 0$ (prendre $u = p$). Ainsi l'ensemble de solutions pour $B(p)$ et γ est donné par ($B(p) = -\omega$ et $\gamma \ge 0$), ou ($B(p) \ge -\omega$ et $\gamma = 0$).

Autre sens de la preuve du théorème 5.5 En reprenant le raisonnement ci-dessus, on peut montrer le sens contraire : si $\check{R}', \check{C}', B$ vérifient (A7′),(A8′),(A10′) et $\check{R}'(p) = 0$, alors les fonctions associées R, C, B vérifient (A7), (A8), (A10).

Lorsque $t \ge p$, on a :

$$\check{R}'(t) - \check{C}(t) = \inf \Big(\inf_{p \le u \le t} \Big[\check{R}(u) - \check{C}(u) + B(p) \Big], \inf_{-\infty \le u \le p} \Big[\check{R}(u) - \check{C}(u) + B(p) \Big], \gamma \Big)$$

Les deux derniers termes sont nuls (l'avant dernier est égal à $\check{R}'(p)$). Et donc :

$$\check{R}'(t) - \check{C}(t) = \inf\Big(\inf_{p \le u \le t} \Big[\check{R}(u) - \check{C}(u) + B(p) \Big], 0 \Big)$$

Si l'on suppose de plus que $B(p) = 0$, on a pour tout $t \geq p$:

$$\check{R}'(t) - \check{C}(t) = \inf\Big(\inf_{p \le u \le t} \Big[\check{R}(u) - \check{C}(u) \Big], 0 \Big)$$

Lorsque $u = p$, le terme principal $\check{R}(p) - \check{C}(p)$ est égal à 0 par définition, donc on peut simplifier l'expression :

$$\check{R}'(t) - \check{C}(t) = \inf_{p \le u \le t} \Big[\check{R}(u) - \check{C}(u) \Big]$$

et donc pour tout $t \geq p$,

$$\check{R}'(t) = \inf_{p \le u \le t} \Big[\check{R}(u) + \check{C}(t) - \check{C}(u) \Big]$$

Les deux fonctions $\check{R}$ et $\check{C}$ sont croissantes, égales à 0 en $t = p$, et positives sur $[p, +\infty[$. C'est exactement le comportement d'un Variable Capacity Node (avec un point de départ en $t = p$). ■

5.3. Enveloppes en RTC

La modélisation de RTC [Wandeler 2006a] combine des bornes inférieures et supérieures sur le trafic et les services fournis par le système. Considérons un RTC Greedy Processor comme illustré par la figure 5.1. Chaque trafic (ou succession d'évènements) est représenté par une courbe cumulée R qui peut être contrainte par un paire de *courbes d'arrivées RTC* $(\alpha^l, \alpha^u) \in \mathcal{F}_{\nearrow} \times \mathrm{F}cumul$: pour tout $s \leq t$, soit $R[s,t)$ le nombre d'évènements arrivés dans l'intervalle $[s,t[$, alors

$$\alpha^l(t-s) \leq R[s,t) \leq \alpha^u(t-s)$$

De la même manière, les ressources sont représentées par une courbe cumulée C qui peut être contrainte par une paire de *courbes de service RTC* $(\beta^l, \beta^u) \in \mathcal{F}_{\nearrow} \times \mathrm{F}cumul$: pour tout $s \leq t$, soit $R[s,t)$ le nombre de cycles de traitement disponibles dans l'intervalle $[s,t[$, alors

$$\beta^l(t-s) \leq C[s,t) \leq \beta^u(t-s)$$

Si les entrées sont modélisées/enveloppées par ces courbes, on peut propager des contraintes de même type en sortie d'un élément du réseau. Les contraintes en sortie sont données par les formules suivantes, qui proviennent de [Wandeler 2006a]. Avec les

notations de la figure 5.1,

$$\begin{aligned}
\alpha'^u &= \min\Big((\alpha^u * \beta^u) \oslash \beta^l, \beta^u\Big) \\
\alpha'^l &= \min\Big((\alpha^l \oslash \beta^u) * \beta^l, \beta^l\Big) \\
\beta'^u &= (\beta^u - \alpha^l) \overline{\oslash} 0 \\
\beta'^l &= (\beta^l - \alpha^u) \overline{*} 0 = (\beta^l - \alpha^u)_{\nearrow}
\end{aligned}$$

Les preuves dans [Wandeler 2006a, annexe A, pages 201-204] font une autre hypothèse sur le modèle utilisé : l'existence d'un instant p arbitrairement éloigné dans le passé tel que $B(p) = 0$. D'après le théorème 5.5, sur les intervalles $[p, +\infty[$ les RTC Greedy Processors se comportent exactement comme des NC Variable Capacity Nodes. On peut donc retrouver en RTC les résultats connus pour les Variables Capacity Nodes, par exemple la relation avec les courbes de service strict du théorème 6.5.

Corollaire 5.6. *Considérons un RTC Greedy Processor dont les entrées* R *et* C *sont données, et où* C *admet la courbe de service RTC inférieure* β^l. *Considérons une solution* (R, C, R', C', B) *vérifiant les équations* (A7),(A8),(A10). *Supposons que* $\exists p \in \mathbb{R}$ *tel que* $B(p) = 0$. *Alors la trajectoire* $(\check{R}, \check{R}')$ *(où* $\check{F}(.) = F[p, .]$*) admet* β^l *comme courbe de service strict.*

Remarque 5.2 (RTC, ancienne et nouvelle présentations)**.** *Les comparaisons ci-dessus utilisent une version récente du RTC. Dans les tous premiers articles sur RTC, les Greedy Processors étaient définis comme des Variable Capacity Nodes [Thiele 2000, Thiele 2001].*

Conclusion

Dans ce chapitre nous avons étudié l'expressivité du modèle RTC par rapport à l'expressivité du Network Calculus. Le théorème 5.5 montre une équivalence entre RTC Greedy Processor et NC Variable Capacity Node en ce qui concerne les bornes inférieures sur le service. La suite, section 5.3 rappelle d'autres caractéristiques de RTC qui font sa spécificité mais qui peuvent être transposées en Network Calculus.

CHAPITRE 6

COMPARAISON DE DIFFÉRENTS TYPES DE SERVICE

Dans ce chapitre, nous étudions l'expressivité de différentes notions de (courbes de) services. Nous nous intéressons à la hiérarchie allant de la définition très restrictive de *variable capacity node* à la définition la plus générale de *courbe de service simple*. Nous donnons les conditions pour lesquelles ces différentes définitions se recouvrent, et discutons de l'existence d'une description canonique pour les systèmes spécifiés par les différentes définitions.

Remarque 6.1. *Dans ce chapitre, nous étudions uniquement les* variable capacity nodes, *les* services stricts, *les* services faiblement stricts, *et les* services simples.

6.1. Introduction

Il est difficile de dessiner les contours du Network Calculus : un des obstacles majeurs est le nombre de définitions différentes de *courbe de service*, qui correspondent à des modélisations différentes. Le lecteur est souvent informé qu'il est important de s'en tenir à la définition proposée dans chaque article pour assurer la validité des analyses qui y sont faites. Mais la question est rarement posée de savoir si un choix différent de courbes de services aurait abouti au même modèle, ou au moins aux mêmes résultats en termes d'évaluation de performance.

Notre objectif est de faire la distinction entre les modèles correspondant aux différentes notions de courbes de service. Tout d'abord, nous étudions l'expressivité des différents types de courbes de service. De telles comparaisons existent en tant que folklore dans la littérature du Network Calculus, mais elles sont éparpillées, certaines preuves manquent, et certaines comparaisons n'ont jamais été faites. De plus, la question de courbes de services canoniques n'a jamais été étudié : étant donné un système contraint par une famille de courbes d'un certains type, est-il possible de réduire ou de transformer cette famille pour obtenir une courbe canonique ? Est-il possible de la traduire en une famille de courbes de service d'un autre type ? Ces questions clés sont les prémices d'une présentation unifiée du Network Calculus.

Dans la section 6.2, nous donnons une hiérarchie entre ces différentes définitions, qui parfois sont équivalentes, mais parfois pas. Nous donnons une caractérisation des cas d'équivalence et montrons par exemple que l'équivalence entre *variable capacity node* et *service strict*, souvent présentée comme faisant partie du folklore, est souvent vraie mais pas systématiquement. Nous donnons aussi plusieurs résultats concernant la traduction de familles de courbes de service vers des familles du même type ou de types différents. Avec les courbes de service simple, les résultats sont la plupart du temps négatifs.

Complément sur les services stricts

La définition de service strict utilisée dans ce manuscrit (voir page 29) est celle de [Schmitt 2006a, Schmitt 2006b]. Certains articles n'utilisent pas exactement la même définition de service strict [Bouillard 2007a, Bouillard 2008a] : les intervalles de périodes chargées $]s,t]$ sont remplacés par $]s,t[$ (on peut avoir $B(s) = A(s)$ dans les deux cas, mais aussi $B(t) = A(t)$ dans la variante). Soit $\beta \in \mathcal{F}$, on appelle $\mathcal{S}'_{strict}(\beta)$ l'ensemble des trajectoires qui satisfont la définition de [Bouillard 2007a].

Quelles différences y a-t-il entre ces deux définitions très proches ? Clairement, on a $\forall \beta \in \mathcal{F}$, $\mathcal{S}'_{strict}(\beta) \subseteq \mathcal{S}_{strict}(\beta)$. Si β est continue à gauche, étant donné que toutes les fonctions cumulées sont supposées continues à gauche, on a l'égalité $\mathcal{S}'_{strict}(\beta) = \mathcal{S}_{strict}(\beta)$: soit $(A,B) \in \mathcal{S}_{strict}(\beta)$, si $]s,t[$ est une période chargée, alors $\forall s < t' < t$, $]s,t']$ est une période chargée, ainsi $B(t') - B(s) \geq \beta(t'-s)$ et $B(t) - B(s) \geq \beta(t-s)$ lorsque t' tend vers t. Si β n'est pas continue à gauche, l'inclusion peut être stricte $\mathcal{S}'_{strict}(\beta) \subsetneq \mathcal{S}_{strict}(\beta)$.

On pourra étudier l'exemple de la figure 6.1 où $A(t) = 1/2$ si $t > 0$ et $= 0$ pour $t = 0$, $B(t) = \min(t/2, 1/2)$ et $\beta(t) = \lfloor t \rfloor$. Il existe une unique période chargée maximale : $]0,1[$. On a $(A,B) \in \mathcal{S}_{strict}(\beta)$ (car $\forall]s,t]$, $0 \leq s \leq t < 1$, $B(t) - B(s) \geq \beta(t-s) = 0$), mais $(A,B) \notin \mathcal{S}'_{strict}(\beta)$ (en effet $B(1) - B(0) = 1/2 \not\geq \beta(1-0) = 1$). Notons d'ailleurs que $(A,B) \in \mathcal{S}_{vcn}(\beta)$ (par exemple, on peut prendre $C(t) = B(t)$ pour $t < 1$ et $= t$ pour $t \geq 1$). Ceci n'est pas étonnant : on montrera au théorème 6.5 que pour cette courbe β on a $\mathcal{S}_{strict}(\beta) = \mathcal{S}_{vcn}(\beta)$.

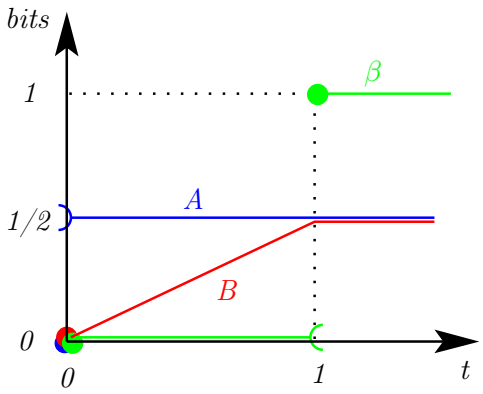


FIGURE 6.1 : Attention à la définition de service strict

6.2. Étude de courbes de service

6.2.1. Monotonie

Toutes les définitions de courbes de service considérées partagent le même comportement monotone vis-à-vis des trajectoires.

Proposition 6.1 (Monotonie). *Pour tous les types $\mathcal{T}$ de courbes de service étudiées, pour tout $\beta, \beta' \in \mathcal{F}$ (pas nécessairement dans $\mathcal{F}_{\nearrow}$), si $\beta \leq \beta'$ alors $\mathcal{S}_{\mathcal{T}}(\beta) \supseteq \mathcal{S}_{\mathcal{T}}(\beta')$.*

De plus, pour les *variable capacity nodes*, les courbes de service strict, les courbes de service faiblement strict, on peut remplacer les courbes de service par leur clôture respective.

Proposition 6.2. *Soit $\beta \in \mathcal{F}$, on a alors*

- $\mathcal{S}_{wstrict}(\beta) = \mathcal{S}_{wstrict}(\beta_{\uparrow})$,
- $\mathcal{S}_{strict}(\beta) = \mathcal{S}_{strict}(\beta_{\uparrow})$, $\mathcal{S}_{strict}(\beta) = \mathcal{S}_{strict}(\beta^{\overline{*}})$.
- $\mathcal{S}_{vcn}(\beta) = \mathcal{S}_{vcn}(\beta_{\uparrow})$ *et* $\mathcal{S}_{vcn}(\beta) = \mathcal{S}_{vcn}(\beta^{\overline{*}})$.

Les résultats précédents sont classiques en Network Calculus. Le théorème suivant est nouveau.

Théorème 6.3 (Monotonie raffinée). *Soit $\beta, \beta' \in \mathcal{F}$,*

1. $\mathcal{S}_{simple}(\beta) \supseteq \mathcal{S}_{simple}(\beta') \Leftarrow \beta \leq \beta'$.
2. $\mathcal{S}_{simple}(\beta) \supseteq \mathcal{S}_{simple}(\beta') \not\Rightarrow \beta \leq \beta'$.
3. $\mathcal{S}_{simple}(\beta) \supseteq \mathcal{S}_{simple}(\beta') \Rightarrow \beta_{\uparrow} \leq \beta'_{\uparrow}$.
4. $\mathcal{S}_{simple}(\beta) \supseteq \mathcal{S}_{simple}(\beta') \not\Leftarrow \beta_{\uparrow} \leq \beta'_{\uparrow}$.
5. $\mathcal{S}_{simple}(\beta_{\uparrow}) \supseteq \mathcal{S}_{simple}(\beta'_{\uparrow})$ *si et seulement si* $\beta_{\uparrow} \leq \beta'_{\uparrow}$.
6. $\mathcal{S}_{wstrict}(\beta) \supseteq \mathcal{S}_{wstrict}(\beta')$ *si et seulement si* $\beta_{\uparrow} \leq \beta'_{\uparrow}$.
7. $\mathcal{S}_{strict}(\beta) \supseteq \mathcal{S}_{strict}(\beta')$ *si et seulement si* $(\beta_{\uparrow})^{\overline{*}} \leq (\beta'_{\uparrow})^{\overline{*}}$.

8. $\mathcal{S}_{vcn}(\beta) \supseteq \mathcal{S}_{vcn}(\beta')$ *si et seulement si* $(\beta_\uparrow)^{\overline{*}} \leq (\beta'_\uparrow)^{\overline{*}}$.

Preuve.

1. D'après la proposition 6.1.
2. Soit $\beta'(t) = 0$ si $t = 0$ ou $t \in]1,2]$ et $\beta'(t) = +\infty$ sinon et $\beta(t) = 0$ si $t \in [0,1]$ ou $t \in]2,+\infty[$ et $\beta(t) = +\infty$ sinon. Alors, $\beta \not\leq \beta'$ mais, $\forall A \in \mathcal{F}_\uparrow$, $(A * \beta')_\uparrow(t) = A(t)$ si $t \in [0,1]$, $= A(1)$ si $t \in]1,2]$ et $= \max(A(1), A(t-2))$ sinon, alors que $(A * \beta)_\uparrow(t) = A(0) = 0$ si $t \in [0,1]$, $= A(t-1)$ si $t \in]1,2]$ et $= A(1)$ sinon. Alors, $(A * \beta')_\uparrow \geq (A * \beta)_\uparrow$ et $\mathcal{S}_{simple}(\beta) \supseteq \mathcal{S}_{simple}(\beta')$. La figure 6.2 illustre cette construction.

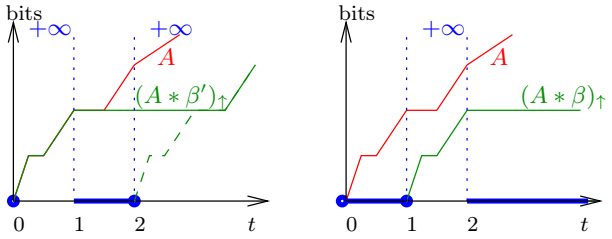


FIGURE 6.2 : $\mathcal{S}_{simple}(\beta) \supseteq \mathcal{S}_{simple}(\beta') \not\Rightarrow \beta \leq \beta'$.

3. $(\delta_0, \beta'_\uparrow) \in \mathcal{S}_{simple}(\beta')$, d'où $(\delta_0, \beta'_\uparrow) \in \mathcal{S}_{simple}(\beta)$ et donc $\beta'_\uparrow \geq (\delta_0 * \beta)_\uparrow = \beta_\uparrow$.
4. Prenons $\beta'(t) = 0$ si $t = 0$ ou $t \in]1,+\infty[$ et $\beta'(t) = +\infty$ sinon, et $\beta = \beta'_\uparrow = \delta_0$. Nous avons $\beta_\uparrow \leq \beta'_\uparrow$ et $\mathcal{S}_{simple}(\beta) = \{(A,A) \mid A \in \mathcal{F}_\uparrow\}$, mais $\forall A \in \mathcal{F}_\uparrow$, $(A, A(\min(.,1)) \in \mathcal{S}_{simple}(\beta')$. La figure 6.3 illustre cette construction.

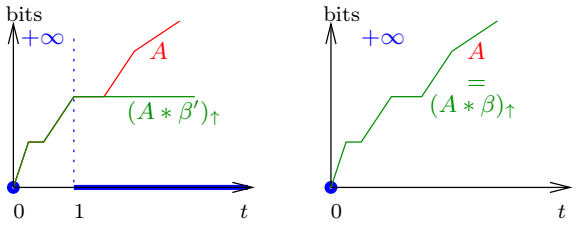


FIGURE 6.3 : $\mathcal{S}_{simple}(\beta) \supseteq \mathcal{S}_{simple}(\beta') \not\Leftarrow \beta_\uparrow \leq \beta'_\uparrow$.

5. $\Rightarrow$: comme en (3)
   $\Leftarrow$: d'après la proposition 6.1.
6. - 8. D'après la proposition 6.2, et :
   $\Rightarrow$: comme en (3)
   $\Leftarrow$: d'après la proposition 6.1.

■

6.2.2. Familles de courbes de service

Nous nous demandons maintenant si un système peut admettre plusieurs courbes de service d'un type donné, ou si elle est unique.

Soit $(\beta_i)_{i\in I}$ un famille (potentiellement infinie) de fonctions de $\mathcal{F}$, et $\mathcal{T}$ un type de courbes de service particulier.

Le fait que le système $\mathcal{S}$ admette toutes les fonctions β_i comme courbes de service de type $\mathcal{T}$ peut aussi s'écrire $\mathcal{S} \subseteq \bigcap_{i\in I} \mathcal{S}_{\mathcal{T}}(\beta_i)$.

Théorème 6.4 (Familles de courbes). *Soit* I *et* J *deux ensembles finis, et* $(\beta_i)_{i\in I}$ *et* $(\beta'_j)_{j\in J}$ *deux familles de* $\mathcal{F}_\uparrow$. *Alors,*

1. $\bigcap_{i\in I} \mathcal{S}_{simple}(\beta_i) = \bigcap_{j\in J} \mathcal{S}_{simple}(\beta'_j)$ *ssi* $\{\beta \in \mathcal{F} \mid \exists i \in I,\ \beta \leq \beta_i\} = \{\beta \in \mathcal{F} \mid \exists j \in J,\ \beta \leq \beta'_j\}$.
2. $\bigcap_{i\in I} \mathcal{S}_{wstrict}(\beta_i) = \mathcal{S}_{wstrict}((\sup_{i\in I} \beta_i)_\uparrow)$.
3. $\bigcap_{i\in I} \mathcal{S}_{strict}(\beta_i) = \mathcal{S}_{strict}((\sup_{i\in I} \beta_i)^{\overline{*}})$.
4. $\bigcap_{i\in I} \mathcal{S}_{vcn}(\beta_i) = \mathcal{S}_{vcn}((\sup_{i\in I} \beta_i)^{\overline{*}})$.

Preuve.

1. Tout d'abord, soit $I = \{0,\dots,k\}$ et considérons $\cap_{i\in I}\mathcal{S}_{simple}(\beta_i)$ tel que les fonctions β_i ne sont pas comparables deux à deux. Alors, il existe $t_1,\dots,t_k$ tel que $\forall i \in I \setminus \{0\}$, $\beta_0(t_i) > \beta_i(t_i)$ et on peut supposer sans perte de généralité que $0 < t_1 \leq \cdots \leq t_k$. Définissons
$$A(t) = \begin{cases} 0 & \text{si } t = 0, \\ \beta_0(t_k) - \beta_0(t_i) & \text{si } t_i \leq t_k - t < t_{i+1}, \\ +\infty & \text{si } t > t_{k-1}. \end{cases}$$
Ainsi, pour tout $i \in I$, $A * \beta_i(t_k) = \inf_j \beta_i(t_j) + A(t_k - t_j) = \inf_j \beta_i(t_j) + \beta_0(t_k) - \beta_0(t_j)$. Alors, $A * \beta_0(t_k) = \beta_0(t_k)$ et $\forall i \in \{1,\dots,k\}$, $A * \beta_i(t_k) \leq \beta_i(t_i) + \beta_0(t_k) - \beta_0(t_i) < \beta_0(t_k)$. Donc on a $\cap_{i\in I}\mathcal{S}_{simple}(\beta_i) \subsetneq \cap_{i\in I/\{0\}}\mathcal{S}_{simple}(\beta_i)$. La figure 6.4 illustre ceci.

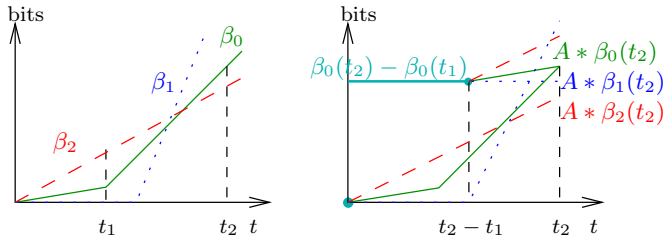


FIGURE 6.4 : Non-inclusion de familles de courbes

Considérons $\cap_{i\in I}\mathcal{S}_{simple}(\beta_i) = \cap_{j\in J}\mathcal{S}_{simple}(\beta'_j)$ où les β_i ne sont pas comparables deux à deux et les β'_j non plus. Nous avons pour tout $j \in J$, $\cap_{i\in I}\mathcal{S}_{simple}(\beta_i) \bigcap \mathcal{S}_{simple}(\beta'_j) = \cap_{i\in I}\mathcal{S}_{simple}(\beta_i)$. Alors, par réciproque du paragraphe précédent, $\exists i \in I$, tel que $\beta'_j \leq \beta_i$. Par symétrie, $\forall i \in I$, $\exists j$ tel que $\beta_i \leq \beta'_j$. Comme les fonctions dans I et J ne sont pas comparables deux à deux, cela signifie que $\forall i \in I$, $\exists j \in J$ tel que $\beta_i = \beta'_j$, et symétriquement aussi.

2. $\forall i \in I$, $\mathcal{S}_{wstrict}(\beta_i) \supseteq \mathcal{S}_{wstrict}(\sup_{i\in I} \beta_i)$, ainsi $\bigcap_{i\in I} \mathcal{S}_{wstrict}(\beta_i) \supseteq \mathcal{S}_{wstrict}(\sup_{i\in I} \beta_i)$.

Soit $(A,B) \in \bigcap_{i\in I} \mathcal{S}_{wstrict}(\beta_i)$. Alors, $\forall t \in \mathbb{R}_+$, $\forall i \in I$, $B(t) \geq A(start(t)) + \beta_i(t - start(t))$, d'où A et B vérifient $B(t) \geq A(start(t)) + \sup_{i\in I} \beta_i(t - start(t))$ et on a bien $(A,B) \in \mathcal{S}_{wstrict}(\sup_{i\in I} \beta_i) = \mathcal{S}_{wstrict}((\sup_{i\in I} \beta_i)\uparrow)$.

3. Comme en (2), sauf : Soit $(A,B) \in \bigcap_{i\in I} \mathcal{S}_{strict}(\beta_i)$. Alors, $\forall s < t \in \mathbb{R}_+$, $\forall i \in I$, $B(t) \geq B(s) + \beta_i(t-s)$, d'où $B(t) \geq B(s) + \sup_{i\in I} \beta_i(t-s)$ et $(A,B) \in \mathcal{S}_{wstrict}(\sup_{i\in I} \beta_i) = \mathcal{S}_{wstrict}((\sup_{i\in I} \beta_i)^{\bar{*}})$.
4. Comme en (3), en remplaçant B par C.

■

6.2.3. Hiérarchie

La hiérarchie suivante, entre différentes notions de courbes de service, est souvent considérée comme du folklore, mais les cas d'égalité n'ont pas été étudiés (par exemple $\mathcal{S}_{vcn}(\beta) = \mathcal{S}_{strict}(\beta)$ est énoncé sans hypothèse sur β dans [Le Boudec 2001] et [Wandeler 2006a]).

Théorème 6.5 (Hiérarchie). *Pour tout $\beta \in \mathcal{F}$, nous avons les inclusions suivantes :*

$$\mathcal{S}_{vcn}(\beta) \subseteq \mathcal{S}_{strict}(\beta) \subseteq \mathcal{S}_{wstrict}(\beta) \subseteq \mathcal{S}_{simple}(\beta).$$

Les égalités nécessitent :

- $\mathcal{S}_{vcn}(\beta) = \mathcal{S}_{strict}(\beta)$ *ssi* $\beta \oslash \beta$ *n'a pas de valeurs infinies.*
- $\mathcal{S}_{strict}(\beta) = \mathcal{S}_{wstrict}(\beta)$ *ssi* $\beta\uparrow = \delta_T$, $T \in \mathbb{R}_+ \cup \{+\infty\}$.
- $\mathcal{S}_{wstrict}(\beta) = \mathcal{S}_{simple}(\beta)$ *ssi* $\beta\uparrow = \delta_0$ *ou* 0.

Preuve. Nous supposons que $\beta(0) \leq 0$, sans quoi tous les ensembles seraient vides.

L'inclusion $\mathcal{S}_{strict}(\beta) \subseteq \mathcal{S}_{wstrict}(\beta)$ est vraie car pour tout $t \in \mathbb{R}_+$, soit $]start(t), t]$ est une période chargée, et alors $B(t) - B(start(t)) \geq \beta(t - start(t))$, soit $]start(t), t[$ est une période chargée et $B(t) = A(t)$ mais dans ce cas $start(t) = t$ et $B(t) - B(start(t)) = 0 \geq \beta(t - start(t)) = 0$.

L'inclusion $\mathcal{S}_{wstrict}(\beta) \subseteq \mathcal{S}_{simple}(\beta)$ vient du fait que, si $(A,B) \in \mathcal{S}_{wstrict}(\beta)$, alors $B(t) \geq B(start(t)) + \beta(t - start(t)) = A(start(t)) + \beta(t - start(t)) \geq \inf_{0\leq s\leq t}(A(s) + \beta(t-s))$.

Montrons que $\mathcal{S}_{vcn}(\beta) \subseteq \mathcal{S}_{strict}(\beta)$. Soit $(A,B) \in \mathcal{S}_{vcn}(\beta)$, il existe $C \in \mathcal{F}_\nearrow$ tel que $\forall t \geq 0$, $B(t) = \inf_{0\leq s\leq t}\left[A(s) + C(t) - C(s)\right]$ et $\forall 0 \leq s \leq t$, $C(t) - C(s) \geq \beta(t-s)$. Considérons la période chargée $]s,t]$ pour (A,B), alors par définition $start(s) = start(t) = p$. D'après le lemme 5.1, on a $B(t) = B(p) + C(t) - C(p)$ et $B(s) = B(p) + C(s) - C(p)$. Ainsi $B(t) - B(s) = C(t) - C(s) \geq \beta(t-s)$ et nous avons montré que β est une courbe de service strict pour (A,B).

Condition d'égalité pour $\mathcal{S}_{vcn}(\beta) = \mathcal{S}_{strict}(\beta)$: $\beta \oslash \beta$ n'a pas de valeurs infinies

Comme $\mathcal{S}_{vcn}(\beta) = \mathcal{S}_{vcn}(\beta^{\bar{*}})$ et $\mathcal{S}_{strict}(\beta) = \mathcal{S}_{strict}(\beta^{\bar{*}})$, on peut considérer que β est sur-additive. On a alors $\beta(t-s) \leq \beta\oslash\beta(t) - \beta\oslash\beta(s)$ et $\sum_{i=1}^{n} \beta\oslash\beta(t_i) + \beta(t_0) \geq \beta(\sum_{i=0}^{n} t_i)$. Supposons que $\beta \oslash \beta < \infty$. Soit $(A,B) \in \mathcal{S}_{strict}(\beta)$: cette trajectoire peut être vue comme une succession

de périodes d'« inactivité » (intervalles de longueur ≥ 0 pendant lesquels la charge est nulle) et de périodes chargées de longueur non nulle. Soit $(t_i)_{i\geq 0}$ la suite croissante des débuts de ces périodes (t_{2i} sont les débuts des périodes d'inactivité et les t_{2i+1} sont les débuts des périodes chargées). Soit $\ell_i = t_{i+1} - t_i$ et

$$C(t) = \begin{cases} A(t_i) + \beta \oslash \beta(t_i) + \sum_{j=0}^{i} \beta \oslash \beta(\ell_j) & \text{si } t = t_i \\ A(t) + \beta \oslash \beta(t) + \sum_{j=0}^{2i} \beta \oslash \beta(\ell_j) & \text{si } t_{2i} < t < t_{2i+1} \\ B(t) - B(t_{2i+1}) - C(t_{2i+1}) & \text{si } t_{2i-1} < t < t_{2i} \end{cases}$$

Alors, $\forall s < t$,

- si $t_{2i-1} \leq s < t < t_{2i}$, alors $C(t) - C(s) = B(t) - B(s) \geq \beta(t-s)$;
- si $t_{2i} \leq s < t < t_{2i+1}$ alors $C(t) - C(s) = \beta \oslash \beta(t) - \beta \oslash \beta(s) + A(t) - A(s) \geq \beta(t-s)$;
- si $t_j \leq s < t_{j+1} \leq t_{2i+1} \leq t \leq t_{2i+2}$, alors $C(t) - C(s) = C(t) - C(t_{2i+1}) + C(t_{2i+1}) - C(s) \geq \beta(t - t_{2i+1}) + \sum_{k=j+1}^{2i+1} \beta \oslash \beta(\ell_j) \geq \beta(t - t_{2i+1} + t_{2i+1} - t_j) \geq \beta(t-s)$;
- si $t_j \leq s < t_{j+1} \leq t_{2i} \leq t \leq t_{2i+1}$, alors $C(t) - C(s) \geq C(t) - C(t_{2i}) + C(t_{2i}) - C(s) \geq \beta(t - t_{2i}) + \sum_{k=j+1}^{2i} \beta \oslash \beta(\ell_j) \geq \beta(t - t_{2i} + t_{2i} - t_j) \geq \beta(t-s)$;

De plus, $B(t) = A(start(t)) + C(t) - C(start(t))$, $\forall s \in [start(t), t]$, $B(t) \leq A(s) + C(t) - C(s)$, et $\forall s < start(t)$, $C(s) - A(s) \geq C(start(t)) - A(start(t))$ d'où $B(t) \leq A(s) + C(t) - C(s)$. Ainsi, $(A, B) \in \mathcal{S}_{vcn}(\beta)$.

D'autre part, si $\beta \oslash \beta(t_0) = \infty$, prenons $(\delta_{t_0}, \beta * \delta_{t_0}) \in \mathcal{S}_{strict}(\beta)$. Pour que le calcul soit possible, $C(t_0)$ doit être fini. Mais, il faut $C(t + t_0) - C(t_0) = \beta(t)$ et $C(t + t_0) - C(0) \geq \beta(t + t_0)$. Ainsi, $\forall t \in \mathbb{R}_+$, $C(t_0) = C(t_0) - C(0) \geq \beta(t + t_0) - \beta(t)$ et $C(t_0) \geq \beta \oslash \beta(t_0) = +\infty$.

Condition d'égalité pour $\mathcal{S}_{strict}(\beta) = \mathcal{S}_{wstrict}(\beta)$: $\beta_\uparrow = \delta_T$, $T \in \mathbb{R}_+ \cup \{+\infty\}$

Si $\beta_\uparrow = \delta_T$, $T \in \mathbb{R}_+ \cup \{+\infty\}$, alors soit $(A, B) \in \mathcal{S}_{wstrict}(\beta)$. Soit $s < t$ dans la même période chargée. Alors, $t - s < T$ et $B(t) \geq B(s) = B(s) + \beta_\uparrow(t-s)$. Ainsi $(A, B) \in \mathcal{S}_{strict}(\beta)$.

Sinon $\beta_\uparrow$ n'est pas un délai : soit $A = \delta_0$. Il existe $s > 0$ tel que $0 < \beta_\uparrow(s) < \infty$. soit $t_0 = \sup\{t \geq 0 \mid \beta_\uparrow(t) = 0\} \leq s$. Définissons $B(t) = \beta_\uparrow(t)$ si $t < (s - t_0)/2$ ou $t > s$, $= \beta_\uparrow(s)$ si $t \in [(s - t_0)/2, s]$. Nous avons $(A, B) \in \mathcal{S}_{wstrict}(\beta) \setminus \mathcal{S}_{strict}(\beta)$. Et effectivement, $B(s) - B((s - t_0)/2) = 0 < \beta_\uparrow((s + t_0)/2)$.

Condition d'égalité pour $\mathcal{S}_{wstrict}(\beta) = \mathcal{S}_{simple}(\beta)$: $\beta_\uparrow = \delta_0$ ou 0

$\mathcal{S}_{simple}(0) = \mathcal{S}_{wstrict}(0) = \{(A, B) \in \mathcal{F}_\uparrow^2 \mid A \geq B\}$; $\mathcal{S}_{simple}(\delta_0) = \mathcal{S}_{wstrict}(\delta_0) = \{(A, A) \mid A \in \mathcal{F}_\uparrow^2\}$. Si $\beta \notin \{0, \delta_0\}$, nous considérons trois cas : soit $t_0 = \inf\{t \geq 0 \mid \beta(t) > 0\}$. Si $t_0 > 0$, prenons s tel que $\beta(s) > 0$ et $A(t) = (\beta(s)/s)\, t$. Comme $\forall t > 0$, $A * \beta(t) \leq A(t - t_0) < A(t)$, la trajectoire $(A, A * \beta) \in \mathcal{S}_{simple}(\beta)$ a une seule période chargée infinie, mais $A(0) + \beta(s) = A(s)$, donc la première période chargée d'un serveur offrant un service faiblement strict doit être de longueur au plus s : $(A, A * \beta) \notin \mathcal{S}_{wstrict}(\beta)$.

Si $t_0 = 0$, s'il existe ρ tel que $A(t) = \rho t$ et $s > 0$ avec $\forall t \in [0, s]$, $A(t) > \beta(t)$ et $\beta^{-1}(A(s)) > s$. Soit B défini par $B(t) = \beta(t)$ si $t < s$, $= A(s)$ si $t = s$, $\max(\beta(t), A(s))$ sinon. Nous avons $A \geq B \geq \beta \geq A * \beta$, donc $(A, B) \in \mathcal{S}_{simple}(\beta)$, mais il existe une période chargée débutant en s, et rien n'est servi entre s et $\beta^{-1}(A(s))$. Donc $(A, B) \notin \mathcal{S}_{wstrict}(\beta)$.

Sinon, $t_0 = 0$ et il existe $\rho, u \in \mathbb{R}_+$, $A(t) = \rho t$ tel que $A \not\geq \beta$ mais $\forall t \in]0, u[$, $A(t) > \beta(t)$. Ainsi, $A * \beta \leq \min(A, \beta) < A$, mais $(A, B) \in \mathcal{S}_{wstrict}(\beta) \Rightarrow B = A$. Donc $(A, A * \beta) \notin \mathcal{S}_{wstrict}(\beta)$. ■

Des cas d'égalité

Une famille importante de courbes de service pour lesquelles l'égalité n'est pas vraie est la famille des délais purs : $\delta_T, T \in \mathbb{R}_+$.

Les autres cas d'égalité semblent très restrictifs. Néanmoins, si on considère l'égalité des processus de sortie quand le service est exact (c'est-à-dire quand la dernière égalité est remplacée par une égalité dans la définition de $\mathcal{S}_{\mathcal{T}}(\beta)$, $\mathcal{T} \in \{simple, wstrict, strict\}$) pour tout processus d'arrivée, les cas d'égalité sont plus fréquents : pour services simple et faiblement strict, l'égalité est vraie pour tout $\lambda_r, r \in \mathbb{R}_+ \cup \{+\infty\}$; et pour service strict et faiblement strict, l'égalité est vérifiée pour toute fonction suradditive.

Impossibilité de traduction

Enfin, le théorème suivant énonce le fait que dans le cas où il n'y a pas d'égalité, il n'est pas possible d'exprimer un type de courbe de service comme combinaison de courbes de service d'un autre type.

Théorème 6.6 (Pas de traduction entre les familles)**.** *Soit $\mathcal{T}$ et $\mathcal{T}'$ deux types différents de courbes de service parmi* vcn, simple, strict (weak), strict. *Alors*

$$\nexists (\beta_i)_{i\in I} \in \mathcal{F}^I,\ (\beta'_j)_{j\in J} \in \mathcal{F}^J,\ \bigcap_{i\in I} \mathcal{S}_{\mathcal{T}}(\beta_i) = \bigcap_{j\in J} \mathcal{S}_{\mathcal{T}'}(\beta'_j),$$

sauf dans les cas d'égalité définis dans le théorème 6.5.

Preuve. Traduction entre vcn et service strict, et entre service faible strict et service strict : il suffit d'étudier le cas où I et J sont des singletons, d'après le théorème 6.4.

Pour $(\mathcal{T}, \mathcal{T}') = \{(strict, vcn), (wstrict, strict)\}$, supposons que $\mathcal{S}_{\mathcal{T}}(\beta) = \mathcal{S}_{\mathcal{T}}(\beta')$. Alors, d'après le théorème 6.5, $\mathcal{S}_{\mathcal{T}'}(\beta) \subseteq \mathcal{S}_{\mathcal{T}'}(\beta')$ et d'après le théorème 6.3, $\beta' \leq \beta$. Ainsi, $(\delta_0, \beta') \in \mathcal{S}_{\mathcal{T}'}(\beta') = \mathcal{S}_{\mathcal{T}}(\beta)$, donc $\beta' \geq \beta$. On a nécessairement $\beta = \beta'$.

Traduction entre service simple et service faiblement strict : considérons $(\beta_i)_{i\in I}$ et β', et supposons que $\cap_{i\in I} \mathcal{S}_{simple}(\beta_i) = \mathcal{S}_{wstrict}(\beta')$. D'après la proposition 6.1 et le théorème 6.5, on a : $\mathcal{S}_{wstrict}(\sup_{i\in I} \beta_i) \subseteq \mathcal{S}_{simple}(\sup_{i\in I} \beta_i) \subseteq \cap_{i\in I} \mathcal{S}_{simple}(\beta_i) = \mathcal{S}_{wstrict}(\beta')$. Donc, $\beta' \leq (\sup_{i\in I} \beta_i)_\uparrow$. D'autre part, $(\delta_0, \beta') \in \mathcal{S}_{wstrict}(\beta') = \cap_{i\in I} \mathcal{S}_{simple}(\beta_i)$, donc $\beta' \geq \sup_{i\in I}(\delta_0 * \beta_i)_\uparrow = (\sup_{i\in I} \beta_i)_\uparrow$ et $\beta' = (\sup_{i\in I} \beta_i)_\uparrow$. De plus, $\mathcal{S}_{wstrict}((\sup_{i\in I} \beta_i)_\uparrow) = \mathcal{S}_{wstrict}(\sup_{i\in I} \beta_i) \subseteq$

$\mathcal{S}_{simple}(\sup_{i\in I}\beta_i) \subseteq \cap_{i\in I}\mathcal{S}_{simple}(\beta_i) = \mathcal{S}_{strict}(\beta')$. Ainsi, toutes les inégalités sont en fait des égalités et on a nécessairement $\mathcal{S}_{simple}(\sup_{i\in I}\beta_i) = \cap_{i\in I}\mathcal{S}_{simple}(\beta_i)$ et $\beta' = \sup_{i\in I}\beta_i = \delta_0$ ou 0. ■

6.3. Conclusion

Nous espérons que cette étude apportera une nouvelle lumière sur les principales définitions de courbe de service utilisées dans les modèles à base d'enveloppes.

Nous avons montré qu'il existe des fossés dans la hiérarchie, comme l'impossibilité d'exprimer un service strict comme une famille de service simples, même en utilisant des familles infinies.

Trouver un cadre unificateur pour l'analyse exacte de performances pire-cas semble donc difficile. De fait, si certaines analyses exactes utilisant des services simples s'appuient exclusivement sur l'algèbre $(\min, +)$ (Window Flow Control, lissage de trafic multimédia [Chang 2000, Le Boudec 2001]), d'autres analyses exactes de courbes strictes n'utilisent pas du tout $(\min, +)$ (performances pire-cas de certains réseaux acycliques dans [Bouillard 2010]).

Néanmoins, toutes les politiques de services n'ont pas été comparées.

On peut étudier la possibilité de « mélanger » différentes définitions, comme par exemple avec les *adaptative service curves* (mélangeant service strict et service simple) dans [Le Boudec 2001], ou essayer d'introduire des définitions intermédiaire entre les définitions classiques exposées dans ce chapitre.

Il faut aussi prendre en compte les courbes de service maximum, comme *les courbes de service simple maximum* définies en Network Calculus [Le Boudec 2001, p. 42-48] ou les *courbes de service supérieures* énormément utilisées en Real-Time Calculus ([Wandeler 2006a]).

Leur utilisation peut en effet introduire de nouvelles relations entre les différentes notions. Par exemple, un serveur à débit constant peut être vu comme un serveur avec des courbes de service simple minimum et maximum toutes deux égales à $\lambda_r(t) = rt$. Dans ce cas, le serveur admet λ_r comme courbe de service strict. Alors qu'un système défini exclusivement par un service minimum peut seulement assurer un service strict λ_0 ou δ_0.

Il existe d'autres aspects aussi important que l'expressivité, comme la robustesse des définitions vis-à-vis du multiplexage de flux. Par exemple un serveur avec priorités fixes offrant un service minimum simple ne peut garantir aucun service simple non nul au flux de priorité la plus basse.

CHAPITRE 7

NOTION DE SERVICE INTERMÉDIAIRE ET SERVICE RÉSIDUEL

Dans ce chapitre nous étudions la stabilité de la notion de service strict pour deux configurations :

- la mise en tandem de serveurs. Le résultat étant négatif, nous nous demandons s'il peut exister une notion de service intermédiaire (entre simple et strict), qui soit préservée par la concaténation ;
- le service résiduel obtenu par un flux agrégé avec une politique de service de type priorité stricte.

7.1. Concaténation de serveurs : convolution

7.1.1. Résultats de stabilité/instabilité

Le théorème suivant montre que la notion de service strict n'est pas préservée par concaténation (deux serveurs en tandem).

Théorème 7.1. *Soit un système* S *constitué de deux serveurs en tandem, de courbes de service respectives* β_1 *pour* A *et* β_2 *pour* B. *Alors*

1. $\beta_1 * \beta_2$ *est une courbe de service pour* S.
2. *Si* A *et* B *fournissent des services stricts de courbes* β_1 *et* β_2, *le service fourni par* S *n'est pas forcément un service strict de courbe* $\beta_1 * \beta_2$.

Preuve. La première partie du thèorème est une propriété classique sur la concaténation de serveurs, que l'on peut trouver dans [Le Boudec 2001, page 28] ou dans [Chang 2000].

Ci-dessous un contre-exemple pour illustrer le fait que la concaténation des deux serveurs ne fournit pas nécessairement un service *strict* de courbe $\beta_1 * \beta_2$.

D'après la définition de service strict (p. 29), un noeud admet δ_T comme courbe de service strict si et seulement si la durée de chaque période chargée est au plus T. Considérons le système de la figure 7.1 composé de deux noeuds en tandem. Supposons que le noeud 1 (resp. le noeud 2) fonctionne comme le serveur suivant : dès qu'une quantité non nulle de données arrive, il attend un temps $T_1 = 2$ (resp. $T_2 = 3$) alors que sa file d'attente se remplit, puis il sert instantanément toutes les données présentes dans la file d'attente. Le noeud 1 (resp. le noeud 2) a une courbe de service strict δ_{T_1} (resp. δ_{T_2}). La figure 7.1 montre ce qui arrive lorsqu'un flux parcourt le noeud 1 et le noeud 2 avec une courbe d'arrivée $A(t)$. La durée de la période chargée pour ce système est 9, et donc $\delta_{T_1} * \delta_{T_2} = \delta_{T_1+T_2}$ n'est pas une courbe de service strict car $T_1 + T_2 = 5 < 9$.

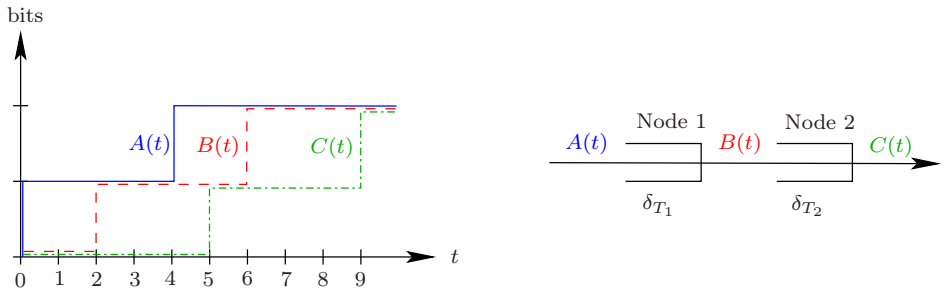


Figure 7.1 : Tandem : service strict * service strict ≠ service strict.

■

7.1.2. Existence d'une notion intermédiaire pour les courbes de service ?

Notre motivation première était d'essayer de définir une notion intermédiaire de courbe de service qui en particulier serait préservée par la concaténation.

Nous allons montrer que ce n'est pas possible : le théorème suivant montre que si l'on impose la stabilité de cette définition par concaténation, la notion intermédiaire de courbe de service doit coïncider avec celle de courbe de service simple. Comme nous le verrons

plus tard, définir un service résiduel n'est pas possible avec des courbes de service simple. Et donc, on ne pourra pas avoir de courbe de service qui, en même temps, soit stable par concaténation, et permette de définir un service résiduel.

Rappelons les propriétés que notre courbe de service intermédiaire doit satisfaire :

1. Pour toute trajectoire du système, β courbe de service strict pour la trajectoire $\Rightarrow$ β courbe de service intermédiaire pour la trajectoire $\Rightarrow$ β courbe de service simple pour la trajectoire [1] ;
2. Si β_1 et β_2 sont respectivement les courbes de service intermédiaires de deux serveurs S_1 et S_2 en tandem, alors $\beta_1 * \beta_2$ est une courbe de service intermédiaire pour la concaténation des deux serveurs.

Théorème 7.2. *Aucune notion de courbe de service intermédiaire, comme définie ci-dessus, ne peut être plus restrictive que la notion de courbe de service simple.*

Preuve. L'idée est de décomposer un serveur de courbe de service (strict) β en une suite de serveurs de courbes de service (β_i) tel que $\beta = \bigstar \beta_i$, et de comparer les processus de départ quand les serveurs sont utilisés comme des serveurs stricts, ou comme des serveurs simples. Nous verrons que quand le nombre de ces serveurs augmente à l'infini, les comportements strict et simple tendent à être les mêmes. Comme la convolution est associative, le processus de départ correspondant à une courbe d'arrivée A traversant un serveur offrant une courbe de service simple β est le même que le processus de départ à travers des serveurs en tandem offrant des courbes de service simples respectives β_i. Nous nous intéresserons donc principalement aux processus de départ pour les serveurs stricts.

Remarque 7.1. *En utilisant le service strict (qui n'est pas préservé par concaténation), nous allons monter que les trajectoires du système pour la notion de service intermédiaire (qui est préservée par concaténation) sont arbitrairement proches des trajectoires correspondant au service simple.*

Courbe de délai pur

Avant de prouver le résultat pour toute courbe de service convexe, prouvons-le d'abord pour une courbe δ_d, où $\delta_d : t \mapsto 0$ si $t \leq d$; $+\infty$ sinon. Ce cas permet de montrer l'intuition de la preuve.

Soit A un processus d'arrivée cumulée. Soit n le nombre de serveurs en tandem et d/n le délai pur de chaque serveur (chaque serveur offre une courbe de service $\delta_{d/n}$). Calculons le délai de chaque paquet dans le processus. Soit B_n le processus de départ de A après avoir traversé n serveurs en tandem quand les services sont *stricts et exacts*, et soit t_0 l'instant où arrive le paquet dont nous voulons calculer le délai.

1. On pourrait écrire $\mathcal{S}_{strict}(\beta) \subseteq \mathcal{S}_{intermediaire}(\beta) \subseteq \mathcal{S}_{simple}(\beta)$.

Le délai dans le premier serveur est borné supérieurement par d/n, et le paquet est servi à l'instant $t_1 \leq t_0 + d/n$. Le processus après le premier serveur contient des rafales (c'est une fonction en escalier), et le temps entre deux rafales est au moins d/n. Alors, au second serveur (qui est exact), comme chaque paquet de la période chargée a été servi au même instant, les paquets arrivent au début d'une période chargée et seront servis à l'instant $t_2 = t_1 + d/n$. Il en est de même pour les $n-2$ serveurs suivants. Alors, le délai total δ pour le paquet considéré est borné par :

$$\frac{n-1}{n} d \leq \delta \leq d.$$

Quand n tend vers l'infini, le délai du paquet tend vers d. Comme cela est vrai pour chaque paquet, on a $B_n(t) \to A(t-d)$.

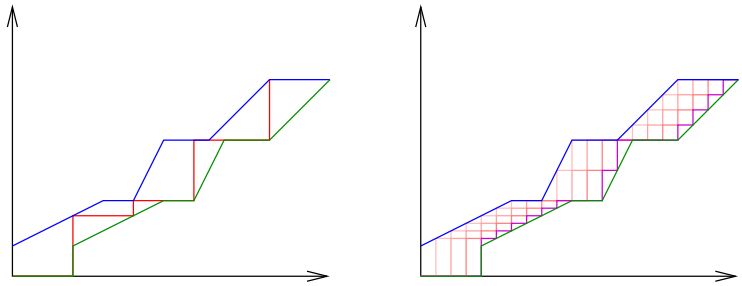


Figure 7.2 : (a) Un seul serveur de service β_d ; (b) quatre serveurs en tandem de services $\beta_{d/4}$: du clair au foncé, processus de départ après chacun des serveurs. Processus : arrivée (bleu), départ strict (rouge), départ simple (vert)

Segment élémentaire : $\beta_{\lambda,d}$

Remarque 7.2. *Dans la suite, on appellera parfois* processus *le mouvement des données en entrée ou en sortie d'un serveur. En fait, cela correspond à la fonction cumulée de la quantité de données en entrée ou sortie du serveur. Le mot « processus » insiste sur le fait qu'on s'intéresse plus particulièrement à chaque bit qui arrive.*

Nous prouvons maintenant le théorème pour une courbe de la forme suivante : $\beta_{\lambda,d}$: $t \mapsto \lambda t$ si $t \leq d$, et $+\infty$ sinon. Ce résultat sera utilisé pour prouver le théorème pour toute fonction convexe.

Comme dans le cas précédent, on décompose le serveur de service $\beta_{\lambda,d}$ en n serveurs de service $\beta^n_{\lambda,d/n}$ avec $\beta_{\lambda,d} = \beta^n_{\lambda,d/n}$ (convolution). Soit A le processus cumulé des arrivées. Le même processus de départ cumulé peut être obtenu en considérant que le flux traverse un serveur de courbe de service simple minimal $\beta_{\lambda,d}$ ou n serveurs en tandem de courbes de service minimal $\beta_{\lambda,d/n}$. Le processus de départ obtenu pour un service simple, si le serveur est exact, est $A * \beta_{\lambda,d}$. Graphiquement, c'est l'enveloppe inférieure de l'ensemble des points

formé par les segments de pente λ et de longueur d et dont l'origine parcourt A, comme montré sur la figure 7.3.

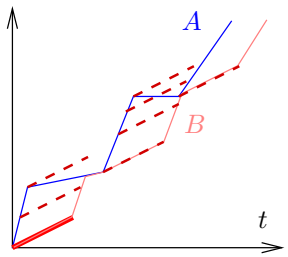


FIGURE 7.3 : Convolution avec un segment élémentaire : $B = A * \beta_{\lambda,d}$. En gras non hachuré : $\beta_{\lambda,d}$. Pointillés : certains segments considérés.

Le comportement est différent quand les services sont stricts. Étudions ce qui se passe : nous supposons que les services sont exacts pour construire les processus d'arrivée et de départ. La période chargée d'un processus traversant un serveur de courbe de service strict $\beta_{\lambda,d/n}$ vaut au plus d/n. Si elle vaut exactement d/n, alors la pente du processus de départ dans cette période chargée vaut exactement λ. Sinon, cela signifie que le processus d'arrivée a « rencontré » le segment de pente λ qui commence au début de la période chargée, ce qui signifie que la période chargée se termine au moment de l'intersection et une nouvelle période chargée peut commencer. Le reste du segment de longueur d n'est alors plus utile pour le calcul de la convolution, car il est recouvert par le segment qui commence à cette intersection. Après avoir traversé un tel serveur, les pentes de $B^{(1)}$ — le processus de départ après avoir traversé le premier serveur — valent au plus λ, et il peut y avoir des discontinuités dues à des rafales. Quand la pente est strictement inférieure à λ, cela signifie que le délai est nul. Lorsqu'il y a des discontinuités, la période chargée qui se termine à ce moment-là est de longueur d/n. Les différents cas qui peuvent se produire sont présentés figure 7.4.

D'après ce qui a été dit précédemment, on peut déduire que le processus de départ $B^{(1)}$ peut s'écrire

$$B^{(1)}(t) = \min_{s \in \mathcal{D} \cap [0,t]} A(s) + \beta_{\lambda,d/n}(t-s),$$

où $\mathcal{D} \subset \mathbb{R}_+$ tel que $0 \in \mathcal{D}$ et $\forall s \in \mathcal{D}, \exists u \in \mathcal{D}$ tel que $u - s \leq d/n$.

Considérons $B^{(i)}$, les processus de départ après avoir traversé le i-ème serveur ($B^{(i-1)}$ est le processus d'arrivée dans le i-ème serveur). On peut faire les mêmes observations que pour le passage de A à $B^{(1)}$: les pentes de $B^{(i)}$ valent au plus λ, et il peut y avoir des rafales. Une période chargée (de longueur non nulle) commence nécessairement lors d'une rafale. Pour chaque rafale dans $B^{(i)}$, $i > 1$, disons au temps t, il y a eu une rafale à l'instant $t - d/n$ dans $B^{(i-1)}$, et par récurrence, au temps $t - (i-1)d/n$ pour $B^{(1)}$; la période chargée dans le premier serveur commence à l'instant $t - id/n$. Les rafales correspondent à la fin d'une période chargée de longueur d/n. Alors, s'il existe une période chargée de longueur

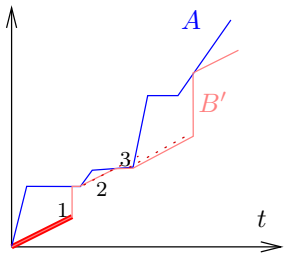


FIGURE 7.4 : Différents cas pour les courbes de service strict (segment élémentaire). 1 : La période chargée vaut exactement d/n, la pente du processus de départ pendant cette période est λ et il y a une rafale à la fin de la période chargée. 2 : la période chargée dure strictement ou moins que d/n car le segment de pente λ et de longueur d/n intersecte la courbe A. Après l'intersection, le segment est au-dessus des courbes ou des segments qui démarrent la prochaine période chargée. 3 : La pente du processus d'arrivée vaut moins que λ à la fin de la période chargée, donc le processus de départ n'est pas retardé.

strictement inférieure à d/n dans le i-ème serveur, et qu'elle commence à l'instant t, cela signifie que les courbes $s \mapsto \mathrm{B}^{(i-1)}(t) + \beta_{\lambda,d/n}(s)$ et $s \mapsto \mathrm{B}^{(i-1)}(t+s)$ s'intersectent (ce qui correspond au cas de la figure 7.4). La pente de $\mathrm{B}^{(i-1)}$ à cette intersection (on l'appellera l'instant t') vaut strictement moins que λ, et on a donc $\mathrm{B}^{(i-1)}(t') = \mathrm{A}(t')$. La période chargée correspondante dans le premier serveur commence à l'instant $t-(i-1)d/n$. Entre les instants $t-(i-1)d/n$ et t', $\mathrm{B}^{(i)}$ est affine de pente λ. En appliquant ce résultat avec $i := n$, on obtient, avec $\mathrm{B}_n = \mathrm{B}^{(n)}$,

$$\mathrm{B}_n(t) = \min_{s\in\mathcal{D}\cap[0,t]} \mathrm{A}(s) + \beta_{\lambda,d}(t-s),$$

où $\mathcal{D} \subset \mathbb{R}_+$ tel que $0 \in \mathcal{D}$ et $\forall s \in \mathcal{D}$, $\exists u \in \mathcal{D}$ tel que $u - s \leq d/n$. Les éléments de $\mathcal{D}$ correspondent aux débuts de périodes chargées dans le premier serveur (dont les périodes chargées de longueur nulle). La distance horizontale entre B et B_n est bornée par d/n :

$$\mathrm{B}_n \geq \mathrm{B} \geq \mathrm{B}_n(.-d/n) \tag{7.1}$$

et quand $n \to \infty$, on a $\mathrm{B}_n \to \mathrm{B}$.

Courbe de service convexe

Soit β une courbe de service strict affine par morceaux et convexe composée de k segments de pentes respectives λ_i et de distance horizontale d_i, et d'une demi-droite de pente $\lambda \geq \lambda_k$. On peut écrire $\beta = \bigstar_{i=1}^{k} \beta_{\lambda_i,d_i} * \beta_{\lambda,\infty}$, et décomposer cette courbe de service en $kn+1$ courbes : la courbe $\beta_{\lambda,\infty}$, plus n courbes $\beta_{\lambda_i,d_i/n}$ pour chaque β_{λ_i,d_i} (chacun des k morceaux finis est coupé en n parties égales).

Les serveurs étant supposés exacts, le comportement d'un serveur de courbe de service $\beta_{\lambda,\infty}$ est le même si le service est strict ou non. Plaçons ce serveur en premier. L'ordre des

serveurs est représenté sur la figure 7.5. Soit A un processus d'arrivée pour la concaténation des serveurs.

Notation Nous appelons B_0 (resp. B'_0) le processus de départ après le premier serveur quand le service est simple (resp. strict), et B_i (resp. B'_i) le processus de départ après $n \times i$ serveurs — de courbes de service respectives $\beta_{\lambda_i,d_i/n}$ — quand le service est simple (resp. strict) et exact. On a $B_0 = B'_0$.

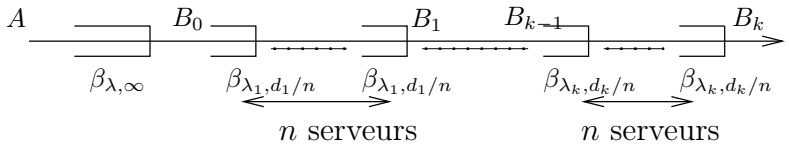


FIGURE 7.5 : Décomposition d'un serveur dont la courbe de service est convexe et affine par morceaux en la concaténation de serveurs élémentaires. La sortie d'un serveur est égale à l'entrée du suivant

Donc ces courbes sont obtenues de la manière suivante (pour n fixé) :

- les B_i, comme des convolutions successives (entre B_{i-1} et B_i on traverse n serveurs de service β_{λ_i,d_i}) :
$$\forall i \in \mathbb{N}^*, B_i = B_{i-1} * \beta_{\lambda_i,d_i};$$
- les B'_i, par constructions successives, comme dans les parties précédentes, que l'on notera $Constr$ ($Constr_i$ correspond à la traversée de n serveurs β_{λ_i,d_i}) :
$$\forall i \in \mathbb{N}, B'_i = Constr_i(B'_{i-1}).$$

Avec ces notations, l'équation 7.1 nous dit que, pour un *même* processus d'arrivée A, après la traversée de n serveurs β_{λ_i,d_i} on a *par construction* :

$$Constr_i(A) \geq A * \beta_{\lambda_i,d_i} \geq Constr_i(A)(. - \frac{d_i}{n}). \tag{7.2}$$

Il y a deux paramètres : A et i.

En utilisant l'isotonicité de la convolution [Le Boudec 2001, p. 115] [2] et l'équation 7.2, nous allons prouver par récurrence sur k que quel que soit $k \in \mathbb{N}^*$:

$$B'_k \geq B_k \geq B'_k(. - \sum_{i=1}^{k} \frac{d_i}{n}) \tag{H_k}$$

Initialisation (H_1) : le cas $k = 1$ correspond à l'application directe de l'équation 7.2 avec $A = B_0 = B'_0$ et $i = 1$ (c'est la traversée de n serveurs β_{λ_1,d_1}) :

2. Si $f_1 \geq f_2$, alors $f_1 * g \geq f_2 * g$.

$$Constr_1(B'_0) \geq B_0 * \beta_{\lambda_1,d_1} \geq Constr_1(B'_0)(. - \frac{d_1}{n}), \text{ soit}$$

$$B'_1 \geq B_1 \geq B'_1(. - \frac{d_1}{n})$$

Récurrence : on suppose (H_k) vraie et on veut montrer (H_{k+1})

- L'équation 7.2 appliquée à $i = k+1$ et $A = B'_k$ donne

  $$Constr_{k+1}(B'_k) \geq B'_k * \beta_{\lambda_{k+1},d_{k+1}}$$

  Et comme par hypothèse de récurrence $B'_k \geq B_k$, l'isotonicité nous donne

  $$B'_k * \beta_{\lambda_{k+1},d_{k+1}} \geq B_k * \beta_{\lambda_{k+1},d_{k+1}}$$

  et on a donc

  $$Constr_{k+1}(B'_k) \geq B_k * \beta_{\lambda_{k+1},d_{k+1}}$$

  soit

  $$B'_{k+1} \geq B_{k+1}$$

  c'est-à-dire la partie de gauche dans (H_{k+1})
- De même, l'équation 7.2 appliquée à $i = k+1$ et $A = B'_k(. - \sum_{i=1}^{k} \frac{d_i}{n})$ donne

  $$B'_k(. - \sum_{i=1}^{k} \frac{d_i}{n}) * \beta_{\lambda_{k+1},d_{k+1}} \geq Constr_{k+1}(B'_k(. - \sum_{i=1}^{k} \frac{d_i}{n}))(. - \frac{d_{k+1}}{n})$$

  Or pas construction on a :

  $$Constr_{k+1}(B'_k(. - \sum_{i=1}^{k} \frac{d_i}{n})) = (Constr_{k+1}(B'_k))(. - \sum_{i=1}^{k} \frac{d_i}{n})$$

  et par hypothèse de récurrence $B_k \geq B'_k(. - \sum_{i=1}^{k} \frac{d_i}{n})$, ce qui donne avec l'isotonicité de la convolution et l'égalité ci-dessus :

  $$B_k * \beta_{\lambda_{k+1},d_{k+1}} \geq (Constr_{k+1}(B'_k))(. - \sum_{i=1}^{k} \frac{d_i}{n})(. - \frac{d_{k+1}}{n})$$

  soit, par définition des B_i, par définition des B'_i, et par composée des décalages :

  $$B_{k+1} \geq B'_{k+1}(. - \sum_{i=1}^{k+1} \frac{d_i}{n})$$

c'est-à-dire la partie de droite dans (H_{k+1})

Le nombre k de segments initiaux étant fixé, lorsque le nombre n de subdivisions tend vers l'infini, la distance horizontale entre B_k et B'_k tend vers 0.

Ceci termine la preuve.

■

7.2. Courbes de service résiduel

7.2.1. Résultats de stabilité pour le modèle fluide

Ci-dessous, un théorème sur le service résiduel qui est fourni à un flux A_2 en présence d'un flux agrégé A_1 en multiplexage aveugle. La deuxième partie du théorème fait une hypothèse supplémentaire sur la politique de service.

Théorème 7.3. *Prenons un serveur offrant un service strict de courbe β aux flux agrégés $A_1 + A_2$ et supposons que le flux A_1 est borné supérieurement par α_1. On a alors :*

1. *le serveur offre un service simple, de courbe $(\beta - \alpha_1)_\uparrow$, au flux A_2 ;*
2. *si de plus A_1 a une priorité fixe supérieure à A_2, alors le flux A_2 reçoit un service de courbe $(\beta - \alpha_1)_\uparrow$ qui est de type* strict.

Un résultat similaire existe dans le modèle RTC avec des priorités fixes. Néanmoins, nous donnons une preuve directe dans le cadre du Network Calculus. En NC, il existe un résultat disant que A_2 reçoit un service de courbe $(\beta - \alpha_1)_+$. Obtenir ce type de résultat avec la clôture positive croissante peut être pratique lorsqu'on manipule des courbes d'arrivée sous-additives mais non concaves.

Preuve. En plus des deux parties du théorème, nous donnons un exemple dans lequel il n'y a pas de priorité fixe entre les deux flux, et où le service obtenu n'est pas strict.

Nous montrons d'abord que si A_1 a une priorité plus élevée que A_2, alors le serveur offre un service strict de courbe $(\beta - \alpha_1)_\uparrow$ à A_2. Soit u et v, $u < v$ deux instants dans la même période chargée pour A_2. Ils sont donc aussi dans la même période chargée pour le flux agrégé, et nous avons :

$$B_1(v) + B_2(v) \geq B_1(u) + B_2(u) + \beta(v - u).$$

Soit p le début de la période chargée de u pour le flux F_1. Comme le période entre p et u est chargée pour F_1, et que u est dans une période chargée pour F_2, aucune donnée ne peut être servie pour F_2 entre p et v. Nous avons alors :

$$\begin{aligned} B_1(u) - B_1(p) &\geq \beta(u-p) && (7.3)\\ B_2(u) - B_2(p) &= 0 && (7.4)\\ A_1(p) - B_1(p) &= 0. && (7.5) \end{aligned}$$

Entre les instants p et v, on a de même :

$$B_2(v) - B_2(p) + B_1(v) - A_1(p) \geq \beta(v-p).$$

Comme $B_1(v) - A_1(p) \leq A_1(v) - A_1(p) \leq \alpha_1(v-p)$, nous obtenons :

$$B_2(v) - B_2(p) \geq \beta(v-p) - \alpha_1(v-p).$$

On peut appliquer la formule ci-dessus pour chaque $w \in [p,v]$, et alors w reste dans la même période chargée que v :

$$\forall w \in [p,v] \;\; B_2(w) - B_2(p) \geq \beta(w-p) - \alpha_1(w-p).$$

Comme B_2 est croissante, nous avons aussi

$$\forall w \in [p,v] \;\; B_2(v) - B_2(p) \geq B_2(w) - B_2(p) \geq \beta(w-p) - \alpha_1(w-p)$$

et

$$\begin{aligned} B_2(v) - B_2(p) &\geq \sup_{w\in[p,v]} \beta(w-p) - \alpha_1(w-p) && (7.6)\\ &\geq \sup_{s\in[0,v-p]} \beta(s) - \alpha_1(s) && (7.7)\\ &\geq \sup_{s\in[0,v-u]} \beta(s) - \alpha_1(s). && (7.8) \end{aligned}$$

Comme $B_2(u) = B_2(p)$, nous avons

$$B_2(v) - B_2(p) \geq \sup_{s\in[0,v-u]} \beta(s) - \alpha_1(s).$$

De plus, comme B_2 est croissante,

$$B_2(v) - B_2(p) \geq \sup_{s\in[0,v-u]} (\beta(s) - \alpha_1(s))_+.$$

Pour montrer qu'en multiplexage aveugle le serveur offre un service simple de courbe $(\beta - \alpha_1)_\uparrow$ est direct d'après ce qu'on a écrit ci-dessus : les équations (7.6) et (7.7) restent

vraies, et $A_2(p) = B_2(p)$, donc

$$B_2(v) \geq A_2(p) + \sup_{s\in[0,v-p]} \beta(s) - \alpha_1(s) \geq A_2 * (\beta - \alpha_1)_\uparrow.$$

Donnons maintenant un exemple où le service résiduel n'est pas strict. Travaillons avec le modèle de données infinitésimal et considérons un noeud qui sert des données avec un débit constant : 2. Ce noeud offre un service strict de courbe $\beta(t) = 2t$. Prenons deux flux (agrégés) qui traversent ce serveur avec des fonctions cumulées d'arrivées respectives $A_1(t) = t + 1$ et $A_2(t) = t$. Une courbe d'arrivée pour le flux 2 est $\alpha_2(t) = t$. La figure 7.6 montre la trajectoire du système pour la politique de service suivante : d'abord, pour $0 \leq t \leq 1$, le flux 1 a la priorité maximale, pour pour $t > 1$, c'est le flux 1 qui a la priorité maximale. Comme on a $A_1(t) + A_2(t) = 2t + 1$, le serveur est toujours en activité et la somme des fonctions de départ cumulées vaut $B_1(t) + B_2(t) = 2t$. Alors, le flux 1 reçoit un service simple de courbe $\beta_1(t) = (\beta - \alpha_2)_+(t) = t$ (en effet, $B_1 \geq A_1 * \beta$), mais ce service n'est pas strict : durant la période $1 \leq t \leq 2$, le flux 1 a des données en file d'attente au niveau du serveur, mais il n'est jamais servi.

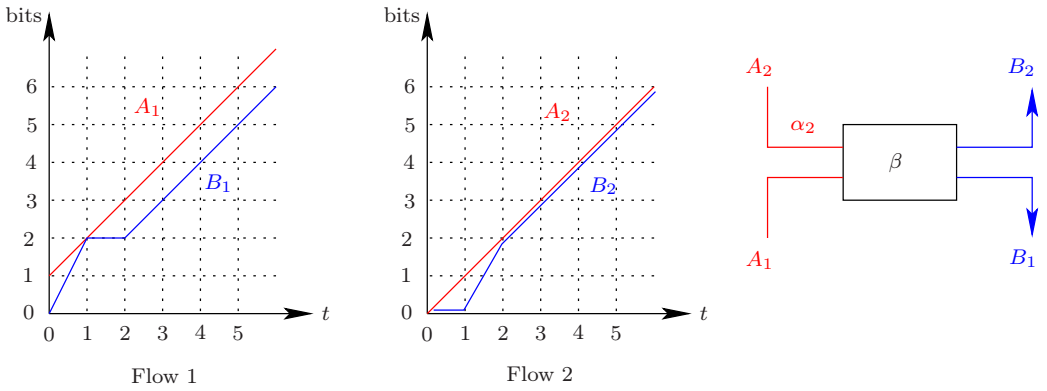


Figure 7.6 : Les services résiduels ne sont pas nécessairement stricts

■

7.2.2. Qu'en est-il de la taille des paquets ?

Théorème 7.4. *Prenons un serveur strict de courbe β pour $A_1 + A_2$. Supposons que le flux A_1 est α_1-contraint et qu'il a une priorité (non-préemptive) supérieure à celle de A_2 ; de plus on note $\ell_{2,\max}$ la taille maximum des paquets dans F_2. On a alors :*

1. *le serveur garantit un service simple de courbe $(\beta - \alpha_1)_\uparrow$ à $F_2^{(0)}$;*
2. *le serveur garantit un service strict de courbe $(\beta - \alpha_1 - \ell_{2,\max})_\uparrow$ à $F_2^{(0)}$;*
3. *le serveur garantit un service strict de courbe $(\beta - \ell_{2,max})_\uparrow$ à $F_1^{(0)}$.*

Preuve. La première partie est une conséquence directe du théorème 7.3-1. Les deuxième et troisième parties peuvent se montrer de la même manière que le théorème 7.3, ou

alors on peut faire les observations suivantes pour appliquer le 7.3-2. En effet, comme le système est non-préemptif, dès qu'un paquet du flux F_2 commence à être servi, un paquet du flux 1 peut arriver dans le serveur et démarrer une période chargée pour le flux 1. Mais le paquet de F_2 peut être vu comme un flux prioritaire à cause de la non-préemption. Or le flux 2 est contraint par $t \mapsto \ell_{2,\max}$ donc le serveur offre un service strict au flux 1 de courbe $(\beta - \ell_{2,max})_\uparrow$

De même, dans la preuve du théorème 7.3, A_1 peut être remplacé par $A_1 + \ell_{2,max}$, qui est un flux contraint par $t \mapsto \ell_{2,\max} + \alpha_1(t)$, et donc A_2 reçoit un service strict de courbe $(\beta_1 - \alpha_1 - \ell_{2,\max})_\uparrow$.

Nous pouvons aussi adapter l'exemple de la preuve du théorème 7.3 pour montrer que $(\beta - \alpha_1)_\uparrow$ peut ne pas être strict. Nous prenons encore $\beta(t) = 2t$, $A_2(t) = \alpha_2(t) = 1 + t$, et nous posons $\ell_{2,\max} = 1$, $\alpha_1(t) = 1/2 + t$ et $A_1(t + 1/2) = \alpha_1(t)$ pour $t < 1/2$. À l'instant 0, un paquet de taille $\ell_{2,\max}$ arrive, donc il est servi car aucun paquet du flux 1 n'est dans le serveur en 0. À $t = 1$, le paquet de taille $\ell_{2,\max}$ est servi, donc le paquet du flux 1 peut être servi. Puis le comportement est le même que dans la preuve du théorème 7.3 : entre les instants 1 et 2, aucun paquet du flux 2 n'est servi, alors que $(\beta - \alpha_1)_\uparrow(t) = 2(t - 1/4)_+$, et ainsi, l'intervalle durant lequel aucun paquet du flux 2 n'est servi est une période chargée pour ce flux, et elle ne peut pas durer plus de 1/4.

■

Appliqué directement, ce théorème peut être utilisé pour calculer une courbe de service (strict) minimal pour chaque flux en priorité fixe.

Corollaire 7.5. *Prenons un unique serveur offrant un service strict minimal de courbe β, et m flux agrégés $F_1, \ldots, F_m$ qui le traversent. Le flux F_i a une taille maximale de paquets $\ell_{1,\max}$, et est contraint supérieurement par la courbe d'arrivée α_i. Les flux sont triés par ordre de priorité : $F_i \prec F_j \Leftrightarrow i < j$. Pour chaque $i \in \{2, \ldots, m\}$,*

1. *le serveur offre un service simple $(\beta - \sum_{j<i} \alpha_j - \vee_{k>i} \ell_{k,max})_+$ au flux F_i.*
2. *le serveur offre un service strict de courbe $(\beta - \sum_{j<i} \alpha_j - \vee_{k\geq i} \ell_{k,max})_+$ au flux F_i.*

7.3. « Composition » de services résiduels

Dans cette section, nous étudions plus précisément les scénarios pour lesquels la composition de services résiduels aurait été d'un intérêt pratique. Nous montrons que même si le caractère strict du service est perdu, les calculs restent possibles et valides. Il existe un type de scénarios où la composition est efficace : les serveurs en tandem avec des flux transverses imbriqués (nested flows).

7.3.1. Serveurs en tandem

Prenons n serveurs simples en tandem avec des courbes de service respectives β_j. S'il n'y a qu'un seul flux, le serveur lui offre un service de courbe $\beta_1 * \cdots * \beta_n$.

7.3.2. Un flux transverse

Prenons de plus un flux F_2 qui traverse les serveurs $\beta_b, \ldots, \beta_e$ (comme *begining* et *end*). Nous aimerions pouvoir écrire que le flux F_1 reçoit un service de courbe

$$\beta_1 * \cdots \beta_{b-1} * (\beta_b * \cdots * \beta_e - \alpha_2)_+ * \beta_{e+1} * \cdots * \beta_n.$$

Les calculs montrent que c'est le cas : en commençant par étudier la courbe de service entre les serveurs b et e, puis en obtenant les autres par concaténation, ce qui est valide pour tout service.

$\forall t_i \in \mathbb{R}_+$, il existe t_{i-1}, le premier instant de la période chargée de t_i qui vérifie

$$F_1^{(i)}(t_i) + F_2^{(i)}(t_i) \geq F_1^{(i-1)}(t_{i-1}) + F_2^{(i-1)}(t_{i-1}) + \beta_i(t_i - t_{i-1}).$$

Alors, $\forall t_e \in \mathbb{R}_+$, il existe $t_{b-1}, \ldots, t_{e-1}$ tel que

$$F_1^{(i)}(t_e) + F_2^{(i)}(t_e) \geq F_1^{(b-1)}(t_{b-1}) + F_2^{(b-1)}(t_{b-1}) + \sum_{i=b}^{e} \beta_i(t_i - t_{i-1}),$$

et

$$F_1^{(e)}(t_e) \geq F_1^{(b-1)}(t_{b-1}) + \sum_{i=b}^{e} \beta_i(t_i - t_{i-1}) - \alpha_2(t_e - t_{b-1}).$$

De plus, $F_1^{(e)}(t_e) \geq F_1^{(b-1)}(t_{b-1})$, d'où

$$F_1^{(e)}(t_e) \geq F_1^{(b-1)}(t_{b-1}) + (\overset{e}{*}_{i=b} \beta_i(t_e - t_{b-1}) - \alpha_2(t_e - t_{b-1}))_+.$$

7.3.3. Plusieurs flux imbriqués

Supposons à présent qu'il existe m flux transverses $F_1, \ldots, F_m$. Le flux F_i interfère avec F_0 sur I_i, un sous-ensemble des n serveurs définis précédemment, tel que $I_i = \{b_i, \ldots, e_i\}$ (un sous-chemin, donc) ; et supposons de plus que les flux transverses sont imbriqués : $\forall i, j$, soit $I_i \cap I_j = \emptyset$, soit $I_i \subseteq I_j$, soit $I_j \subseteq I_i$. Enfin, supposons que les priorités des flux sont fixes, forment un ordre total, et respectent la règle suivante :

$$\text{(R)} \qquad \forall i, j \in \{1, \ldots, m\}, \qquad F_i > F_j \Rightarrow I_i \subseteq I_j.$$

Pour $i \in \{0, \ldots, m\}$, nous notons P_i l'ensemble des flux qui ont une priorité plus élevée

que celle du flux F_i et qui interfèrent avec ce flux ($P_i = \{j \in \{0,\dots,m\} \mid F_j \succ F_i \text{ and } I_j \subseteq I_i\}$) et P_i^{im} l'ensemble des flux qui ont *la* priorité la plus petite parmi les flux de P_i, soit $P_i^{im} = \{j \in P_i \mid \forall k \in P_i \setminus \{j\}\ F_k \succ F_j\}$. Enfin, notons N_i l'ensemble des serveurs traversés par qui ne sont pas croisés par des flux de priorité plus élevée, soit $N_i = \{j \in I_i \mid \forall k \in \{1,\dots,m\},\ F_k \succ F_i \Rightarrow j \notin I_k\}$.

Nous allons montrer par récurrence que les services associé aux différents flux peuvent être supprimé un par un, dans l'ordre décroissant des priorités, pour obtenir une courbe résiduelle globale, même si le caractère strict des services n'est pas préservé. Plus formellement, nous avons le théorème suivant :

Théorème 7.6. *Si les flux respectent la règle (R), et en utilisant les définitions ci-dessus, pour tout $i \in \{0,\dots,m\}$, le serveur fournit au flux F_i un service simple de courbe*

$$\beta_i' = \mathop{\ast}_{j\in N_i} \beta_j * \mathop{\ast}_{k\in P_i^{im}} (\beta_k' - \alpha_k)_+.$$

Preuve.

Nous montrons ce résultat par récurrence.

Hypothèse de récurrence (H_m) Pour m flux transverses : soit m flux $F_1,\dots,F_m$, qui interfèrent avec une ligne de n serveurs en priorités, et qui respectent la règle (R). Alors pour tout flux agrégé F_{m+1} de priorité plus basse que celles de $F_1,\dots,F_m$, on a :

1. pour chaque serveur $j \notin \bigcup_{i\in\{1,\dots,m\}} I_i$, $\forall t \in \mathbb{R}_+, \exists u_j$ tel que

$$F_{m+1}^{(j)}(t) \geq F_{m+1}^{(j-1)}(u_j) + \beta_j(t-u_j);$$

2. pour tout $k \in P_{m+1}^{im}$, $\forall t \in \mathbb{R}_+$, $\exists u_k$ tel que

$$F_{m+1}^{(e_k)}(t) \geq F_{m+1}^{(b_k-1)}(u_k) + (\beta_k' - \alpha_k)_+(t-u_k).$$

Initialisation (H_0) est vraie : comme il n'y a pas de flux qui interfère il suffit, pour tout serveur, d'étudier le premier cas. Par définition des courbes de service strict, le premier cas est valide.

Récurrence Supposons (H_m) vraie, et montrons (H_{m+1}). Prenons une ligne de n serveurs, ainsi que $m+1$ flux avec priorités fixes. Sans perte de généralité, on peut supposer que F_{m+1} a la priorité la plus basse et que (H_m) est vraie pour $F_1,\dots,F_m$. Prenons aussi F_{m+2} un serveur de priorité plus basse que F_{m+1}. Alors, trois cas peuvent se produire :

1. le serveur j n'interfère pas avec $F_1,\dots,F_{m+1}$. Alors le serveur j offre un service strict

de courbe β_j, et $\forall t \in \mathbb{R}_+$, $\exists u_j$ tel que

$$F^{(j)}_{m+2}(t) \geq F^{(j-1)}_{m+2}(u_j) + \beta_j(t - u_j).$$

2. pour chaque $k \in P^{im}_{m+2}$ tel que $I_k \cap I_{m+1} = \emptyset$, comme F_k a une priorité plus élevée que F_{m+1}, ces deux flux n'interfèrent pas. Ainsi, en utilisant l'hypothèse de récurrence, nous avons $\forall t \in \mathbb{R}_+$, $\exists u_k$ tel que

$$F^{(e_k)}_{m+2}(t) \geq F^{(b_k-1)}_{m+2}(u_k) + (\beta'_k - \alpha_k)_+(t - u_k).$$

3. les flux F_{m+2} et F_{m+1} interfèrent sur I_{m+1}. Chaque flux qui interfère avec F_{m+1} a une priorité plus élevée et respecte la règle (R). Soit $\tilde{\beta_1}, \ldots, \tilde{\beta_\ell}$ la suite triée des courbes de service des serveurs traversés par F_{m+2} et F_{m+1} (c'est-à-dire seulement celles traversée par F_{m+1}), $\tilde{\beta_q}$ étant de la forme β_j pour un j donné, ou de la forme $(\beta'_k - \alpha_k)_+$ pour un k vérifiant (H_m). Pour tout u_ℓ, il existe $u_0, \ldots, u_{\ell-1}$ tel que pour chaque $p \in \{1, \ldots, \ell\}$, on a
$$F^{(r_p)}_{m+2}(u_p) + F^{r_p}_{m+1}(u_p) \geq F^{q_p}_{m+2}(u_{p-1}) + F^{q_p}_{m+1}(u_{p-1}) + \tilde{\beta_p}(u_p - u_{p-1}),$$
avec $r_p = j$ et $q_p = j - 1$ si $\tilde{\beta_p} = \beta_j$ ou $r_p = e_k$ et $q_p = b_k - 1$ if $\tilde{\beta_p} = (\beta'_k - \alpha_k)_+$. De plus, comme les serveurs forment une suite, nous avons $r_p = q_{p+1}$ pour $p \in \{1, \ldots, \ell - 1\}$ et $q_1 = b_{m+1} - 1$ et $r_\ell = e_{m+1}$.
En sommant ces ℓ inéquations, nous obtenons :

$$F^{(e_{m+1})}_{m+2}(u_\ell) + F^{(e_{m+1})}_{m+1}(u_\ell) \geq F^{(b_{m+1}-1)}_{m+2}(u_0) + F^{(b_{m+1}-1)}_{m+1}(u_0) + \sum_{p=1}^{\ell} \tilde{\beta_p}(u_p - u_{p-1}).$$

Comme $F^{(e_{m+1})}_{m+1}(u_\ell) - F^{(b_{m+1}-1)}_{m+1}(u_0) \leq F^{(b_{m+1}-1)}_{m+1}(u_\ell) - F^{(b_{m+1}-1)}_{m+1}(u_0) \leq \alpha_{m+1}(u_p - u_0)$ et $F^{(e_{m+1})}_{m+2}(u_\ell) \geq F^{(b_{m+1}-1)}_{m+2}(u_0)$, nous avons

$$F^{(e_{m+1})}_{m+2}(u_\ell) \geq F^{(b_{m+1}-1)}_{m+2}(u_0) + (\sum_{p=1}^{\ell} \tilde{\beta_p}(u_p - u_{p-1}) - \alpha_{m+1}(u_p - u_0))_+$$

et

$$F^{(e_{m+1})}_{m+2}(u_\ell) \geq F^{(b_{m+1}-1)}_{m+2} * (\divideontimes_{p=1}^{\ell} \tilde{\beta_p} - \alpha_{m+1})_+$$

(H_{m+1}) est alors vérifée. Le reste de la preuve correspond au troisième cas de la preuve.

■

Conclusion

Nous avons comparé les services stricts et simples, et explicité ce qui peut être fait avec quel type de service, vis-à-vis de la composition de services, et du calcul du service résiduel. De plus, nous expliquons pourquoi il n'est pas possible de définir une nouvelle notion de courbe de service « intermédiaire » qui soit stable pour ces deux opérations.

Cela a mis en lumière des détails dont nous n'étions pas conscients, comme par exemple, en ce qui concerne les services résiduels, le fait qu'il y a une différence entre le multiplexage aveugle et les priorités fixes. En particulier, le théorème 7.3 montre que travailler avec des priorités fixes permet, lorsque l'on calcule des services résiduels, de ne pas perdre le caractère strict des serveurs considérés. L'étude du modèle RTC nous a été très précieuse dans ce cadre.

Troisième partie :

Utiliser le Network Calculus pour obtenir des bornes

CHAPITRE 8

CONVOLUTION MULTIDIMENSIONNELLE

Dans ce chapitre, nous proposons un nouvel opérateur $(min, +)$ pour traiter le calcul de performances en cas d'agrégation de flux.

Introduction

Nous étudions des configurations dans lesquelles un flux principal subit un trafic transverse qui interfère sur des *sous-chemins*. Nous supposons de plus qu'il n'y a pas d'autre information sur la politique de service que FIFO par flux (multiplexage aveugle).

De telles configurations ont d'abord été étudiées dans [Schmitt 2006a, Schmitt 2006b], où il a été identifié un phénomène dit « Pay Multiplexing Only Once » (PMOO), puis dans [Bouillard 2007a, Bouillard 2008a] où un opérateur (min,+) multidimensionnel a été introduit pour calculer une courbe de service minimum sur un chemin entier. Il était annoncé dans [Bouillard 2007a] que ce nouvel opérateur permettait de calculer une courbe de service bout en bout en temps polynomial, mais un défaut dans l'algorithme a été décelé dans [Bouillard 2008a] et corrigé dans un cadre plus restreint.

Pour expliquer le phénomène PMOO rapidement, disons que lorsque le flux observé est agrégé avec des flux transverses, le service qu'il reçoit peut être énormément réduit au niveau du premier noeud où ils se croisent. Cependant, si c'est le cas, dans les noeuds suivants l'interférence due aux flux transverses ne peut plus être aussi impor-

tante car la compétition pour la ressource (le service) a déjà eu lieu au premier noeud (voir [Schmitt 2006a, Schmitt 2006b] pour les détails concernant le terme PMOO).

Nous reprenons ici l'étude faite dans [Bouillard 2007a]. Avec des hypothèses classiques en NC (courbes d'arrivée concaves, courbes de service convexes), nous montrons certaines propriétés de ce nouvel opérateur ainsi que son application pour obtenir des bornes sur les délais de bout en bout et les charges locales en temps polynomial.

Le début du chapitre (section 8.2) étudie les propriétés ainsi que le calcul de la convolution multidimensionnelle avec ces hypothèses : nous montrons que la convexité est préservée (théorème 8.2) et même si la complexité du calcul de la courbe de sortie reste un problème ouvert, on peut calculer des bornes sur la charge locale pire-cas ou le délai pire-cas de bout en bout en temps polynomial en utilisant l'optimisation linéaire (théorème 8.3). Cela confirme l'importance de la géométrie algorithmique [Bouillard 2008c]. Nous étudions aussi le cas, plus simple, où il n'y a pas de trafic transverse. L'analyse se réduit alors à la $(\min, +)$ convolution de toutes les courbes de service sur un chemin. Avec des courbes de service convexes et affines par morceaux, un théorème (p. 149) nous permet de calculer efficacement cette convolution. Nous nous intéressons aussi à la complexité du calcul des bornes de performance dans ce cas.

8.1. Quantifier le phénomène « Pay Multiplexing Only Once »

Il existe une formule intéressante qui donne une courbe de service de bout en bout pour un chemin dans le cas où le trafic transverse interfère sur des *sous-chemins*, c'est-à-dire des ensembles consécutifs de noeuds. Elle généralise la formule, obtenue dans [Schmitt 2006b], pour un exemple avec trois noeuds et deux flux transverses. Voir ci-dessous les configurations que nous étudions (appelées PMOO) et les notations que nous utilisons, illustrées par la figure 8.1.

Configurations PMOO

- Le flux principal F_0 suit un chemin $\mathfrak{p}$ de n noeuds indexés par $1, 2, ..., n$ et qui ont pour courbes de service respectives β_j, $1 \leq j \leq n$. Le service peut être strict ou simple.
- Le trafic transverse qui interfère avec le chemin est composé des flux F_i, $1 \leq i \leq k$, de courbes d'arrivée respectives sur le chemin α_i.
- Chaque flux transverse F_i intersecte le chemin $\mathfrak{p}$ sur un ensemble de noeuds consécutifs de $\mathfrak{p}$. Il rejoint le chemin observé à partir du noeud s_i et le quitte juste après le noeud e_i (en particulier $s_0 = 1$ et $e_0 = n$).
- Considérons le flux i, $1 \leq i \leq k$. La quantité de données qui ont été servies par le

noeud j jusqu'à l'instant t est noté $A_i^{(j)}(t)$. La quantité de données qui sont entrées dans le noeud s_i jusqu'à l'instant t est noté $A_i^{(s_i-1)}(t)$.

- Dans la suite, nous supposerons qu'il y a un multiplexage aveugle à chaque noeud : la politique de service appliquée à chaque noeud est inconnue, mais on impose qu'elle soit FIFO par flux.

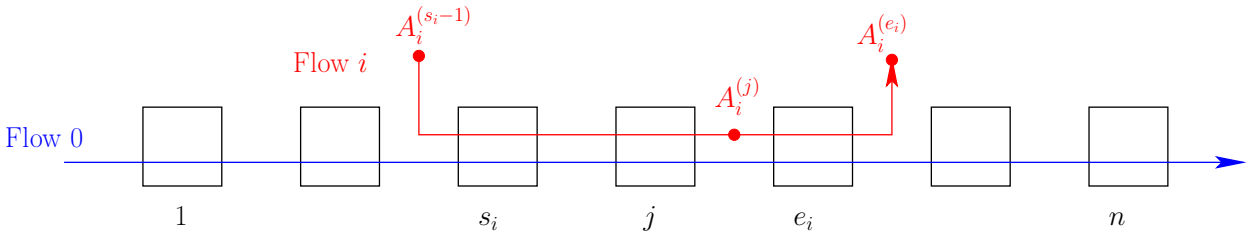


FIGURE 8.1 : Chemin pour lequel chaque flux transverse F_i interfère seulement sur un sous-chemin

Le prochain théorème est prouvé dans [Bouillard 2007a] avec l'hypothèse que tous les services (de courbes β_j) sont stricts. Nous réécrivons ce théorème pour mettre en avant le rôle de l'hypothèse sur les courbes de services stricts.

Théorème 8.1 (Opérateur PMOO multidimensionnel [Bouillard 2007a])**.** *Pour une configuration PMOO, une courbe de service sur l'ensemble du chemin* $\mathfrak{p}$ *au flux* F_0 *est :*

$$\psi(t) = \inf_{\substack{u_1,\dots,u_n \geq 0 \\ u_1+\cdots+u_n = t}} \sum_{j=1}^{n} \beta_j(u_j) - \sum_{i=1}^{k} \alpha_i\Big(\sum_{j=s_i}^{e_i} u_j\Big).$$

Si tous les services, de courbes β_j*, sont stricts, une courbe de service offerte par l'ensemble du chemin est* $\phi(t) = \psi(t)_+$*, où* $x_+ \overset{def}{=} \max(x, 0)$.

Deux résultats classiques de NC peuvent être vus comme des corollaires de ce théorème.

- le cas $n = 2$ sans trafic transverse : deux flux en tandem (théorème 2.3) ;
- le cas $n = k = 1$, sur le service résiduel (théorème 2.4).

8.2. Analyse de performance avec la convolution multidimensionnelle

Dans cette section, nous définissons un nouvel opérateur qui généralise l'inf-convolution classique englobe la formule du théorème 8.1 : nous étudions la complexité du calcul de cet opérateur et des bornes sur les délais et les charges dans le réseau.

8.2.1. Convolution multidimensionnelle

Définition 8.1. *Soit* $\mathrm{J} = \{1, \dots, n\}$ *et* $\mathrm{I} = \{1, \dots, k\}$. *Soit* $\{f_i\}_{i \in \mathrm{I}}$ *une famille de fonctions dans* $\mathcal{F}$, *et* $\{\mathrm{J}_i\}_{i \in \mathrm{I}}$ *une famille de sous-ensembles de* J. *La* convolution multidimensionnelle *associée à ces familles est la fonction* $\psi \in \mathcal{F}$ *définie par :*

$$\psi(t) = \inf_{\substack{u_1, \dots, u_n \geq 0 \\ u_1 + \cdots + u_n = t}} \sum_{i \in \mathrm{I}} f_i\Big(\sum_{j \in \mathrm{J}_i} u_j \Big).$$

Théorème 8.2. *Avec les notations de la définition 8.1, si les fonction* f_i *sont convexes, alors* ψ *est aussi convexe. Si les* f_i *sont de plus affines par morceaux, alors* ψ *est convexe affine par morceaux et peut être calculé en un temps exponentiel en* $n + \sum_{i=1}^{k} |f_i|$.

Preuve. La convexité est préservée car ψ peut se réécrire

$$\psi(t) = \inf_{(u_1, \dots, u_n) \in \mathbb{R}_+^n} \mathrm{H}(t, u_1, \dots, u_n) + \Lambda(t, u_1, \dots, u_n)$$

avec

$$\mathrm{H}(t, u_1, \dots, u_n) \overset{def}{=} \sum_{i \in \mathrm{I}} f_i\Big(\sum_{j \in \mathrm{J}_i} u_j \Big)$$

$$\Lambda(t, u_1, \dots, u_n) \overset{def}{=} \begin{cases} 0 & \text{si } t = u_1 + \cdots + u_n, \\ +\infty & \text{sinon.} \end{cases}$$

Or les deux fonctions H et Λ sont convexes de $\mathbb{R}_+^{n+1}$ dans $\mathbb{R}_{\min}$. Ainsi leur somme est aussi convexe, et en tant qu'infimum calculé sur l'ensemble convexe $\mathbb{R}_+^n$, la fonction ψ est aussi convexe (voir [Rockfellar 1996] pour les opérations préservant la convexité, comme l'infimum de fonctions convexes sur un ensemble convexe). Étudions maintenant le cas où les f_i sont affines par morceaux.

Pour tout $j \in \mathrm{J}$, nous noterons $\mathrm{I}_j = \{i \in \mathrm{I} \mid j \in \mathrm{J}_i\}$. Nous pouvons supposer sans perte de généralité que pour tout $1 \leq i \leq k$, $f_i(0) = 0$. Sinon, il suffit de remarquer que ψ est égal à la même formule avec $\tilde{f}_i(x) = f_i(x) - f_i(0)$ plus la constante $\sum_{i=1}^{k} f_i(0)$. On peut de plus supposer sans perte de généralité que toutes ces fonctions sont croissantes. Si ce n'est pas le cas, on posera $\lambda < 0$ la plus petite des pentes des f_i. Pour tout $t \in \mathbb{R}_+$, on a alors

$$\psi(t) = k\lambda t + \inf_{\substack{u_1, \dots, u_n \geq 0 \\ u_1 + \cdots + u_n = t}} \sum_{i=1}^{k} \tilde{f}_i\Big(\sum_{j \in \mathrm{J}_i} u_j \Big) + \sum_{j=1}^{n} \lambda(|\mathrm{I}_j| - k) u_j.$$

Ainsi, quitte à ajouter la fonction linéaire négative $t \mapsto k\lambda t$, ψ est la convolution multidimensionnelle de $(n+k)$ fonctions croissantes convexes et affines par morceaux. Remarquons

que les nouvelles instances de fonctions obtenues peuvent être stockées en espace linéaire vis-à-vis de l'instance initiale.

Pour chaque fonction f_i, $1 \le i \le k$, avec $\ell_i \stackrel{def}{=} |f_i|$ morceaux dans son support, soit $r_{i,\ell}$ (resp. $\tau_{i,\ell}$) la pente (resp. la distance horizontale, potentiellement $+\infty$) du ℓ-ième morceau à partir de la « gauche ». Comme tous les f_i sont convexes et affines par morceaux, pour une valeur $t \in \mathbb{R}_+$ fixée, la valeur $\psi(t)$ est exactement la solution d'une instance d'un problème d'optimisation linéaire (le *primal*). Cette instance utilise les variables positives $(u_j)_{1\le j\le n}$, et $(v_{i,\ell})_{1\le\ell\le\ell_i}$ pour tout $1 \le i \le k$. La valeur $\psi(t)$ est égale à

$$\inf \sum_{i=1}^{k} \sum_{\ell=1}^{\ell_i} r_{i,\ell} v_{i,\ell}$$

avec les contraintes linéaires : $u_1 + \cdots + u_n \ge t$ et pour tout $1 \le i \le k$,

$$\begin{cases} \sum_{j\in \mathrm{J}_i} u_j \le \sum_{\ell=1}^{\ell_i} v_{i,\ell} \\ v_{i,\ell} \le \tau_{i,\ell} \text{ pour tout } 1 \le \ell \le \ell_i. \end{cases}$$

Nous voulons d'abord trouver le support de ψ : il suffit de remplacer la fonction objectif par $\sup t$ avec les mêmes contraintes, les solutions T (potentiellement $+\infty$) peuvent être trouvées en temps polynomial en résolvant cette instance de programme d'optimisation linéaire. Pour $t > \mathrm{T}$ on a $\psi(t) = +\infty$.

Alors, le théorème de dualité forte [Schrijver 1998] d'optimisation linéaire assure que, lorsqu'il est fini, $\psi(t)$ est égal à l'optimum de l'instance *duale* définie pour les variables positives y, $(y_i)_{1\le i\le k}$ et $(y_{i,\ell})_{1\le\ell\le\ell_i}$. La valeur $\psi(t)$ est égale à $\sup yt - \sum_{i=1}^{k} \sum_{\ell=1}^{\ell_i} y_{i,\ell}\tau_{i,\ell}$ avec les contraintes :

$$\begin{cases} y_i \le y_{i,\ell} + r_{i,\ell} & \text{pour tout } 1 \le i \le k, 1 \le \ell \le \ell_i, \\ y \le \sum_{i\in \mathrm{I}_j} y_i & \text{pour tout } 1 \le j \le n. \end{cases}$$

Ces $c = n + \sum_{i=1}^{k} \ell_i$ contraintes linéaires sur $d = c + k + 1$ variables définissent un polyèdre convexe dans $\mathbb{R}_+^d$ qui est indépendant de t (ce polyèdre n'est pas borné). Comme nous connaissons le support de ψ, nous nous intéressons seulement aux valeurs finies de ψ. Le supremum, lorsqu'il est fini, est atteint en un point extrémal du polyèdre dans $\mathbb{R}_+^d$. De plus, l'ensemble des points extrémaux est fini, son cardinal peut être exponentiel en d, et cet ensemble peut être calculé en temps exponentiel en d [Boissonnat 2001, Schrijver 1998]. Chaque point extrémal $(y, (y_i)_{1\le i\le k}, (y_{i,\ell})_{1\le\ell\le\ell_i}) \in \mathbb{R}_+^d$ définit une fonction affine $t \mapsto yt - \sum_{i=1}^{k} \sum_{\ell=1}^{\ell_i} y_{i,\ell}\tau_{i,\ell}$ et, sur son support, $\psi(t)$ est le supremum de cet ensemble fini de fonctions affines. Par conséquent, ψ est une fonction convexe affine par morceaux qui peut être calculée en temps exponentiel en d. ■

Pour l'instant, lorsque les fonctions f_i sont convexes et affines par morceaux, si l'on veut

calculer et stocker toute la fonction ψ, nous avons seulement un algorithme exponentiel, à part dans des cas très spécifiques où un algorithme polynomial est connu :

- $J_i = J$ pour tout i (cela revient à la somme de fonctions convexe affines par morceaux).
- $I = J$ et $J_i = \{i\}$ pour tout i (cela revient à la convolution classique — voir section 8.3).

Néanmoins, il est d'ores et déjà possible de calculer, en temps polynomial, les bornes pire-cas sur les délais et les charges locales et utilisant ψ, mais sans calculer explicitement la courbe globale.

8.2.2. Bornes sur les délais et les charges locales

Théorème 8.3. *Dans une configuration PMOO, si toutes les courbes de service β_j (resp. les courbes d'arrivée α_i, ainsi que α pour le flux principal F_0) sont convexes (resp. concaves), et affines par morceaux avec un nombre fini de morceaux, alors les bornes pire-cas $D_{\max}(\alpha, \psi)$ et $B_{\max}(\alpha, \psi)$ pour le flux principal F_0 sont calculables en temps polynomial par résolution d'une instance LP avec $\mathcal{O}(\sum_{i=1}^{k} |\alpha_i| + \sum_{j=1}^{n} |\beta_j|)$ contraintes linéaires sur $\mathcal{O}(\sum_{i=1}^{k} |\alpha_i| + \sum_{j=1}^{n} |\beta_j|)$ variables (il faut remplacer ψ par $\phi = \psi_+$ si les services sont stricts).*

Preuve. Considérons $C(\alpha, \psi) \stackrel{def}{=} \{(t, z) \in \mathbb{R}_+ \times \mathbb{R} \mid \psi(t) \le z \le \alpha(t)\}$ l'aire comprise entre la courbe d'arrivée α et la courbe de service de bout en bout ψ définie au théorème 8.1. Cette fonction ψ est un cas particulier de la convolution multidimensionnelle, et avec les hypothèses sur α_i et β_j, le théorème 8.2 assure que ψ est convexe (de même pour ψ_+). Comme α est concave, $C(\alpha, \psi)$ est un ensemble convexe. En fait, en réécrivant $\alpha(t) \stackrel{def}{=} \min_{1 \le p \le |\alpha|}(a_p t + b_p)$, on a $(t, z) \in C(\alpha, \psi)$ si et seulement si $z \le a_p t + b_p$ pour tout $1 \le p \le |\alpha|$ et (t, z) vérifie les contraintes de notre instance LP primale présentée dans la preuve du théorème 8.2, appliquée à la formule PMOO. Il suffit de remplacer la fonction objectif par la contrainte $z \ge \sum_{i=1}^{k} \sum_{\ell=1}^{\ell_i} r_{i,\ell} v_{i,\ell}$ après avoir appliqué les transformations de cette preuve. Alors $D_{\max}(\alpha, \psi) = \sup\{t' - t \mid (t, z) \in C(\alpha, \psi),\ (t', z) \in C(\alpha, \psi),\ z \in \mathbb{R}_+\}$ et $B_{\max}(\alpha, \psi) = \sup\{z' - z \mid (t, z) \in C(\alpha, \psi),\ (t, z') \in C(\alpha, \psi),\ t \in \mathbb{R}_+\}$. Dans les deux cas, la valeur considérée est l'optimum d'une instance LP avec moins de $2(2n + k + 2 + |\alpha| + \sum |\alpha_i| + \sum |\beta_j|)$ contraintes sur moins de $2(n + \sum |\alpha_i| + \sum |\beta_j|) + 3$ variables.

Si tous les services sont stricts, prendre $\phi = \psi_+$ à la place de ψ donne des bornes plus *tight* et rajoute une seule contrainte linéaire. ■

Remarquons que, dans certains cas simples en Network Calculus (par exemple seau percé ou débit-latence), on suppose que $|\alpha_i|$ et $|\beta_j|$ sont égaux à 1 ou à 2, ce qui donne de petites instances LP.

Remarque 8.1. *L'objectif premier du Network Calculus est de fournir des bornes sur les mesures de performance pire-cas mais trouver des bornes* tight *est un problème difficile. Les premières analyses prenant en compte le phénomène PMOO ont soulevé la question pour*

les configurations en tandem que nous étudions ici [Schmitt 2006a, Schmitt 2006b]. Les travaux [Bouillard 2007a, Bouillard 2008a] étudiant la formule PMOO du théorème 8.1 ont montré sur un petit exemple que dans des cas particuliers la courbe de service de bout en bout $\phi = \psi_+$ pouvait donner des bornes sur le délai et la charge moins bonnes que d'autres courbes de service aussi fournies par le chemin, mais mieux adaptées à la courbe d'arrivée du flux principal. En fait, le travail très minutieux de [Schmitt 2007] suggère que cette question est plus compliquée que prévu, et requiert le développement de nouvelles méthodes pour obtenir des bornes tight, *quitte à perdre une partie de l'aspect algébrique qui est l'essence du Network Calculus. Néanmoins, l'utilisation de notre formule PMOO donne de meilleures bornes que les approches précédentes comme le montrent les expériences dans [Schmitt 2006a, Schmitt 2006b, Bouillard 2008a]. De plus, d'après le théorème 8.3, ces bornes peuvent être calculées en temps polynomial, là où d'autres solutions requièrent un temps exponentiel ($\prod_{i=1}^{k} |\alpha_i| \prod_{j=1}^{n} |\beta_j|$ instances LP à résoudre dans [Schmitt 2007]).*

8.3. Cas particulier : la convolution (min,+) classique

Dans cette section, nous nous intéressons aux configurations sans trafic transverse sur le chemin : cela revient à un calcul classique de convolution (min,+). Nous rappelons un théorème classique de NC — dont la preuve sera étudiée dans le chapitre 10 — à propos de la convolution de fonctions convexes affines par morceaux. Puis nous étudions la complexité du calcul des bornes sur les délais et les charges dans ce cas particulier.

Convolution de fonctions convexes affines par morceaux

Théorème 8.4 ([Le Boudec 2001])**.** *Soit f et g deux fonctions convexes affines par morceaux définies sur $\mathbb{R}_+$, avec un nombre fini de morceaux. Soit ρ_f (resp. ρ_g) la pente du dernier segment semi-infini de f (resp. de g) s'il en existe un ; ou bien $\rho_f = +\infty$ (resp. $\rho_g = +\infty$) si f (resp. g) est égal à $+\infty$ à partir d'un certain point. Alors la convolution $f * g$ consiste tout d'abord en la concaténation, par ordre de pente croissant et en commençant en $f(0) + g(0)$, de tous les segments de f et g de pente $< \min(\rho_f, \rho_g)$; puis termine par la concaténation d'un segment semi-infini de pente $\min(\rho_f, \rho_g)$ si cette valeur est finie.*

Ce théorème est étudié plus particulièrement au chapitre 10.

Bornes sur les délais et les charges

Théorème 8.5. *Dans les configurations PMOO simples sans trafic transverse, une courbe de service simple pour l'ensemble du chemin est $\beta = \beta_1 * \beta_2 * \cdots * \beta_n$. Si toutes les courbes de service sont convexes et affines par morceaux avec un nombre fini de morceaux, alors la taille de ψ vérifie $|\psi| \leq \sum_{j=1}^{n} |\beta_j|$ et ψ peut être calculé en temps $O(\log_2 n \times \sum_{j=1}^{n} |\beta_j|)$.*

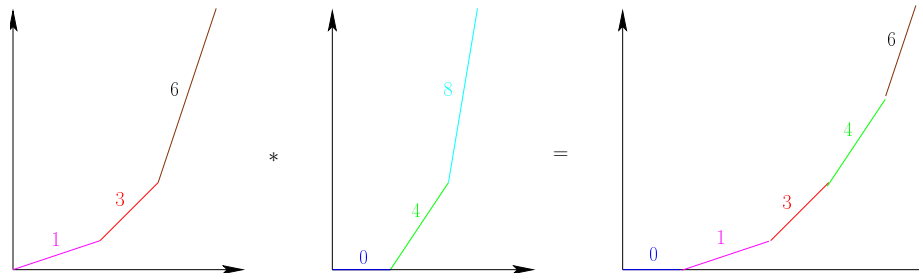


FIGURE 8.2 : Convolution de deux fonctions convexes affines par morceaux (théorème 8.4)

Etant donnée une courbe d'arrivée α concave et affine par morceaux avec un nombre fini de morceaux, si les morceaux de α (resp. ψ) sont stockés dans un tableau trié selon les abscisses, alors on peut calculer $D_{max}(\alpha,\beta)$ *et* $B_{max}(\alpha,\beta)$ *en temps* $\mathcal{O}(\log^2(|\alpha|+|\beta|))$.

Preuve. D'après le théorème 8.4, le calcul de $\beta = \beta_1 * \beta_2 * \cdots * \beta_n$ revient à fusionner n listes triées de taille respective $|\beta_j|$. La complexité annoncée dans le théorème peut être obtenue en progressant de gauche à droite dans les listes (au sens des abscisses) tout en maintenant un tas binaire qui contient le premier morceau disponible de chaque liste et permet de trouver efficacement le segment de plus petite pente [Cormen 2001].

Pour calculer la borne $B_{max}(\alpha,\beta)$, on peut utiliser une sorte de dichotomie. Deux pointeurs $l_\alpha \le r_\alpha$ (resp. $l_\beta \le r_\beta$) sont associés avec chacun des tableaux triés : ils pointent donc sur deux morceaux de α et deux morceaux de β et indiquent les extrémités de l'intervalle où la distance verticale maximum est atteinte. Ces pointeurs se déplaceront pendant la recherche de ce maximum, et sont initialisés aux extrémités de chacun des tableaux : on a donc $l_\alpha = 1$, $r_\alpha = |\alpha|$, $l_\beta = 1$ et $r_\beta = |\beta|$. Une étape élémentaire de l'algorithme sera décomposée comme suit :

1. choisir, parmi α et β, la fonction qui maximise $\max(r_\alpha - l_\alpha, r_\beta - l_\beta)$. Supposons par exemple que c'est α (le cas de β est symétrique) ;
2. prendre le morceau médian entre l_α et r_α, c'est-à-dire le morceau d'indice $\lceil l_\alpha + r_\alpha \rceil/2$, et rechercher le ou les morceaux de β à la verticale de chacune des extrémités de ce morceau en utilisant une recherche dichotomique dans le tableau trié de β ;
3. comparer les pentes du morceau médian de α et du ou des morceaux de β à la verticale de ses extrémités (il faudra dans certains cas vérifier les pentes des morceaux adjacents), pour décider si la distance verticale maximum est à gauche ou à droite, ou à la verticale du morceau médian de α. Les pointeurs de α peuvent être mis à jour en conséquence, tout comme les pointeurs de β, étant donné que nous avons déjà calculé les indices des segments à la verticale du morceau médian de α.

Chaque étape élimine au moins 1/4 des morceaux de α et de β, et chaque recherche dichotomique s'exécute en temps $\mathcal{O}(\log_2(|\alpha|+|\beta|))$. La complexité globale en temps de cet algorithme est donc bornée par $\mathcal{O}(\log_2(|\alpha|+|\beta|)\log_{3/4}(|\alpha|+|\beta|))$.

Le même type d'algorithme s'applique au calcul de $D_{max}(\alpha,\beta)$. ■

CHAPITRE 9

ÉTUDE DES RÉSEAUX SANS DÉPENDANCES CYCLIQUES : OPTIMISATION LINÉAIRE

Le but de ce chapitre est l'analyse pire-cas exacte de réseaux sans dépendances cycliques. Nous utilisons pour cela une méthode utilisant la résolution d'un ensemble de problèmes d'optimisation linéaire.

Une étude des réseaux en tandem [Schmitt 2006a] a souligné un nouveau phénomène appelé *Pay Multiplexing Only Once (PMOO)* : la compétition, entre les différents flux, pour les ressources est amortie sur l'ensemble des serveurs. Cet article présentait une nouvelle méthode prenant en compte le PMOO, et les expériences montraient une amélioration significative sur les bornes des délais de bout en bout par rapport aux approches NC précédentes.

Cette méthode peut s'exprimer comme une nouvelle multi-convolution dans $(\min, +)$ [Bouillard 2008a], qui préserve l'esprit du NC tout en étant un bon candidat pour l'analyse exacte de réseau avec multiplexage aveugle. Cependant, l'article majeur [Schmitt 2008b] a

montré que ces bornes pouvaient être arbitrairement éloignées des valeurs pire-cas exactes, même dans des réseaux acycliques apparemment simples (deux flux et deux serveurs). Cet article suggérait une nouvelle approche par optimisation linéaire, mais seul un réseau à trois flux et trois serveurs, ainsi que des réseaux *sink-tree* en tandem, avaient pu être analysés de manière exacte.

Ce chapitre décrit le premier algorithme qui calcule le délai bout en bout maximal pour un flux, ainsi que la charge maximale pour un serveur, pour tout réseau acyclique en multiplexage aveugle, avec des courbes d'arrivée concaves et des courbes de service convexes. If fournit aussi une trajectoire critique pour le système, qui donne la valeur pire-cas.

Sa complexité peut sembler très élevée (potentiellement super-exponentielle), mais le problème est intrinsèquement difficile (NP-difficile, voir [Bouillard 2010]). Heureusement dans certains cas comme les réseaux en tandem, c'est-à-dire les scénarios étudiés dans [Schmitt 2008b, Schmitt 2006a], la complexité devient polynomiale.

En plus du fait que notre solution s'applique à tout réseau acyclique, et bien qu'elle utilise aussi l'optimisation linéaire, plusieurs propriétés distinguent notre approche de celle de [Schmitt 2008b] : nous étudions aussi bien le délai des flux que la charge des serveurs, nous calculons directement les pire-cas sans calculer de courbe de service de bout en bout, nous évitons la décomposition/recomposition de courbes convexes/concaves qui pourrait empirer les bornes obtenues.

La suite est organisée ainsi : après une présentation du modèle de réseau utilisé et des principales notions de NC dans la section 9.1, nous décrivons puis analysons notre algorithme dans la section 9.2. La section 9.3 montre comment cette méthode s'applique au réseaux en tandem et compare notre solution aux travaux précédents [Schmitt 2008b, Schmitt 2007]. Les expérimentations sont étudiées dans la section 9.4 pour montrer le gain par rapport aux précédentes méthodes NC.

D'autres extensions intéressantes et problèmes ouverts sont présentés dans la section 9.4.2. Nos résultats s'appliquent au Network Calculus et à ses extensions, comme le Real-Time Calculus [Thiele 2001] (qui utilise des courbes de service strict). Mais pas à ses extensions stochastiques [Fidler 2006b, Jiang 2008].

9.1. Modélisation du réseau

Sans perte de généralité, un réseau sera modélisé par un graphe orienté. Les flux qui traversent le réseau doivent suivre les arêtes du graphe ; et les serveurs (switches, liens de transmission, routeurs...) sont représentés par les noeuds du graphe.

Les serveurs et les flux seront identifiés par des indices. L'ensemble des serveurs est $\mathbb{S} = \{1, \ldots, n\}$ et chaque serveur j offre un service *strict* $\beta_j \in \mathcal{F}_{\nearrow}$ qui est affine par morceaux (avec

un nombre fini $|\beta_j|$ de morceaux) et convexe. Et β peut donc s'écrire $\beta_j(t) = \max_{1\leq\ell\leq|\beta_j|}(r_{j,\ell}t - h_{j,\ell})$ avec $r_{j,\ell}, h_{j,\ell} \in \mathbb{R}_+$.

L'ensemble des flux est $\mathbb{F} = \{1,\dots,p\}$. Chaque flux i correspond au couple (α_i, μ_i) où μ_i est la suite ordonnée (et finie) des serveurs traversés par le flux, et α_i est la courbe d'arrivée qui modèle le trafic du flux i avant d'entrer dans le premier serveur.

Par hypothèse, α_i est positive, croissante, affine par morceaux (avec un nombre fini $|\alpha_i|$ de morceaux) et concave. Rq : quitte à imposer une valeur nulle en $t = 0$, elle appartient à $\mathcal{F}_\nearrow$, cependant nous n'imposons pas cette contrainte afin de l'exprimer comme le minimum de fonctions affines, ce qui sera utile pour exprimer les contraintes en optimisation linéaire.

On peut donc écrire $\alpha_i(t) = \min_{1\leq\ell\leq|\alpha_i|}(\sigma_{i,\ell} + \rho_{i,\ell}t)$ avec $\sigma_{i,\ell}, \rho_{i,\ell} \in \mathbb{R}_+$. Soit $i \in \mathbb{F}$, nous notons first(i) (resp. last(i)) l'indice du premier (resp. dernier) serveur rencontré par le flux i.

Remarque 9.1. *Par abus de notation, nous noterons $j \in i$ pour dire que le serveur j appartient à la suite μ_i. Pour $j \in i$, nous noterons $prec_i(j)$ l'indice du serveur qui précède j dans la suite μ_i (par convention, $prec_i(first(i)) = 0$). Ces notations sont illustrées dans la figure 9.1.*

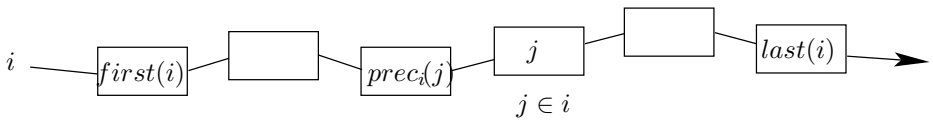


Figure 9.1 : Notations pour le flux i.

Le réseau global $\mathcal{N}$ est défini par $\mathbb{S}$, $\mathbb{F}$ et les ensembles $\{\beta_j,\ 1 \leq j \leq n\}$, $\{(\alpha_i, \mu_i),\ 1 \leq i \leq p\}$. Le graphe orienté induit par $\mathcal{N}$ est $\mathcal{G}(\mathcal{N}) = (\mathbb{S}, \mathbb{A})$, où $\mathbb{S}$ est l'ensemble des sommets et $(j, j') \in \mathbb{A}$ si et seulement si j et j' sont deux serveurs consécutifs pour une certaine suite μ_i. Les suites μ_i sont des chemins dans le graphe orienté (la réciproque n'est pas vraie).

Chaque chemin dans $\mathcal{G}(\mathcal{N}) = (\mathbb{S}, \mathbb{A})$ sera désigné par la suite ordonnée de sommets, et souvent écrit comme un *mot sur l'alphabet* $\mathbb{S}$.

Nous utiliserons souvent π pour désigner un chemin et, par exemple, $j\pi$ représentera le chemin commençant par le serveur j, suivi du chemin π. Nous utiliserons les notations suivantes, similaires à celle présentes dans [Schmitt 2008b].

Pour tout $i \in \mathbb{F}$,

- la fonction cumulée du flux i à l'entrée du réseau est $F_i^{(0)}$;
- pour tout $j \in i$, la fonction cumulée du flux i à la sortie du serveur j est $F_i^{(j)}$.

Un ensemble de fonctions cumulées $\{F_i^{(j)} \in \mathcal{F}_\nearrow \mid i \in \mathbb{F},\ j \in \mathbb{S},\ j \in i\}$ sera appelé *trajectoire du réseau $\mathcal{N}$* si elle respecte les contraintes NC du réseau :

(T1) $\forall i \in \mathbb{F}, \forall j \in i, F_i^{(\text{prec}_i(j))} \geq F_i^{(j)}$;

(T2) $\forall i \in \mathbb{F}, F_i^{(0)}$ est contrainte supérieurement par α_i ;

(T3) $\forall j \in \mathbb{S}$, $(\sum_{i \ni j} \mathrm{F}_i^{(\mathrm{prec}_i(j))}, \sum_{i \ni j} \mathrm{F}_i^{(j)}) \in \mathcal{S}_{strict}(\beta_j)$.

Étant donné que nous travaillons en *blind multiplexing*, il n'y a pas d'autres contraintes. L'ensemble des trajectoires de $\mathcal{N}$ sera noté $\mathrm{Traj}(\mathcal{N})$.

Ci-dessous, les fonctions objectifs que nous considérons :

1. Pire délai de bout en bout. Étant donné un flux i_0, nous souhaitons calculer le pire délai de bout en bout subi par des données dans ce flux, c'est-à-dire :

$$\sup_{\{\mathrm{F}_i^{(j)}\} \in \mathrm{Traj}(\mathcal{N})} \sup_{0 \le s \le t} \{t - s \mid \mathrm{F}_{i_0}^{(0)}(s) > \mathrm{F}_{i_0}^{(\mathrm{last}(i_0))}(t)\}.$$

2. Charge locale pire-cas. Étant donné un serveur j_0, nous souhaitons calculer la pire charge possible au niveau de ce serveur, c'est-à-dire :

$$\sup_{\{\mathrm{F}_i^{(j)}\} \in \mathrm{Traj}(\mathcal{N})} \sup_{t \ge 0} \{\sum_{i \ni j_0} \mathrm{F}_i^{(\mathrm{prec}_i(j_0))}(t) - \sum_{i \ni j_0} \mathrm{F}_i^{(j_0)}(t)\}.$$

Ces supremums sont calculés sur des ensembles infinis, néanmoins ils peuvent être calculés comme expliqué ci-dessous.

Remarque 9.2. *Ces supremums ne sont pas nécessairement atteints pour une trajectoire donnée, ou pour des instants s, t. Par exemple, avec un unique flux traversant un unique serveur, soit $\mathrm{F}_1^{(0)}(t) = t$ et $\mathrm{F}_1^{(1)}(t) = 2(t-1)_+$. Alors le pire délai est 1, mais il n'est atteint en aucun instant s ou t (le pire cas se produit lorsque $t = 1$ et s tend vers 0). De même, si $\mathrm{F}_1^{(0)}(0) = 0$, $\mathrm{F}_1^{(0)}(t) = 1$ pour $t > 0$ et $\mathrm{F}_1^{(1)}(t) = t$, la pire charge est 1, mais elle n'est jamais atteinte (il faut faire tendre t vers 0).*

9.2. Analyse des réseaux sans dépendances cycliques

Remarque 9.3. *Dans la suite nous construisons un ensemble de* problèmes d'optimisation linéaire, *que nous appellerons aussi* programmes linéaires, *parfois noté* LP.

Théorème 9.1. *Soit $\mathcal{N}$ un réseau composé de n serveurs et p flux. Si son graphe induit $\mathcal{G}(\mathcal{N})$ est acyclique, alors pour tout flux i (resp. tout serveur j), il existe un ensemble fini Λ de* programmes linéaires *dont les optimums respectifs sont les opt_λ, $\lambda \in \Lambda$, de telle sorte que $\max_{\lambda \in \Lambda} opt_\lambda$ est le pire délai de bout en bout pour le flux i (resp. la pire charge au serveur j). Chaque programme linéaire a $\mathcal{O}(p|\Pi|)$ variables et $\mathcal{O}(p|\Pi|^2)$ contraintes linéaires où Π est l'ensemble des chemins qui se terminent par le noeud $last(i)$ (resp. j). On a alors $|\Pi| \le 2^{n-1}$ et $|\Lambda| \le |\Pi|! 2^{|\Pi|-1}$.*

Remarque 9.4. *La description des différentes instances LP ainsi que la preuve du théorème seront illustrées à l'aide d'un exemple petit, mais classique, le* réseau en diamant, *voir figure 9.2.*

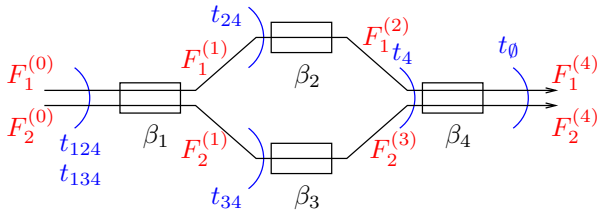


FIGURE 9.2 : Réseau en diamant : deux flux et quatre serveurs.

9.2.1. Les instances LP

Remarque 9.5. *Nous considérons ici un ensemble de* problème d'optimisation linéaire *(*linear programming*). Les différents problèmes générés sont appelés* instances LP.

Nous présentons ici un ensemble d'instances LP, de telle sorte que chaque trajectoire du réseau satisfasse au moins l'une d'entre elles.

Variables

Soit i le flux observé (resp. j le serveur observé) : on souhaite calculer le pire délai de bout en bout (resp. la pire charge) pour ce flux (resp. ce serveur).

Soit Π l'ensemble des chemins qui se terminent au noeud last(i) (resp. au noeud j), donc le chemin vide $\emptyset$.

Voici les noms de variables qui apparaîtront dans les différentes instances LP :

- t_π pour tout $\pi \in \Pi$.
  Interprétation : $t_\emptyset$ est l'instant auquel le pire cas se produit (sortie des données qui subissent le pire délai ou la pire charge). Alors, pour tout $j\pi \in \Pi$, $t_{j\pi} = start_j(t_\pi)$, le début de la période chargée de t_π au serveur j.
- $\mathrm{F}_i^{(j)}(t_\pi)$ pour tout $i \in \mathbb{F}$, $j \in i$, $\pi \in \Pi_i^j = \{\pi', j\pi' \mid j\pi' \in \Pi\}$.
  Interprétation : la valeur de la fonction cumulée $\mathrm{F}_i^{(j)}$ au temps t_π.
- $\mathrm{F}_i^{(0)}(t_\pi)$ pour tout $i \in \mathbb{F}$, $\pi \in \Pi_i = \bigcup_{j\in i} \Pi_i^{(j)}$.
  Interprétation : la valeur de la fonction cumulée $\mathrm{F}_i^{(0)}$ au temps t_π.

Exemple : diamant

- Si nous étudions le pire délai de bout en bout pour le flux 1 (ou le flux 2, ou la pire charge au serveur 4), nous devons considérer l'ensemble des chemins qui se terminent au serveur 4, c'est-à-dire $\Pi = \{\emptyset, 4, 24, 34, 124, 134\}$. Les variables temporelles sont donc $t_\emptyset$, t_4, t_{24}, t_{34}, t_{124}, t_{134}.
- Pour le flux 1 et le serveur 1, l'ensemble $\Pi_1^{(1)}$ est $\{124, 24, 134, 34\}$. Les variables correspondantes sont $\mathrm{F}_1^{(1)}(t_{124})$, $\mathrm{F}_1^{(1)}(t_{24})$, $\mathrm{F}_1^{(1)}(t_{134})$, $\mathrm{F}_1^{(1)}(t_{34})$.
- Pour le flux 1, nous avons $\Pi_1 = \{\emptyset, 4, 24, 34, 124, 134\}$. Les variables correspondant à l'entrée sont donc $\mathrm{F}_1^{(0)}(t_{124})$, $\mathrm{F}_1^{(0)}(t_{24})$, $\mathrm{F}_1^{(0)}(t_{134})$, $\mathrm{F}_1^{(0)}(t_{34})$, $\mathrm{F}_1^{(0)}(t_4)$, $\mathrm{F}_1^{(0)}(t_\emptyset)$.

Remarque 9.6. *On aurait pu introduire les variables* $\mathrm{F}_i^{(j)}(t_\pi)$ *pour tous les* $\pi \in \Pi$, *mais la plupart de ces variables n'interviennent pas dans la description du déroulement des événements qui aboutissent au pire cas, et ne sont pas nécessaires pour reconstruire les trajectoires critiques.*

Remarque 9.7. *Dans la suite, nous distinguerons les expressions « la variable* x *» qui désigne la variable non affectée, et « la valeur* x *», qui désigne la valeur affectée à la variable* x.

Contraintes temporelles

Un *ensemble de contraintes temporelles* $\mathcal{T}$ sur un sous-ensemble $\Pi' \subseteq \Pi$ est un ensemble d'égalités et d'inégalités de la forme $t_{\pi_1} = t_{\pi_2}$, $t_{\pi_1} \leq t_{\pi_2}$ ou $t_{\pi_1} < t_{\pi_2}$ où $\pi_1, \pi_2 \in \Pi'$, et tel que son ensemble de solutions $\mathrm{Sol}(\mathcal{T}) \subseteq \mathbb{R}_+^{\Pi'}$ est non vide.

Pour assurer la cohérence des valeurs t_π avec leur interprétation comme débuts de périodes chargées, nous introduisons deux prédicats sur les variables t_π :

(P1) Pour tout $j\pi \in \Pi$, $t_{j\pi} \leq t_\pi$.

(P2) pour tout $j\pi_1, j\pi_2 \in \Pi$, $t_{j\pi_1} < t_{j\pi_2} \implies t_{j\pi_1} \leq t_{\pi_1} < t_{j\pi_2} \leq t_{\pi_2}$.

Le prédicat (P1) vient du fait que pour toute trajectoire et tout instant t, au serveur j, nous avons $start(t) \leq t$. De même pour tout t, t', au serveur j, on ne peut pas avoir $start(t) < start(t') \leq t$ (la période chargée de t' commencerait pendant celle de t), d'où le prédicat (P2).

Un ensemble de contraintes temporelles $\mathcal{T}$ sur Π' satisfait les prédicats (P1) et (P2) si toute solution pour $\mathcal{T}$ à valeurs réelles satisfait à la fois (P1) et (P2).

Cette définition fait intervenir un nombre *a priori* infini de tests, mais on peut décider si $\mathcal{T}$ satisfait (P1) et (P2) en temps $\mathcal{O}(|\mathcal{T}||\Pi'|)$: on peut obtenir toutes les comparaisons possibles entre les variables t_π, $\pi \in \Pi'$ par clôture transitive de $\mathcal{T}$ (complexité : $\mathcal{O}(|\mathcal{T}||\Pi'|)$). Vérifier (P1) est alors immédiat, et vérifier (P2) revient à vérifier qu'il n'y a pas de configuration $t_{j\pi_1} < t_{j\pi_2} \leq t_{\pi_1}$, ce qui peut être fait en temps $\mathcal{O}(|\mathcal{T}|^2)$.

Un ensemble de contraintes temporelles $\mathcal{T}$ est un *ordre total* sur un sous-ensemble $\Pi' \subseteq \Pi$ s'il est de la forme $\{t_{\pi_1} \lhd_1 t_{\pi_2} \lhd_2 \cdots \lhd_{N-1} t_{\pi_N}\}$ où $\pi_1, \pi_2, \ldots, \pi_N$ est une permutation de tous les éléments de Π' et pour tout $1 \leq k \leq N-1$, $\lhd_k \in \{=, <, \leq\}$.

Pour tout $\pi, \pi' \in \Pi'$, en considérant la clôture transitive, $\mathcal{T}$ impose une comparaison entre π et π' qui sera forcément $=, <, >, \leq$ ou $\geq$. Cette comparaison sera notée $\mathcal{T}(\pi, \pi')$.

L'ensemble de tous les ordres sur Π' qui satisfont (P1) et (P2) est noté $\mathrm{Tot}(\Pi')$. On peut énumérer tous les éléments de $\mathrm{Tot}(\Pi')$ de la manière suivante : générer l'ensemble des contraintes temporelles qui vérifient (P1), ce qui correspond à un ordre partiel arborescent, puis générer toutes ses extensions linéaires, générer pour chaque extension linéaire toutes les combinaisons possibles de comparaisons $=$, $<$ ou $\leq$, et vérifier pour chacune si elle satisfait (P2). Cet algorithme marche, avec une complexité de $\mathcal{O}(|\mathrm{Tot}(\Pi')| 3^{|\Pi'|} |\Pi'|^2)$ [Pruesse 1994]. Nous n'avons pas recherché d'algorithme plus ra-

pide ; il y a probablement d'autres façons d'accélérer cette étape, mais tout algorithme sera au moins d'une complexité $|\mathrm{Tot}(\Pi')|$ qui peut être exponentiel en $|\Pi'|$.

Nous étudions maintenant notre réseau. À chaque flux i, nous avons associé l'ensemble des chemins $\Pi_i = \{\pi, j\pi \mid j \in i,\ j\pi \in \Pi\}$ et nous savons que $\Pi = \bigcup_{1\leq i\leq n} \Pi_i$. Soit $(\mathcal{T}_1, \ldots, \mathcal{T}_p) \in \mathrm{Tot}(\Pi_1)\times\cdots\times\mathrm{Tot}(\Pi_p)$; on dit que les ordres totaux $(\mathcal{T}_1, \ldots, \mathcal{T}_p)$ sont *mutuellement compatibles* si pour tout $1 \leq i_1, i_2 \leq p$ et $\pi, \pi' \in \Pi_{i_1} \cap \Pi_{i_2}$, on a $\mathcal{T}_{i_1}(\pi, \pi') = \mathcal{T}_{i_2}(\pi, \pi')$. Ceci peut être vérifié avec un algorithme en $\mathcal{O}(p|\Pi|^2)$ pour toutes les paires π, π'. Cette condition assure qu'il existe une solution $(t_\pi)_{\pi\in\Pi} \in \mathbb{R}_+^\Pi$ pour l'ensemble des contraintes $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_p$. De plus, cette solution satisfera les prédicats (P1) et (P2).

Chaque combinaison $(\mathcal{T}_1, \ldots, \mathcal{T}_p) \in \mathrm{Tot}(\Pi_1) \times \cdots \times \mathrm{Tot}(\Pi_p)$ d'ordres totaux mutuellement compatibles nous donnera un ensemble d'instances LP. Le principal problème, qui nous conduit à faire une distinction de cas, est que contrairement à (P1), le prédicat (P2) ne peut pas être exprimé par un seul ensemble de contraintes linéaires.

Pour éviter les cas redondants, on utilisera des ensembles de contraintes $\mathcal{T}_i$ telles que leur ensemble de solutions $\mathrm{Sol}(\mathcal{T}_i)$ est maximal au sens de l'inclusion (par exemple, parmi deux ensembles $\mathcal{T}^1 = \{t_{12} = t_{13} < t_2 = t_3\}$ et $\mathcal{T}^2 = \{t_{12} = t_{13} \leq t_2 \leq t_3\}$, il suffit d'étudier $\mathcal{T}^2$).

Exemple : diamant

On considère l'ensemble $\Pi = \{\emptyset, 4, 24, 34, 124, 134\}$ et $\Pi_1 = \Pi_2 = \Pi$.

Pour satisfaire le prédicat (P1), on a les relations :

$$t_{124} \leq t_{24} \leq t_4 \leq t_\emptyset \text{ et } t_{134} \leq t_{34} \leq t_4 \leq t_\emptyset.$$

De plus, il faut ordonner t_{124} et t_{134}. Il y a quatre ordres totaux maximaux qui satisfont aussi le prédicat (P2) :

- $\mathcal{T}^1 = \{t_{124} \leq t_{24} < t_{134} \leq t_{34} \leq t_4 \leq t_\emptyset\}$;
- $\mathcal{T}^2 = \{t_{134} \leq t_{34} < t_{124} \leq t_{24} \leq t_4 \leq t_\emptyset\}$;
- $\mathcal{T}^3 = \{t_{124} = t_{134} \leq t_{24} \leq t_{34} \leq t_4 \leq t_\emptyset\}$;
- $\mathcal{T}^4 = \{t_{124} = t_{134} \leq t_{34} \leq t_{24} \leq t_4 \leq t_\emptyset\}$.

Contraintes de trajectoires

Soit $(\mathcal{T}_1, \ldots, \mathcal{T}_p) \in \mathrm{Tot}(\Pi_1) \times \cdots \times \mathrm{Tot}(\Pi_p)$ des ordres totaux mutuellement compatibles.

Voici l'ensemble des égalités et inégalités qui forment les contraintes sur les états du système pour les événements qui nous intéressent :

- *Contraintes temporelles :*

$$\mathcal{T} = \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_p$$

- *Contraintes de service strict* : pour tout $j \in \mathbb{S}$ et $j\pi \in \Pi$, on ajoute les contraintes

$$\{\sum_{i \ni j} \mathrm{F}_i^{(j)}(t_\pi) - \sum_{i \ni j} \mathrm{F}_i^{(j)}(t_{j\pi}) \geq \beta_j(t_\pi - t_{j\pi})\}$$

  (c'est-à-dire $|\beta_j|$ inégalités linéaires car β_j est un maximum de fonctions affines). De plus, pour tout $j\pi_1, j\pi_2 \in \Pi$ tel que $\mathcal{T}(j\pi_1, j\pi_2) \in \{=\}$ et $\mathcal{T}(\pi_1, \pi_2) \in \{=, \leq, <\}$, on ajoute les contraintes

$$\{\sum_{i \ni j} \mathrm{F}_i^{(j)}(t_{\pi_2}) - \sum_{i \ni j} \mathrm{F}_i^{(j)}(t_{\pi_1}) \geq \beta_j(t_{\pi_2} - t_{\pi_1})\}$$

- *Débuts des périodes chargées :* pour tout $j \in \mathbb{S}$, $j\pi \in \Pi$ et $i \ni j$, on ajoute les contraintes

$$\{\mathrm{F}_i^{(\mathrm{prec}_i(j))}(t_{j\pi}) = \mathrm{F}_i^{(j)}(t_{j\pi})\}$$

- *Contraintes de flux :* pour tout $i \in \mathbb{F}$, $j \in i$ et $j\pi \in \Pi$, on ajoute

$$\{\mathrm{F}_i^{(0)}(t_{j\pi}) \geq \mathrm{F}_i^{(j)}(t_{j\pi}),\ \mathrm{F}_i^{(0)}(t_\pi) \geq \mathrm{F}_i^{(j)}(t_\pi)\}$$

- *Fonctions croissantes :* pour tout $i \in \mathbb{F}$, $j \in i$ et $\pi_1, \pi_2 \in \Pi_i^{(j)}$, si $\mathcal{T}(\pi_1, \pi_2) \in \{=\}$, alors on ajoute

$$\{\mathrm{F}_i^{(j)}(t_{\pi_1}) = \mathrm{F}_i^{(j)}(t_{\pi_2})\}$$

  et si $\mathcal{T}(\pi_1, \pi_2) \in \{\leq, <\}$, alors on ajoute

$$\{\mathrm{F}_i^{(j)}(t_{\pi_1}) \leq \mathrm{F}_i^{(j)}(t_{\pi_2})\}$$

- *Contrainte d'arrivée :* pour tout $1 \leq i \leq p$, et $\pi_1, \pi_2 \in \Pi_i$ tel que $\mathcal{T}(\pi_1, \pi_2) \in \{=, \leq, <\}$, on ajoute

$$\{\mathrm{F}_i^{(0)}(t_{\pi_2}) - \mathrm{F}_i^{(0)}(t_{\pi_1}) \leq \alpha_i(t_{\pi_2} - t_{\pi_1})\}$$

  (c'est-à-dire $|\alpha_i|$ inégalités linéaires car α_i est un minimum de fonctions affines).

Exemple : diamant

Pour, par exemple, $\mathcal{T}^1 = \{t_{124} \leq t_{24} < t_{134} \leq t_{34} \leq t_4 \leq t_\emptyset\}$, la figure 9.3 donne les contraintes sur les flux, sur les fonctions croissantes et sur les débuts de périodes chargées.

Pour le serveur 1, les contraintes de service strict sont :

- $(\mathrm{F}_1^{(1)}(t_{24}) + \mathrm{F}_2^{(1)}(t_{24})) - (\mathrm{F}_1^{(1)}(t_{124}) + \mathrm{F}_2^{(1)}(t_{124})) \geq \beta_1(t_{24} - t_{124})$,
- $(\mathrm{F}_1^{(1)}(t_{34}) + \mathrm{F}_2^{(1)}(t_{34})) - (\mathrm{F}_1^{(1)}(t_{134}) + \mathrm{F}_2^{(1)}(t_{134})) \geq \beta_1(t_{34} - t_{134})$.

Pour le flux 1, les contraintes d'arrivée sont :

- $\mathrm{F}_1^{(0)}(t_{24}) - \mathrm{F}_1^{(0)}(t_{124}) \leq \alpha_1(t_{24} - t_{124})$,
- $\mathrm{F}_1^{(0)}(t_{134}) - \mathrm{F}_1^{(0)}(t_{124}) \leq \alpha_1(t_{134} - t_{124})$,

$$
\begin{array}{ccccccccccc}
F_1^0(t_{124}) & \leq & F_1^0(t_{24}) & \leq & F_1^0(t_{134}) & \leq & F_1^0(t_{34}) & \leq & F_1^0(t_4) & \leq & F_1^0(t_\emptyset) \\
\| & & \wedge & & \| & & \wedge & & \wedge & & \wedge \\
F_1^1(t_{124}) & \leq & F_1^1(t_{24}) & \leq & F_1^1(t_{134}) & \leq & F_1^1(t_{34}) & & & & \\
 & & \| & & & & & & F_1^2(t_4) & & \\
 & & F_1^2(t_{24}) & \leq & & & & & & & \\
 & & & & & & & & \| & & \\
 & & & & & & & & F_1^4(t_4) & \leq & F_1^4(t_\emptyset)
\end{array}
$$

FIGURE 9.3 : Réseau en diamant : contraintes pour le flux 1 et $\mathcal{T}^1$ (sauf les contraintes de service et d'arrivée)

- $\mathrm{F}_1^{(0)}(t_{134}) - \mathrm{F}_1^{(0)}(t_{24}) \leq \alpha_1(t_{134} - t_{24})$, ...

9.2.2. Objectifs

Pire délai de bout en bout pour le flux i_0

On maximise la fonction objectif $(t_\emptyset - u)$, où $t_\emptyset - u$ est le délai subi par les données qui sont entrées dans le réseau à l'instant u et sorties à l'instant $t_\emptyset$. Il faut par conséquent ajouter plusieurs contraintes liées à u, et éventuellement faire une distinction de cas selon le choix de $\mathcal{T}_{i_0}$ dans $\mathrm{Tot}(\Pi_{i_0})$:

- *Insertion :* il faut inclure u dans l'ordre total $\mathcal{T}_{i_0}$. Pour chaque $\pi \in \Pi_{i_0}$, on génère une instance LP différente en ajoutant les contraintes :
  - *Position et monotonie :* on ajoute $\{t_\pi \leq u,\ \mathrm{F}_{i_0}^{(0)}(t_\pi) \leq \mathrm{F}_{i_0}^{(0)}(u)\}$, et pour $t_{\pi'}$ le successeur de t_π dans $\mathcal{T}_{i_0}$ (s'il y en a un), on ajoute $\{u \leq t_{\pi'},\ \mathrm{F}_{i_0}^{(0)}(u) \leq \mathrm{F}_{i_0}^{(0)}(t_{\pi'})\}$.
  - *Contraintes de courbe d'arrivée :* pour tout $\pi' \in \Pi_{i_0}$, si $\mathcal{T}(\pi', \pi) \in \{=, \leq, <\}$, on ajoute $\{\mathrm{F}_{i_0}^{(0)}(u) - \mathrm{F}_{i_0}^{(0)}(t_{\pi'}) \leq \alpha_{i_0}(u - t_{\pi'})\}$; sinon, on ajoute $\{\mathrm{F}_{i_0}^{(0)}(t_{\pi'}) - \mathrm{F}_{i_0}^{(0)}(u) \leq \alpha_{i_0}(t_{\pi'} - u)\}$.
- *Instant d'arrivée :* $\{\mathrm{F}_{i_0}^{(0)}(u) > \mathrm{F}_{i_0}^{\mathrm{last}(i_0)}(t_\emptyset)\}$.

En fait, on peut étudier moins de cas : il peut y avoir des éléments égaux dans $\mathcal{T}_{i_0}$ ce qui donnerait des contraintes redondantes pour u. Et, de manière plus significative, il n'est pas nécessaire d'étudier toutes les positions possibles de u entre deux éléments consécutifs de $\mathcal{T}_{i_0}$. Il suffit de positionner u entre deux éléments consécutifs (pour l'ordre usuel) de $Start_0 = \{t_{\mathrm{first}(i_0)\pi} \mid \mathrm{first}(i_0)\pi \in \Pi_{i_0}\}$, l'ensemble de tous les débuts de périodes chargées du serveur $\mathrm{first}(i_0)$. Cette idée générale est illustrée dans le cas particulier du théorème 9.4, mais nous gardons l'ensemble des cas mentionnés plus tôt pour faciliter la construction des trajectoires critiques.

Exemple : diamant

Pour la contrainte $\mathcal{T}^1$, on peut par exemple avoir $t_{124} \leq u \leq t_{24}$, ou $t_{24} \leq u \leq t_{134}$...

Pire charge pour le serveur j_0

Il faut maximiser la fonction objectif

$$\sum_{i \ni j_0} \mathrm{F}_i^{(\mathrm{prec}_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} \mathrm{F}_i^{(j_0)}(t_\emptyset)$$

On n'a pas à introduire de nouveaux cas ou de nouvelles contraintes linéaires.

9.2.3. Des trajectoires aux solutions LP

Soit λ une instance LP (un problème d'optimisation linéaire), potentiellement avec des inégalités strictes et dont la valeur optimale est opt_λ. Nous appelons *solution* de λ une affectation des variables qui satisfait les contraintes linéaires, et *solution optimale* une solution pour laquelle opt_λ est atteint (il peut ne pas y exister de solution optimale si $\lambda = +\infty$ ou lorsqu'il y a des inégalités strictes dans les contraintes).

Lemme 9.2. *Soit $\mathcal{N}$ un réseau acyclique et i_0 le flux observé (resp. j_0 le serveur observé). Soit Λ l'ensemble des instances LP correspondant à $\mathcal{N}$ construites à la section 9.2.1. Pour toute trajectoire $\{\mathrm{F}_i^{(j)}\} \in \mathrm{Traj}(\mathcal{N})$ pour laquelle un bit du flux i_0 subit un délai de bout en bout d (resp. la charge du serveur j_0 prend la valeur b) : il existe une instance LP $\lambda \in \Lambda$ qui admet une solution telle que $t_\emptyset - u = d$ (resp. $\sum_{i \ni j_0} .\mathrm{F}_i^{(prec_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} \mathrm{F}_i^{(j_0)}(t_\emptyset) = b$).*

Preuve. La trajectoire $\{\mathrm{F}_i^{(j)}\} \in \mathrm{Traj}(\mathcal{N})$ contient toutes les informations nécessaires. On fixe la valeur de la variable $t_\emptyset$ comme étant l'instant auquel le bit qui subit le délai d quitte le réseau (resp. l'instant auquel la charge est b). Dans le cas du délai, il existe un instant u tel que $u \leq t_\emptyset$, $\mathrm{F}_{i_0}^{(0)}(u) > \mathrm{F}_{i_0}^{(\mathrm{last}(i))}(t_\emptyset)$ (c'est la définition du délai), et $d = t_\emptyset - u$. On peut donc calculer la valeur de u, et on sait de plus que la variable u satisfait les contraintes ci-dessus.

En utilisant l'interprétation des variables t_π (comme débuts de périodes chargées), présentées à la section 9.2.1, on peut trouver de proche en proche leurs valeurs respectives dans la trajectoire (il faut calculer, pour chaque serveur j, les grandeurs $\sum_{i \ni j} \mathrm{F}_i^{(\mathrm{prec}_i(j))}$ et $\sum_{i \ni j} \mathrm{F}_i^{(j)}$ pour détecter les débuts de périodes chargées). Les valeurs $\{t_\pi,\ \pi \in \Pi\}$ sont totalement ordonnées (ce sont des valeurs) et satisfont les prédicats (P1) et (P2) (par construction). Il existe donc une famille de contraintes temporelles $(\mathcal{T}_1, \ldots, \mathcal{T}_p) \in \mathrm{Tot}(\Pi_1) \times \cdots \times \mathrm{Tot}(\Pi_p)$ qui sont toutes satisfaites par l'ensemble des valeurs $\{t_\pi,\ \pi \in \Pi\}$, et qui n'utilisent que des contraintes de type $=$ ou $<$ (on les obtient d'après les valeurs). Par conséquent, ces contraintes temporelles sont mutuellement compatibles, et pour tout i, j, t_π, on lit la valeur $\mathrm{F}_i^{(j)}(t_\pi)$ à partir de la trajectoire (on connaît les t_π). Le choix de $(\mathcal{T}_1, \ldots, \mathcal{T}_p)$ (et, dans le cas du délai, la position de u dans $\mathcal{T}_{i_0}$) définit l'instance LP $\lambda \in \Lambda$ du lemme.

La définition des valeurs t_π (et u), et le fait que les trajectoires vérifient (de fait) les contraintes de trajectoires, assurent que toutes les contraintes linéaires de λ sont satisfaites.

De plus, par définition de $t_\emptyset$ (et u), on a $t_\emptyset - u = d$ (resp. $\sum_{i \ni j_0} \mathrm{F}_i^{(\mathrm{prec}_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} \mathrm{F}_i^{(j_0)}(t_\emptyset) = b$). ■

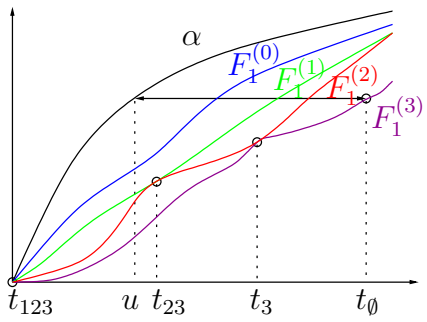


FIGURE 9.4 : Lecture des valeurs t_π à partir d'une trajectoire (un flux et trois serveurs). On lit $t_\emptyset$, puis t_3 sur $\mathrm{F}_1^{(3)}$, puis t_{23} sur $\mathrm{F}_1^{(2)}$...

9.2.4. Des solutions LP aux trajectoires

Lemme 9.3. *Soit $\mathcal{N}$ un réseau acyclique et i_0 le flux observé (resp. j_0 le serveur observé). Soit Λ l'ensemble des instances LP — correspondant à $\mathcal{N}$ — construites à la section 9.2.1. Alors, pour toute instance $\lambda \in \Lambda$ admettant au moins une solution, il existe une trajectoire $\{\mathrm{F}_i^{(j)}\} \in \mathrm{Traj}(\mathcal{N})$ telle que le pire délai de bout en bout d pour le flux i_0 vérifie $d = t_\emptyset - u$ (resp. la pire charge b pour le serveur j_0 vérifie $b = \sum_{i \ni j_0} \mathrm{F}_i^{(prec_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} \mathrm{F}_i^{(j_0)}(t_\emptyset)$).*

Preuve. Avant de décrire la construction de la trajectoire $\{\mathrm{F}_i^{(j)}\}$, faisons quelques remarques sur les instances LP et leurs solutions. Nous rappelons que chaque flux i possède une courbe d'arrivée α_i affine par morceaux et concave.

Existence de solutions

Pour la charge pire-cas, toutes nos instances LP admettent au moins une solution : il suffit de prendre des valeurs t_π vérifiant les contraintes temporelles et de fixer toutes les valeurs $\mathrm{F}_i^{(j)}(t_\pi)$ à zéro. Pour le pire délai de bout en bout, il existe des solutions si et seulement si α_{i_0} n'est pas la fonction nulle : dans ce cas, comme elle est concave, on a $\alpha_{i_0}(u) > 0$ pour tout $u > 0$. Ainsi, on choisira des valeurs t_π et $u > 0$ vérifiant les contraintes temporelles et on fixera toutes les valeurs $\mathrm{F}_i^{(j)}(t_\pi)$ à zéro sauf $\mathrm{F}_{i_0}^{(\mathrm{first}(i_0))}(t_\pi) = \alpha(t_\pi)$, ce qui assure $\mathrm{F}_{i_0}^{(0)}(u) > \mathrm{F}_{i_0}^{\mathrm{last}(i_0)}(t_\emptyset)$.

Le problème des inégalités strictes

Notre instance λ peut comprendre des inégalités strictes. Cependant les solveurs LP n'utilisent que des inégalités larges et donc l'instance qui sera effectivement résolue est l'instance $\hat{\lambda}$ où tous les signes $<$ sont remplacés par $\leq$. Comme l'ensemble des solutions

de $\hat{\lambda}$ est la clôture de l'ensemble convexe non vide des solutions de λ, on a $opt_{\hat{\lambda}} = opt_{\lambda}$ et pour toute solution de $\hat{\lambda}$, il existe une solution de λ arbitrairement proche. Par conséquent, si le solveur LP ne trouve pas de solution réalisant opt_{λ} et vérifiant λ, on travaillera avec une suite de solutions de λ dont les fonctions objectifs tendent vers $opt_{\lambda} = opt_{\hat{\lambda}}$.

Problème d'affectation pour les variables $F_i^{(j)}(t_\pi)$

Comme notre instance LP λ peut utiliser des inégalités larges, le cas suivant peut se produire dans une solution de λ : pour $i \in \mathbb{F}$, $j \in i$, $\pi, \pi' \in \Pi_i^{(j)}$, on peut avoir $t_\pi = t_{\pi'}$ mais $F_i^{(j)}(t_\pi) < F_i^{(j)}(t_{\pi'})$. Or, cette dernière forme de contrainte (mais avec inégalité large) n'est rajoutée dans λ que lorsque $\mathcal{T}(\pi, \pi') \in \{\leq, <\}$ donc on avait la contrainte $\pi \leq \pi'$ (puisque $\pi = \pi'$), et l'inégalité $F_i^{(j)}(t_\pi) < F_i^{(j)}(t_{\pi'})$ est en fait large (cf. contraintes de croissance des fonctions). Nous pouvons donc transformer notre solution en remplaçant $F_i^{(j)}(t_\pi)$ par la valeur $F_i^{(j)}(t_{\pi'})$; cette transformation ne viole aucune contrainte, et la nouvelle solution reste dans λ. En répétant cette opération si nécessaire, on aura finalement $F_i^{(j)}(t_\pi) = \max_{\{\pi', t_{\pi'} = t_\pi\}} F_i^{(j)}(t_{\pi'})$ qui est bien défini pour chaque t_π.

Lemme de translation, et ses applications

Étant donné une solution de λ et une constante arbitraire $c_i \in \mathbb{R}_+$, remplaçons les valeurs $F_i^{(j)}(t_\pi)$ par les valeurs $F_i^{(j)}(t_\pi) - c_i$ (on suppose ici que c_i est suffisamment petit pour que toutes ces valeurs restent positives). Alors, la nouvelle affectation des variables est toujours une solution de λ (l'ordre des $F_i^{(j)}(t_\pi)$ reste le même).

Étant donné une solution de λ, nous utilisons ce lemme de translation sur chaque flux i. Chaque ensemble de valeurs $\{t_\pi,\ \pi \in \Pi_i\}$ admet un minimum $t_{\pi min(i)}$; et d'après les contraintes LP, la valeur $F_i^{(0)}(t_{\pi min(i)})$ est inférieure à $F_i^{(j)}(t_\pi)$ pour tout $j \in i$. Fixons $c_i = F_i^{(0)}(t_{\pi min(i)})$ et soustrayons c_i de toutes les valeurs $F_i^{(j)}(t_\pi)$.

Une fois cette translation effectuée sur tous les flux, nous avons toujours une solution de λ. En effet toutes les contraintes, en particulier celles concernant plusieurs flux, font intervenir des grandeurs du type $F_i^{(\dots)}(\dots) - F_i^{(\dots)}(\dots)$ (dans les exemples précédents, on avait par exemple : $(F_1^{(1)}(t_{24}) + F_2^{(1)}(t_{24})) - (F_1^{(1)}(t_{124}) + F_2^{(1)}(t_{124})) \geq \beta_1(t_{24} - t_{124})$) et donc les constantes c_i se compensent flux par flux.

Après translation, pour tout $i \in \mathbb{F}$, nous avons $F_i^{(0)}(t_{\pi min(i)}) = 0$. Nous pouvons alors rajouter le point $(0, 0)$ parmi les points $(t_\pi, F_i^{(0)}(t_\pi))$, $\pi \in \Pi_i$, et continuer à vérifier les contraintes de courbe d'arrivée, car

$$F_i^{(0)}(t_\pi) - 0 = F_i^{(0)}(t_\pi) - F_i^{(0)}(t_{\pi min(i)}) \leq \alpha(t_\pi - t_{\pi min(i)}) \leq \alpha(t_\pi) - 0)$$

Lemme d'interpolation linéaire

Soit α (resp. β) une fonction concave (resp. convexe) de $\mathcal{F}_{\nearrow}$. Soit $x_1 < x_2 < \cdots < x_k$ et $y_1 \leq y_2 \leq \cdots \leq y_k$ deux ensembles de valeurs dans $\mathbb{R}_+$, tel que $\forall 1 \leq \ell \leq \ell' \leq k$, $y_{\ell'} - y_\ell \leq \alpha(x_{\ell'} - x_\ell)$ (resp. $y_{\ell'} - y_\ell \geq \beta(x_{\ell'} - x_\ell)$). Alors la fonction F de $[x_1, x_k]$ dans $\mathbb{R}_+$ obtenue par interpolation linéaire des points $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$, est croissante et vérifie pour tout $x_1 \leq s \leq t \leq x_k$, $\mathrm{F}(t) - \mathrm{F}(s) \leq \alpha(t-s)$ (resp. $\mathrm{F}(t) - \mathrm{F}(s) \geq \beta(t-s)$). La preuve découle directement de l'utilisation de la concavité (resp. de la convexité).

Reconstruction des fonctions cumulées pour chaque flux

Étant donné une solution de λ, appliquons tout d'abord les transformations décrites ci-dessus. Puis considérons un flux i, et définissons toutes les fonctions $\mathrm{F}_i^{(j)}$, $j \in i$ ou $j = 0$, comme nulles de 0 à $t_{\pi min(i)}$, la plus petite valeur de $\{t_\pi,\ \pi \in \Pi_i\}$. Décrivons maintenant chaque $\mathrm{F}_i^{(j)}(t)$, $j \in i$ ou $j = 0$, pour $t \geq t_{\pi min(i)}$.

On définit la fonction $\mathrm{F}_i^{(0)}$ comme l'interpolation linéaire des points $\{(t_\pi, \mathrm{F}_i^{(0)}(t_\pi)) \mid \pi \in \Pi_i\}$, puis qui reste constante à partir de son point maximum.

Construisons alors les fonctions $\mathrm{F}_i^{(j)}$ par induction en « avançant » sur le chemin μ_i. Supposons que la fonction $\mathrm{F}_i^{\mathrm{prec}_i(j)}$ a été construite. Pour construire $\mathrm{F}_i^{(j)}$, nous devons définir des intervalles où les données du flux i sont dans la file d'attente du serveur j. Considérons les valeurs $\{t_{j\pi} \mid j\pi \in \Pi_i\}$ qui correspondent aux débuts de telles périodes chargées. Certaines de ces valeurs pourraient être égales, mais nous considérons seulement les valeurs distinctes, notées $t_1 < \cdots < t_m$. Par convention, nous notons $t_{m+1} = +\infty$. Et à chaque t_v, nous associons $t_v^+ = \max\{t_\pi \mid t_v \leq t_\pi < t_{v+1},\ \pi \in \Pi_i^{(j)}\}$ et les points $\mathrm{P}_v = \{(t_\pi, \mathrm{F}_i^{(j)}(t_\pi)) \mid t_v \leq t_\pi < t_{v+1},\ \pi \in \Pi_i^{(j)}\}$. Alors $\mathrm{F}_i^{(j)}$ est défini comme l'interpolation linéaire des points P_v sur chaque intervalle $[t_v, t_v^+]$ et est égal à $\mathrm{F}_i^{(\mathrm{prec}_i(j))}$ sinon.

Soit $j_{\max}$ le dernier serveur à partir duquel il y a un chemin dans Π. Alors, pour tous les j suivants sur μ_i, ,nous posons $\mathrm{F}_i^{(j)} = \mathrm{F}_i^{(j_{\max})}$.

D'après les lemmes précédents, les $\mathrm{F}_i^{(j)}$ ainsi construits appartiennent à $\mathcal{F}_{\nearrow}$ et satisfont les contraintes de trajectoire.

L'obtention de ces trajectoires par interpolation des points-solutions de λ nous assure que le flux i_0 subit un pire délai de bout en bout égal à $t_\emptyset - u$ (resp. la pire charge pour le serveur j_0 est égal à $\sum_{i \ni j_0} \mathrm{F}_i^{(\mathrm{prec}_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} \mathrm{F}_i^{(j_0)}(t_\emptyset)$). Cela vient du fait que $\mathrm{F}_{i_0}^{(\mathrm{last}(i_0))}(t) = \mathrm{F}_{i_0}^{(0)}(t)$ pour $t \geq t_\emptyset$.

■

Les lemmes 9.2 et 9.3, ainsi que le fait que $opt_{\widehat{\lambda}} = opt_\lambda$ prouvent le théorème 9.1, mais sans étudier la question des délais ou charges infinies. Le solveur LP donnera un résultat $+\infty$ si et seulement s'il s'agit effectivement du pire cas. Même s'il serait plus satisfaisant de caractériser de tels scénarios, il est plus simple ici d'utiliser le résultat classique suivant : dans un réseau acyclique avec une politique de service FIFO par flux,

pour tout flux i (resp. server j), le pire délai de bout en bout (resp. la pire charge locale) est borné si et seulement si chaque serveur sur le chemin menant à last(i) (resp. menant au serveur j) a un facteur d'utilisation (débit asymptotique du service ÷ somme des débits asymptotiques des arrivées) inférieur à 1, comme montré dans [Le Boudec 2001].

9.3. Scénario en tandem : algorithme polynomial

Nous étudions ici un cas particulier parmi les réseaux acycliques : les réseaux en tandem, c'est-à-dire les réseaux $\mathcal{N}$ dont le graphe orienté induit est un chemin orienté sans raccourcis. Ce qui implique que chaque flux traverse une suite de serveurs consécutifs du chemin. De tels scénarios ont déjà été étudiés dans [Schmitt 2006a, Schmitt 2008b, Lenzini 2008].

Pour cette classe de réseaux, le calcul du pire cas se réduit à résoudre une seule instance LP avec un nombre polynomial de variables et de contraintes, donc de complexité polynomiale. De plus, nous montrons pour chaque flux comment reconstruire une courbe de service minimal de bout en bout, qui, dans un certain sens, est optimale.

9.3.1. Algorithme pour le scénario en tandem

Théorème 9.4. *Soit $\mathcal{N}$ un réseau en tandem de n serveurs et p flux. Alors, étant donné un flux i (resp. un serveur j), il existe une instance LP avec $\mathcal{O}(pn)$ variables et $\mathcal{O}(pn^2)$ contraintes, dont l'optimum est le pire délai bout en bout pour le flux i (resp. la pire charge au serveur j).*

Preuve. Sans perte de généralité, nous supposerons que $i = 1$ et last(i) $= n$ (resp. $j = n$) car ce qui se passe après le dernier serveur observé n'impacte pas la dynamique de la première partie du réseau (les serveurs qui suivent le dernier serveur observé n'apparaissent pas dans le problème d'optimisation linéaire).

Une application directe du théorème 9.1 aux réseaux en tandem génère un unique ordre sur les $n+1$ variables t_{π_j}, pour les chemins $\pi_{j-1} = j \cdots n$ et $\pi_n = \emptyset$. Cet ordre est : $t_{\pi_0} \le t_{\pi_1} \le \cdots \le t_{\pi_n}$. Et donc, le calcul de la pire charge au serveur n se réduit à une seule instance LP, tandis que le calcul du pire délai de bout en bout introduit de plus la variable u et potentiellement n instances LP selon la position de u dans l'ordre des t_{π_j}.

Posons $f_1 = \text{first}(1) - 1$ et $e_1 = \text{last}(1) = n$. Nous montrons qu'il est inutile de considérer plusieurs instances LP pour calculer le pire délai : il n'y a pas besoin d'étudier toutes les positions de u dans l'ordre $t_{\pi_{f_1}} \le \cdots \le t_{\pi_{e_1}}$. Il suffit en effet d'une seule instance LP et de rajouter les contraintes où u doit apparaître : $t_{\pi_{f_1}} \le u \le t_{\pi_{e_1}}$, $\mathrm{F}_1^{(0)}(u) > \mathrm{F}_1^{(n)}(t_{\pi_{e_1}})$ et $\mathrm{F}_1^{(0)}(u) - \mathrm{F}_1^{(0)}(t_{\pi_{f_1}}) \le \alpha_1(u - t_{\pi_{f_1}})$. La fonction objectif et les autres contraintes restent inchangées. Remarquons de plus que maximiser la fonction objectif nous donnera l'égalité : $\mathrm{F}_1^{(0)}(u) - \mathrm{F}_1^{(0)}(t_{\pi_{f_1}}) = \alpha_1(u - t_{\pi_{f_1}})$.

Nous raisonnons par l'absurde : considérons une trajectoire pire-cas pour un réseau en tamdem. Cette trajectoire est obtenue comme solution de nos n instances LP, avec la méthode du cas général. Si cette trajectoire pire-cas n'était pas obtenue par la méthode alternative (une seule instance LP), on aurait $F_1^{(0)}(u) - F_1^{(0)}(t_{\pi_{f_1}}) < \alpha_1(u - t_{\pi_{f_1}})$, et on pourrait donc remplacer $F^{(0)}_{1|[t_{\pi_{f_1}},u]}$ par $\alpha_{1|[t_{\pi_{f_1}},u]}$. Ce faisant, la trajectoire reste valide pour le système : les contraintes de flux, de croissance des fonctions, d'arrivée et de service strict sont encore satisfaites. En ce qui concerne les débuts des périodes chargées, les données qui sont arrivées en plus dans le serveur sont transmises au début de la période chargée du prochain serveur. Ceci assure que la fonction cumulée en entrée dans ce prochain serveur n'est pas inférieure à la fonction cumulée en entrée d'origine, et donc la période chargée ne se terminera pas avant la période chargée de la fonction cumulée d'origine. Comme $F_1^{(0)}(u) < \alpha_1(u - t_{\pi_{f_1}})$, et que la fonction cumulée $F_1^{(e_1)}$ est croissante, $t_{\pi_{e_1}}$ doit augmenter, et on obtient ainsi un délai plus grand que dans la trajectoire d'origine. ■

9.3.2. Des délais aux courbes de service de bout en bout

Soit $\mathcal{N}$ un réseau en tandem, et intéressons-nous au flux 1. Jusqu'ici, nous avons cherché un moyen de calculer le pire délai pour des contraintes fixées $(\alpha_i)_{i\in\mathbb{F}}$ et $(\beta_j)_{j\in\mathbb{S}}$. On peut vouloir mesurer le comportement global du réseau vis-à-vis du flux 1, en particulier on peut se demander si le réseau lui garantit une courbe de service minimal de bout en bout. Soit $\beta \in \mathcal{F}_\nearrow$, nous disons que β est une *courbe de service (simple) de bout en bout* ou *courbe de service résiduel* [Schmitt 2006a, Schmitt 2008b]) si $F_i^{out} \geq \beta * F_i^{in}$. On l'appelle courbe de service bout en bout *universelle* si β est indépendant de α_1 (c'est-à-dire que β reste une courbe de service bout en bout pour tout choix de α_1). Précalculer une telle courbe universelle peut être utile afin de pouvoir obtenir des bornes sur les délais bout en bout pour le flux 1 avec plusieurs courbes α_1 différentes (en utilisant la distance horizontale du théorème 2.2). Dans le cas des réseaux en tandem, nous montrons maintenant que l'on peut calculer une courbe de service bout en bout universelle, qui est optimale dans un certain sens.

Théorème 9.5. *Soit $\mathcal{N}$ un réseau en tandem avec n serveur et p flux. Alors on peut calculer une courbe de service de bout en bout universelle pour le flux 1, qui est le maximum de toutes les courbes de service de bout en bout universelle.*

Preuve. Nous allons montrer que cette courbe de service universelle peut être calculée en utilisant le problème dual d'une instance LP.

Pour calculer une courbe de service de bout en bout universelle pour le flux 1, l'idée est d'envoyer une rafale de taille σ (avec $\alpha_1(t) = \sigma$) et de calculer le délai pire-cas pour ce flux grâce à notre instance LP. Soit $d(\sigma)$ ce délai maximum : nous allons calculer pour chaque σ la valeur correspondante de la fonction $d : \sigma \mapsto d(\sigma)$, qui est croissante. Nous

montrerons d'abord que sa pseudo-inverse $\beta : t \mapsto \inf\{\sigma \geq 0 \mid d(\sigma) \geq t\}$ est une courbe de service (universelle) pour le flux 1, puis nous verrons comment calculer d.

Nous raisonnerons par récurrence sur le nombre de serveurs et nous nous intéresserons au flux 1, en supposant qu'il traverse tous les serveurs. Cette hypothèse permet de simplifier la présentation, et on pourra généraliser en prenant $\mathbf{H}(\text{first}(1))$ comme initialisation et en remplaçant le serveur 1 par le serveur first(1)).

Hypothèse de récurrence : $\mathbf{H}(n)$

Soit $\{\mathrm{F}_i^{(j)} \in \mathcal{F}_\nearrow \mid i \in \mathbb{F},\ j \in \mathbb{S},\ j \in i\}$ une trajectoire pour le système avec n serveur en tandem. Pour chaque $t \in \mathbb{R}_+$, il existe $t_0 \in \mathbb{R}_+$ et une trajectoire $\{\tilde{\mathrm{F}}_i^{(j)} \in \mathcal{F}_\nearrow \mid i \in \mathbb{F},\ j \in \mathbb{S},\ j \in i\}$ pour le système qui vérifient :

1. $\tilde{\mathrm{F}}_1^{(0)}(s) = \mathrm{F}_1^{(0)}(t)$ si $s \geq t_0$ et $\tilde{\mathrm{F}}_1^{(0)}(s) = \mathrm{F}_1^{(0)}(s)$ sinon ;
2. $\forall i, \forall j \in i$, $\tilde{\mathrm{F}}_i^{(j)}(s) = \mathrm{F}_i^{(0)}(s)$ si $s \leq t_0$;
3. Soit $t_n = start_n(t)$, $\tilde{\mathrm{F}}_1^{(n)}(s) = \mathrm{F}_1^{(n)}(s)$ si $s \geq t_n$ et t_n est encore le début de la période chargée de t pour le serveur n dans la trajectoire $\{\tilde{\mathrm{F}}_i^{(j)}\}$.

Initialisation : $\mathbf{H}(1)$ est vraie

Soit $t \in \mathbb{R}_+$, et $t_1 = start_1(t)$. On définit $t_0 := t_1$ et la trajectoire $\tilde{\mathrm{F}}_i^{(j)}$ par $\forall i$,

- $\tilde{\mathrm{F}}_i^{(0)}(s) = \mathrm{F}_i^{(0)}(t)$ si $s \geq t_1$ et $\tilde{\mathrm{F}}_i^{(0)}(s) = \mathrm{F}_i^{(0)}(s)$ sinon ;
- $\tilde{\mathrm{F}}_i^{(1)}(s) = \mathrm{F}_i^{(0)}(s)$ si $s \leq t_1$ et $\tilde{\mathrm{F}}_i^{(1)}(s) = \mathrm{F}_i^{(1)}(s)$ sinon.

C'est une trajectoire pour le système : avant le temps t_0, le système se comporte comme un serveur infini et a donc le même comportement que le système d'origine. Comme le serveur est strict, le comportement pendant une période chargée dépend seulement de la trajectoire sur cette période. De plus, à l'instant t_0, une rafale arrive et une période chargée commence ; comme pendant cette période jusqu'au temps t, $\tilde{\mathrm{F}}_i^{(0)}(s) = \mathrm{F}_i^{(0)}(t) > \mathrm{F}_i^{(1)}(t) \geq \mathrm{F}_i^{(1)}(s) = \tilde{\mathrm{F}}_i^{(1)}(s)$, alors cette période chargée ne peut pas se terminer avant l'instant t_0, et comme le serveur est strict, nous avons une trajectoire pour le système.

Récurrence

Supposons que $\mathbf{H}(n-1)$ est vrai, et considérons un réseau en tandem à n serveurs et une trajectoire $\{\mathrm{F}_i^{(j)} \in \mathcal{F}_\nearrow \mid i \in \mathbb{F},\ j \in \{1,\dots,n-1\},\ j \in i\}$ de ce système. Soit $t \in \mathbb{R}_+$ et $t_n = start_n(t)$. Appliquons l'hypothèse $\mathbf{H}(n-1)$ aux $n-1$ premiers serveurs et à t_n. Soit $a = \mathrm{F}_1^{(n)}(t) - \mathrm{F}_1^{(n)}(t_n)$. Nous modifions les trajectoires $\tilde{\mathrm{F}}_1^{(j)}$ jusqu'au temps t_n, de la manière suivante : si $\tilde{\mathrm{F}}_1^{(j)}(s) \geq \mathrm{F}_1^{(n)}(t_n)$, alors $\tilde{\mathrm{F}}_1^{(j)}(s) := \tilde{\mathrm{F}}_1^{(j)}(s) + a$, et on ne change rien sinon. En d'autres termes, nous ajoutons une rafale de taille a à l'instant t_0, et la servons en rafale quand les données arrivées à l'instant $(\tilde{\mathrm{F}}_1^{(0)})^{-1}(\mathrm{F}_1^{(n)}(t_0))$ sont servies, dans chaque serveur $\mathrm{S}_1,\dots,\mathrm{S}_{n-1}$. C'est encore une trajectoire pour le système.

Nous déduisons alors $\{\tilde{F}_i^{(n)}\}$ d'après la trajectoire $\{\tilde{F}_i^{(j)} \in \mathcal{F}_{\nearrow} \mid i \in \mathbb{F},\ j \in \{1,\dots,n-1\},\ j \in i\}$ pour les $n-1$ premiers serveurs, qui satisfont les trois conditions. La première condition est déjà satisfaite car elle concerne seulement $F_1^{(0)}$. La deuxième condition peut également être satisfaite : d'après l'hypothèse de récurrence et les contraintes de flux, nous avons $\tilde{F}_1^{(n)}(s) \leq \tilde{F}_1^{(n-1)}(s) = \tilde{F}_1^{(0)}(s)$. Nous pouvons donc définir $\tilde{F}_1^{(n)}(s) = \tilde{F}_1^{(0)}(s)$ si avant l'instant t_0 chaque serveur se comporte comme un serveur infini.

Considérons à présent le n-ième serveur, à partir de l'instant t_n. Par construction, nous avons $F_1^{(n-1)}(t_n) = F_1^{(n)}(t_n) + a$, et $t_n = start_n(t)$. La troisième condition peut alors être satisfaite en définissant $\tilde{F}_1^{(n)}(s) = F_1^{(n)}(s)$. Alors l'hypothèse $\mathbf{H(}n\mathbf{)}$ est vraie.

Retour à l'instance LP

Étudions la trajectoire $\{\tilde{F}_i^{(j)}\}$. Jusqu'à l'instant t_0, les serveurs se comportent comme des serveurs infinis. Alors, la fonction de départ après l'instant t_0 ne dépend pas de la trajectoire avant t_0. A l'instant t_0, on a une rafale de taille $b = F_1^{(n)}(t) - F_1^{(0)}(t_0)$. Le délai maximum pour les paquets entrant à l'instant t_0 peut donc être calculé par notre problème d'optimisation linéaire avec $\alpha_1 : s \mapsto b$. Donc, comme $\tilde{F}_1^{(n)}(t) = F_1^{(n)}(t)$ et $\tilde{F}_1^{(0)}(t_0) = F_1^{(0)}(t_0) + b$, nous avons $F_1^{(n)}(t) \geq F_1^{(0)}(t_0) + \beta(d^{-1}(b))$.

Notons que la courbe de service est optimal dans le sens où c'est le maximum de toutes les courbes de service de bout en bout universelles pour le flux 1. D'ailleurs, s'il existait une courbe de service bout en bout universelle β' et $\sigma \in \mathbb{R}_+$ tels que $\beta^{-1}(\sigma) > \beta'^{-1}(\sigma)$, cela invaliderait le fait que la borne sur le délai calculée par notre instance LP est tight quand la courbe d'arrivée pour le flux 1 est $\alpha_1 : t \mapsto \sigma$.

Revenons au cas où l'objectif est de maximiser le délai dans un système où la courbe d'arrivée pour le flux 1 est $\alpha_1 : t \mapsto \sigma$. Les seules contraintes dans lesquelles apparaît σ sont les contraintes d'arrivée du flux 1, c'est-à-dire $\forall f_1 \leq j' < j \leq e_1 = n$, $F_1^{(0)}(t_{\pi_j}) - F_1^{(0)}(t_{\pi_{j'}}) \leq \sigma$ et $F_1^{(0)}(u) - F_1^{(f_1)} \leq \sigma$. De plus, σ n'apparaît pas dans l'objectif. On peut donc exprimer notre problème d'optimisation linéaire avec des matrices par

$$\begin{gathered}\text{Maximiser } AX \\ \text{Sachant } BX \leq C(\sigma) \text{ et } X \geq 0,\end{gathered}$$

où seule C dépend de σ. D'après le théorème de dualité forte ([Vanderbei 2000]), le problème suivant a la même solution :

$$\begin{gathered}\text{Minimiser } C^t(\sigma)Y \\ \text{Sachant } B^tY \leq A^t \text{ et } Y \geq 0.\end{gathered}$$

Alors, seule la fonction objectif dépend de σ. Les contraintes $B^tY \leq A^t$ et $Y \geq 0$ définissent un polyèdre convexe. Pour toute fonction objectif, l'optimum est obtenu en un

sommet extrêmal du polyèdre. Ainsi, le délai peut être calculé en fonction de σ en calculant $\min_Y C^t(\sigma)Y$, où Y appartient à l'ensemble des sommets extrêmaux du polyèdre. Le nombre de ces sommets est fini mais peut être exponentiel en le nombre des contraintes. ■

Il faut faire néanmoins attention au fait que, même si cette courbe β est le maximum de toutes les courbes de service bout en bout universelles pour le flux 1, rien n'assure que pour toute courbe d'arrivée α_1 pour le flux 1, la distance horizontale entre α_1 et β sera le délai pire-cas exact (sauf pour $\alpha_1(t) = \sigma$ pour laquelle cela est garanti par définition de β). Cette distance est une borne supérieure mais n'est pas tight pour tout α_1.

Remarquons que l'objectif idéal consistant à trouver une courbe de service bout en bout universelle et optimale (tight pour tout α_1), est inatteignable pour certaines politiques de service. Par exemple, dans les réseaux FIFO, même pour un seul serveur avec un seul flux transverse, il y a une infinité de courbes de service simples non comparables, et leur maximum n'est pas une courbe de service [Le Boudec 2001, Lenzini 2008].

9.3.3. Travaux connexes

Commencée dans [Schmitt 2006a], l'étude des réseaux en tandem avec multiplexage aveugle a été grandement améliorée dans [Schmitt 2008b]. Dans cet article, les auteurs calculent des bornes de bout en bout tight pour certains réseaux en tandem — en détaillant tous les calculs pour un réseau à trois serveurs et trois flux —, et pour des réseaux *sink tree*. Dans ces cas particuliers, les auteurs arrivent à donner une formule close (avec disjonctions de cas). Une méthode est suggérée pour le cas général des réseaux en tandem dans le rapport technique correspondant [Schmitt 2007] mais tous les détails ne sont pas précisés. Nous étudions maintenant les similarités et les différences de nos deux approches.

Pour les scénarios traités dans [Schmitt 2008b], nous avons vérifié numériquement sur plusieurs exemples que notre algorithme utilisant des solutions LP donnait les mêmes résultats que leur formule (mais nous n'avons pas essayé de résoudre les instances LP à la main pour vérifier si nous pouvions obtenir les mêmes distinctions de cas que dans leur formule close).

La principale similitude réside dans le choix des variables et des ensembles de contraintes qui décrivent le système. Nous commençons en écrivant les contraintes du système sur les fonctions cumulées au début des périodes chargées, ce qui donne le même ensemble d'égalités et d'inégalités pour les réseaux en tandem, à un renommage près : par exemple, si $n = 2$, on aura $t_2, t_\emptyset$ d'un côté et t_1, t_0 de l'autre, et $\{F_2^{(1)}(t_2) - F_2^{(0)}(t_\emptyset) \leq \alpha_2(t_2 - t_\emptyset)\}$ d'un côté et $\{F_2^{(1)}(t_1) - F_2^{(0)} = \alpha_2(t_1 - t_0) - s_2^{(1)};\ s_2^{(1)} \geq 0\}$ de l'autre. Ces égalités et inégalités sont linéaires (pour le seau percé et les services latence-débit comme souligné dans [Schmitt 2008b] ; pour les courbes concaves/convexes affines par morceaux comme souligné dans les sections précédentes).

Cela étant, deux différences majeures distinguent nos approches. Premièrement, tandis

que nous essayons de résoudre directement cet ensemble de contraintes en utilisant l'optimisation linéaire (quitte à ajouter certaines contraintes et distinguer certains cas, à cause de l'ordre des événements), l'approche de [Schmitt 2007, Schmitt 2008b] pour des courbes d'arrivée/service de type seau percé/latence-débit consiste en :

- Effectuer des manipulations algébriques (en particulier en utilisant le lemme sur la convolution de fonctions latence-débit) pour obtenir, à partir de l'ensemble initial d'égalités et d'inégalités, une inégalité qui ressemble à une courbe de service simple de bout en bout. Par exemple pour le flux 1 dans le scénario à 3 serveurs et 3 flux de [Schmitt 2008b] : $F_1^{(2)}(t_3) \geq F_1^{(0)}(t_0) + \beta_{R,T}(t_3 - t_0)$ où le débit R est fixé, mais la latence T dépend de variables libres $s_2^{(1)}$ et $s_3^{(1)}$. Ces paramètres R et T ne dépendent pas de α_1 la courbe d'arrivée du flux 1.
- Cet ensemble de courbes de service simple de bout en bout universelles $\beta_{R,T}$ admet un maximum β (car R est fixé). Il est atteint pour un bon choix de valeurs $s_2^{(1)}$ et $s_3^{(1)}$ — qui optimisent T —, qui peut être calculé en résolvant une instance LP.
- Il est alors montré que pour toute courbe d'arrivée α_1, la distance horizontale $h(\alpha_1, \beta)$ entre α_1 et β, qui est une borne supérieure sur le pire délai de bout en bout, est en fait tight, c'est-à-dire qu'il existe une trajectoire qui réalise cette borne. Cette trajectoire critique est complètement décrite dans le cas des tandem *sink tree* et dans un cas de scénario à 3 serveurs et 3 flux.

La seconde différence concerne le traitement des courbes d'arrivée/service de la forme concave/convexe, en les décomposant en seau percé/latence-débit. Contrairement à l'algorithme polynomial du théorème 9.4, la méthode utilisée dans [Schmitt 2008b] est de décomposer les courbes comme un maximum de courbes de service latence-débit (resp. un minimum de courbes d'arrivée de type seau percé), puis de calculer le service résiduel pour chaque combinaison de courbes de service/courbe d'arrivée, et enfin de calculer le maximum de toutes ces courbes. Le fait que le maximum de ces courbes de service simple soit une courbe de service simple pour le système n'est pas garanti, et les bornes tight non plus. La garantie est sur le fait que la distance horizontale entre la courbe d'arrivée du flux observé et la courbe maximum est une borne supérieure sur les délais.

Comme souligné dans l'article, cette méthode a un coût ($\prod_{i=1}^{p} |\alpha_i| \prod_{j=1}^{n} |\beta_j|$ combinaisons à étudier). Une comparaison plus poussée sur un cas particulier se trouve dans [Bouillard 2009c].

9.4. Résultats

Nous comparons nos résultats avec d'autres méthodes existantes. Jusqu'alors, deux types de méthodes ont été utilisées (voir [Schmitt 2008b, Schmitt 2007] pour les détails) : *Total Flow Analysis* (TFA) et *Separated Flow Analysis* (SFA), voir section 2.3.

À notre connaissance, ce sont les deux seules méthodes existantes pour l'analyse systématique d'un réseau acyclique dans le cas général.

Toutes les expérimentations qui suivent ont été effectuées par Anne Bouillard à l'aide du programme « nc-tandem-tight », qu'elle a programmé en OCaml. Il génère, à partir d'une description du système, les contraintes LP décrites dans ce chapitre. Les problèmes d'optimisation linéaire utilisés pour les expérimentation peuvent être trouvés à l'adresse suivante : http://perso.bretagne.ens-cachan.fr/~bouillar/NCbounds/.

Voici les formules que nous utilisons pour TFA et SFA avec des courbes d'arrivée *leaky bucket* et des courbes de service strict latence-débit. Soit un serveur (cf. figure 9.5) offrant une courbe de service strict $\beta(t) = R(t-T)_+$, et traversé par deux flux 1 et 2 (on considérera, comme souvent, que le flux 1 est le flux observé, et le flux 2 représente l'agrégation des autres flux traversant le serveur).

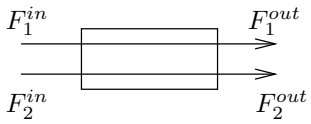


Figure 9.5 : Un serveur traversé par deux flux

Si F_1^{in} est borné supérieurement par $\alpha_1(t) = \sigma_1 + \rho_1 t$, et F_2^{in} est borné supérieurement par $\alpha_2(t) = \sigma_2 + \rho_2 t$, alors F_1 reçoit un service simple de courbe $\beta_1 = (\beta - \alpha_2)_+$.

Une borne supérieur pour le délai pour F_1 — qui sera utilisée en TFA — est

$$d_1 = T + \frac{\sigma_1 + \sigma_2 + \rho_2 T}{R - \rho_2}.$$

La courbe de service simple pour F_1 — qui sera utilisée en SFA — est

$$\beta_1(t) = (R - \rho_2)\left(t - T - \frac{\sigma_2 + \rho_2 T}{R - \rho_2}\right)_+.$$

Alors, la fonction cumulée F_1^{out} est bornée supérieurement par

$$\alpha_1'(t) = \sigma_1 + \rho_1\left(T + \frac{\sigma_2 + \rho_2 T}{R - \rho_2}\right) + \rho_1 t.$$

Étant en multiplexage aveugle, et par symétrie, ces résultats s'appliquent aussi au flux 2 en échangeant les indices 1 et 2 dans les formules.

Pour obtenir les bornes sur le réseau global, on peut utiliser ces formules sur l'ensemble des noeuds du réseau, triés par ordre topologique.

9.4.1. Scénario en tandem

Pour générer les fichiers du problème d'optimisation linéaire associés à un réseau en tandem, dans le but d'obtenir des bornes sur les délais et la charge pire cas, Anne Bouillard a écrit un programme qui peut être téléchargé (cf. page 142). Ce programme a été écrit en Ocaml [1], et génère un programme LP à partir d'un petit fichier décrivant le réseau en tandem. Ce programme peut être résolu à l'aide d'un solveur comme lp_solve [2] par exemple.

Nous comparons tout d'abord nos résultats pour le scénario en tandem, où le flux observé traverse tous les serveurs, et les flux transverses intersectent le flux observé sur deux serveurs consécutifs (sauf aux extrémités). Un exemple est présenté figure 9.6. Les serveurs ont les mêmes caractéristiques : ils ont une latence de 0.1s, et un débit (service) de 10Mbps. Les flux on une rafale maximale de 1Mb et un débit (arrivée) de 0.67Mbps.

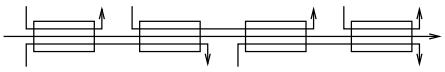


FIGURE 9.6 : Scénario en tandem avec 4 serveurs

La figure 9.7 montre les bornes sur le délai obtenues avec chacune des trois méthodes (TFA, SFA et la méthode LP tight). Sans surprise, les trois méthodes donnent les mêmes résultats quand il n'y a qu'un seul serveur. Pour un réseau à 20 serveurs, la méthode LP réduit la borne SFA par un facteur $8/5 = 1,6$, pour un taux d'utilisation des serveurs de 20%.

La figure 9.8 montre les variations entre les méthodes SFA et LP quand le taux d'utilisation des serveurs varie pour un nombre de serveurs fixé à 20 (seuls les débits d'arrivée varient). Quand le taux d'utilisation augmente, le gain de la méthode LP devient très important.

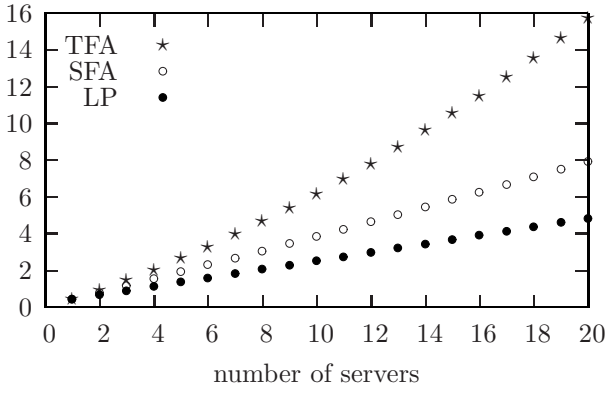


FIGURE 9.7 : Bornes supérieurs sur le délai pour le scénario de la figure 9.6

1. http ://caml.inria.fr/ocaml
2. http ://lpsolve.sourceforge.net/5.5/

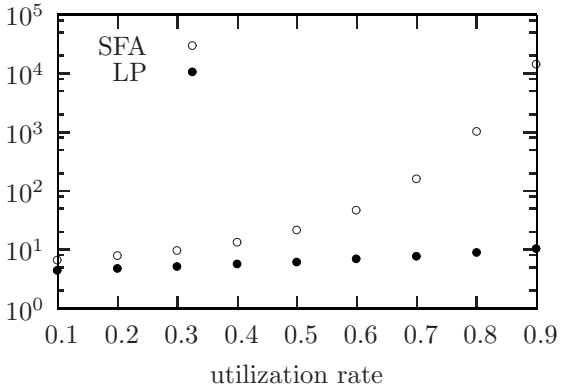


FIGURE 9.8 : Borne supérieure sur le délai pour le scénario de la figure 9.6 (avec 20 serveurs) quand le taux d'utilisation des serveurs augmente

9.4.2. Scénario sans dépendances cycliques

Nous montrons nos résultats sur un petit exemple, présenté sur la figure 9.9. Il y a quatre serveurs (avec les mêmes caractéristiques que dans l'exemple précédent), et quatre flux, ayant tous les mêmes caractéristiques, à savoir une rafale maximale de 1Mb, et nous faisons varier le débit d'arrivée de 0.5Mbps à 4.5Mbps — de telle sorte que le taux d'utilisation des serveurs passe de 10% à 90%.

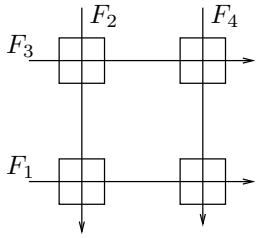


FIGURE 9.9 : Réseau en carré

Nous calculons les bornes sur le délai pour le flux F_1 avec quatre méthodes :

- la méthode (TFA) ;
- la méthode (SFA) ;
- la méthode (LP) en générant 11 programmes LP (un pour chaque ordre possible sur les variables temporelles, comme expliqué dans la section 9.2 ;
- la méthode (ULP), obtenue en résolvant un unique programme LP, où seules les contraintes communes aux 11 programmes de la méthode (LP) sont gardées, c'est-à-dire les contraintes dues au prédicat (P1) (p. 128).

La méthode ULP ne donne pas de bornes tight, mais ses résultats sont bien meilleurs que ceux obtenus avec TFA ou SFA. C'est donc un bon candidat pour être utilisé en pratique. Les résultats sont présentés figure 9.10.

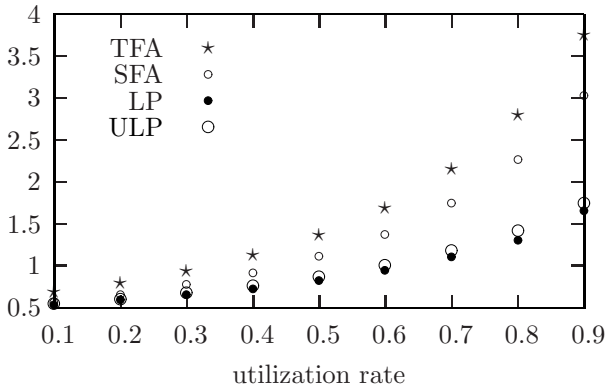


FIGURE 9.10 : Bornes supérieures sur les délais pour le flux 1 dans le réseau en carré de la figure 9.9.

Conclusion

Nous avons montré que l'on peut calculer des bornes exactes sur les pires charges locales et les pires délais de bout en bout, dans le cadre du Network Calculus, pour les réseaux sans dépendances cycliques, et en multiplexage aveugle [3].

Le nombre et la taille des problèmes d'optimisation linéaires qu'il faut résoudre peuvent être petits ou très grands selon le réseau considéré. Même si le problème est intrinsèquement difficile [Bouillard 2010], on piste à explorer est la diminution du nombre de problèmes d'optimisation linéaire à résoudre ainsi que leur taille — nos instances LP peuvent en effet comporter des inégalités redondantes (par exemple entre les contraintes de croissance des fonctions, et de service minimum). On peut aussi aussi fixer la valeur de $t_\emptyset$, étant donné que l'ensemble des trajectoires et donc l'ensemble des solutions LP est invariant par décalage temporel. Ce sont des petites améliorations, mais on peut espérer qu'il existe des réductions significatives.

On peut aussi raffiner le modèle, comme prendre en compte la taille des paquets, ou considérer des courbes de service maximum (resp. d'arrivée minimum), comme en RTC [Thiele 2000] — ce qui empêcherait les phénomènes de propagation (resp. d'épuisement) instantané des données.

La question du calcul exact des performances pire-cas pour des topologies générales en blind multiplexing, ou pour des réseaux sans dépendances cycliques avec d'autres politiques de service comme FIFO, restent un problème ouvert.

Pour des résultats plus récents sur cette méthode, appliquée dans le cas de politique de service à priorité fixe, voir [Junier 2010].

3. L'implémentation pour des réseaux sans dépendances cycliques quelconques est en développement.

Quatrième partie :

Remarques sur les notations et les preuves

CHAPITRE 10

OUTILS DE PREUVE EN NETWORK CALCULUS

Le théorème présenté dans ce chapitre est un classique en Network Calculus (cf. [Le Boudec 2001]), et il a été utilisé par de nombreux auteurs : [Bouillard 2007a, Bouillard 2008c], [Kim 2004], [Nemeth 2005], [Pandit 2006a, Pandit 2006b], [Schmitt 2006a, Schmitt 2006b]. De plus, il ne présente pas de difficulté particulière.

Nous souhaitons ici présenter trois preuves différentes de ce théorème. Notre but n'est évidemment pas de prouver ce résultat présent dans la littérature, mais d'explorer, à l'aide de cet exemple, les différents outils de preuve qui s'offrent à nous en NC. En effet, il est à notre sens intéressant de voir qu'en NC, certaines preuves sont purement mathématiques, d'autres utilisent l'interprétation des mouvements qui peuvent se produire dans le réseau, d'autres enfin utilisent des arguments purement géométriques — en utilisant la forme des courbes considérées.

Remarque 10.1 (Qu'est-ce qu'une preuve ? Équation et interprétation)**.** *Il est parfois tentant de construire une preuve à partir de l'« interprétation », c'est-à-dire de décrire ce qui se passe dans le système. Évidement, cela est souvent pratique, et parfois nécessaire, en particulier quand le formalisme associé n'existe pas. Mais les risques d'erreur et d'ambiguïté sont grands.*

Nous reprenons ici le théorème 8.4.

Théorème 10.1 ([Le Boudec 2001])**.** *Soit f et g deux fonctions convexes affines par morceaux définies sur $\mathbb{R}_+$ par un nombre fini de morceaux. Soit ρ_f (resp. ρ_g) la pente du dernier segment semi-infini de f (resp. g) s'il existe, ou $\rho_f = +\infty$ (resp. $\rho_g = +\infty$) si f (resp. g) est égal à $+\infty$ à partir d'un certain point. Alors la convolution $f * g$ consiste tout d'abord en la concaténation, par ordre croissant des pentes, et à partir de $f(0)+g(0)$, de tous les morceaux de pente $< \min(\rho_f, \rho_g)$*

de f et g ; puis se termine par la concaténation d'un segment semi-infini de pente $\min(\rho_f, \rho_g)$ *si cette valeur est finie.*

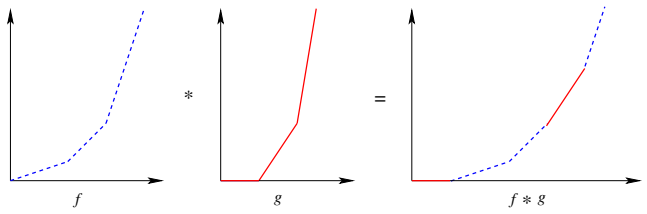


FIGURE 10.1 : Convolution de deux fonctions convexes affines par morceaux.

10.1. Première preuve : différenciation locale

Soit $f, g \in \mathcal{F}$ deux fonctions convexes. En tant que fonctions convexes, f et g sont continues et admettent une dérivée à gauche et à droite.

Pour toute fonction f admettant une dérivée à gauche (resp. droite), notons f'_ℓ (resp. f'_r) la dérivée à gauche (resp. à droite) de f. Si f est convexe, f'_ℓ et f'_r sont croissantes et $\forall u \in \mathbb{R}_+$, $f'_\ell(u) \leq f'_r(u)$. De plus elles sont respectivement continue à gauche et continue à droite.

Par convention, on définit $f'_\ell(0) = g'_\ell(0) = -\infty$.

Les deux lemmes suivants sont valables pour toute fonction convexe dans $\mathcal{F}$.

Lemme 10.2 (Folklore)**.** *Soit f une fonction convexe. Pour tout $u, v \in \mathbb{R}_+$, avec $u < v$, on a*

$$f'_r(u) \leq \frac{f(v) - f(u)}{v - u} \leq f'_\ell(v).$$

Lemme 10.3. *Soit $f, g \in \mathcal{F}$ deux fonctions convexes. Alors, pour tout $t \in \mathbb{R}_+$ et $u, v \in \mathbb{R}_+$ tel que $u + v = t$, les deux propriétés suivantes sont équivalentes :*

- $(f * g)(t) = f(u) + g(v)$.
- $g'_l(v) \leq f'_r(u)$ *et* $f'_l(u) \leq g'_r(v)$.

Pour tout $t \in \mathbb{R}_+$, il existe toujours de tels $u, v \in \mathbb{R}_+$.

Preuve.

Comme f et g sont continues et $f * g(t)$ est un infimum sur le compact $[0, t]$, on a l'existence de u et v.

Supposons que l'on peut trouver u, v tels que $u + v = t$, $g'_l(v) \leq f'_r(u)$ et $f'_l(u) \leq g'_r(v)$. Considérons u' et v' tels que $f * g(t) = f(u') + g(v')$ avec $u' + v' = t$ (alors $u - u' = v' - v$).

Sans perte de généralité, prenons $u' \leq u$. Alors,

$$\begin{aligned}
&f(u)+g(v)\\
&= f(u')+[f(u)-f(u')]+g(v')+[(g(v)-g(v')]\\
&= f(u')+g(v')+(u-u')\left[\tfrac{f(u)-f(u')}{u-u'}-\tfrac{g(v')-g(v)}{v'-v}\right]\\
&\leq f(u')+g(v')+(u-u')(f_l'(u)-g_r'(v))\\
&\leq f(u')+g(v').
\end{aligned}$$

En conséquence, $f(u)+g(v) = f*g(t)$.

Réciproquement, soit $f*g(t) = f(u)+g(v)$, $u+v=t$. Alors $f(u)+g(v) \leq \inf_{0<\varepsilon<u} f(u-\varepsilon)+g(v+\varepsilon)$. Ce qui implique $0 \leq \inf_{\varepsilon>0}[-f(u)+f(u-\varepsilon)+g(v+\varepsilon)-g(v)]$. Par convexité, $g(v+\varepsilon)-g(v) \leq \varepsilon g_r'(v+\varepsilon)$ et $f(u)-f(u-\varepsilon) \geq \varepsilon f_l'(u-\varepsilon)$. Ainsi $f_l'(u-\varepsilon) \leq g_r'(v+\varepsilon)$. La fonction f_l' (resp. g_r') est continue à gauche (resp. à droite). En faisant tendre ε vers 0, on obtient $f_l'(u) \leq g_r'(v)$. Symétriquement, on a $g_l'(v) \leq f_r'(u)$.

Montrons que de tels u et v existent. Soit $X_1 = \{u \in [0,t] \mid f_l'(u) \leq g_r'(t-u)\}$ et $X_2 = \{u \in [0,t] \mid g_l'(t-u) \leq f_r'(u)\}$.

Les ensembles X_i ne sont pas disjoints : f_l' est croissante et $g_r'(t-.)$ est décroissante. De plus, $X_1 \cup X_2 = [0,t]$: sinon, il existerait u tel que $f_l'(u) > g_r'(t-u) \geq g_l'(t-u) > f_r'(u)$, ce qui est impossible.

f_l' est continue à gauche, et $g_r'(t-.)$ aussi. De plus $0 \in X_1$. Donc X_1 est fermé. Enfin, comme X_1 et X_2 sont des ensembles fermés non disjoints de $[0,t]$, leur intersection est non vide, donc u et $v = t-u$ existent. ■

Nous appliquons maintenant ce théorème aux fonctions affines par morceaux. Soit $t \in \mathbb{R}_+$, et u, v tels que $f*g(t) = f(u)+g(v)$. Nous étudions la continuité, ainsi que la dérivée, de f et g en ces points.

- Si f' est définie en u et est continue sur $[u, u+\varepsilon]$, alors $f'(u) = f_l'(u+\varepsilon)$, $f_l(u+\varepsilon) \leq g_r'(v)$ et $g_l'(v) \leq f_r'(u) \leq f_r'(u+\varepsilon)$. Donc, $f*g(t+\varepsilon) = f(u+\varepsilon)+g(v)$ et $f*g$ est affine, de pente $f'(u)$ sur $[t, t+\varepsilon]$. On peut choisir ε de manière à ce que f change de pente en $u+\varepsilon$.
- De même, par symétrie, pour g et v si g' est continue sur un intervalle $[v, v+\varepsilon]$.
- Si f' et g' ne sont pas définies en u et v, et si $f_r'(u) \leq g_r'(v)$ (les rôles de f et g sont symétriques), alors $f_l'(u+\varepsilon) = f_r'(u) \leq g_r'(v)$ et $g_l'(v) \leq f_r'(u) = f_r'(u+\varepsilon)$. Donc $f*g(t) = f(u+\varepsilon)+g(v)$ et la pente de $f*g$ après t est $f_r'(u)$. Ceci est vrai pour tout ε plus petit que la longueur (du projeté sur l'axe des abscisses) du segment de pente $f_r'(u)$ dans f. Si ce segment est de longueur infinie, ceci est vrai pour tout $\varepsilon \in \mathbb{R}_+$.

Ainsi, on peut construire $f*g$ en augmentant u et v (parfois alternativement). Le premier point montre que tous les morceaux de f et g apparaissent dans la construction. Le troisième point donne la règle de concaténation : le morceau de plus petite pente apparaît en premier. Mais aussi le fait que lorsqu'on arrive sur un segment de longueur infinie, on n'en change plus. ■

10.2. Deuxième preuve : algorithmique

Considérons le premier segment de f, de pente s_f, et le premier segment de g, de pente s_g, et supposons que $s_f \leq s_g$ et que la longueur du segment de f est $\ell_f \in]0, \infty]$. Alors, pour tout $t \in [0, \ell_f]$,

$$\begin{aligned} f * g(t) &= \inf_{u+v=t, u,v\geq 0} f(u) + g(v) \\ &= \inf_{u+v=t, u,v\geq 0} f(0) + s_f u + g(0) + v\frac{g(v)-g(0)}{v-0}. \end{aligned}$$

Comme g est convexe, $\frac{g(v)-g(0)}{v-0} \geq s_g \geq s_f$ donc $f * g(t) = g(0) + f(t)$.

Si $\ell_f = \infty$, alors $f * g$ est une fonction affine constitué des segments de f et g de pente plus petite.

Sinon, $f * g$ est constitué du segment de plus petite pente sur $[0, \ell_f]$.

Soit $t > \ell_f$ et supposons que $f * g(t) = f(u) + g(v)$, $u + v = t$ et $u < \ell_f$.

$$\begin{aligned} & f(\ell_f) + g(t - \ell_f) \\ &= [f(\ell_f) - f(u)] + f(u) + [g(t - \ell_f) - g(v)] + g(v) \\ &= f(u) + g(v) + s_f(\ell_f - u) - (\ell_f - u)\tfrac{g(t-\ell_f)-g(v)}{t-\ell_f-v}. \end{aligned}$$

Mais, $\frac{g(t-\ell_f)-g(v)}{t-\ell_f-v} \geq s_g \geq s_f$, donc $f(\ell_f) + g(t - \ell_f) \leq f(u) + g(v) = f * g(t)$.

D'où, $f * g(t) = f(\ell_f) + g(t - \ell_f)$ et on peut écrire

$$f * g(t) = f(\ell) + \tilde{f} * g(t - \ell_f),$$

avec $\tilde{f}(t - \ell_f) = f(t) - f(\ell_f)$. Autrement dit, $\tilde{f}$ est construit à partir de f en supprimant le premier segment de f (et $\tilde{f}(0) = 0$). Alors $\tilde{f}$ est aussi convexe, et on peut construire le reste $f * g$ par itération. Les segments de f et g sont donc concaténés par ordre croissant des pentes. Lorsqu'on arrive sur un segment de pente infinie, tous les segments de pente plus grande sont ignorés. ■

10.3. Troisième preuve : transformée de Legendre-Fenchel

Preuve. Sans perte de généralité, nous pouvons supposer que f et g sont croissantes et que $f(0) = g(0) = 0$. En effet, si f ou g a des morceaux de pente négative, soit R la pente la plus petite parmi toutes les pentes de f et g (donc $\mathrm{R} < 0$). Pour tout $t \in \mathbb{R}_+$, on a $(f * g)(t) - \mathrm{R}t = \inf_{u+v=t, u\geq 0, v\geq 0}((f(u) - \mathrm{R}u) + (g(v) - \mathrm{R}v))$. Les fonctions $u \mapsto f(u) - \mathrm{R}u$ et

$v \mapsto g(v) - Rv$ sont aussi convexes et affines par morceaux, mais sont croissantes. On peut alors travailler avec ces fonctions, puis obtenir $f * g$ en ajoutant $t \mapsto Rt$ (propositon 4.3).

De même, supposons que $f(0) \neq 0$ ou $g(0) \neq 0$. On travaillera alors avec $u \mapsto f(u) - f(0)$ et $v \mapsto g(v) - g(0)$ qui permet d'obtenir le cas général car $(f * g)(t) - f(0) - g(0) = \inf_{u+v=t, u\geq 0, v\geq 0}((f(u) - f(0)) + (g(v) - g(0)))$.

Soit $\mathcal{F}^+$ l'ensemble des fonctions croissantes h de $\mathbb{R}_+$ dans $\mathbb{R}_+ \cup \{+\infty\}$ telles que $h(0) = 0$. Nous utilisons la transformée de Legendre-Fenchel restreinte à $\mathcal{F}^+$. Elle associe à toute fonction $f \in \mathcal{F}^+$ la fonction $\widehat{f} \in \mathcal{F}^+$ définie par $\widehat{f}(\lambda) \stackrel{def}{=} \sup_{t\geq 0}(\lambda t - f(t))$ pour tout $\lambda \in \mathbb{R}_+$. Cet outil puissant d'analyse convexe [Rockfellar 1996, Touchette 2005, Chang 1999, Fidler 2006c, Lucet 2010] a des propriétés intéressantes :

- (LF1) Pour tout $f, g \in \mathcal{F}^+$, $\widehat{f * g} = \widehat{f} + \widehat{g}$.
- (LF2) Pour tout $f \in \mathcal{F}^+$, $\widehat{\widehat{f}} = f$ si et seulement si f est convexe.

Une conséquence de la seconde propriété est que $f \mapsto \widehat{f}$ est une involution (et donc une bijection) sur $\mathcal{C}^+$ l'ensemble des fonctions convexes de $\mathcal{F}^+$.

Le lemme suivant donne une formule pour la transformée de Legendre-Fenchel d'une fonction convexe affine par morceaux.

Lemme 10.4. *Soit $f \in \mathcal{F}^+$ une fonction convexe affine par morceaux avec ℓ morceaux sur son support. Pour chaque $1 \leq i \leq \ell$, notons r_i (resp. τ_i) la pente (resp. la distance horizontale) du i-ème morceau (en commençant à compter par la gauche, avec la convention $\tau_\ell = +\infty$ si le dernier morceau est semi-infini). Alors pour tout $\lambda \in \mathbb{R}_+$:*

$$\widehat{f}(\lambda) = \sum_{i=1}^{\ell} \tau_i (\lambda - r_i)_+$$

avec la convention $+\infty \times 0 = 0$.

La preuve de ce lemme est facile en utilisant la définition de la transformée de Legendre-Fenchel. Elle utilise le fait que la suite $(r_i)_{1\leq i\leq \ell}$ est croissante, afin de déterminer la valeur t pour laquelle le supremum est atteint.

D'après la propriété (LF1) et le lemme 10.4, pour tout $\lambda \in \mathbb{R}_+$,

$$\widehat{f * g}(\lambda) = \widehat{f}(\lambda) + \widehat{g}(\lambda)$$

$$= \sum_{(r,\tau)\in C_f} \tau(\lambda - r)_+ + \sum_{(r,\tau)\in C_g} \tau(\lambda - r)_+$$

où C_f (resp. C_g) contiennent les couples (pente, distance horizontale) de tous les morceaux de f (resp. g).

De plus, nous savons que $f * g$ est convexe. Ceci est vrai pour tout $f, g \in \mathcal{F}$ (même si elles ne sont pas affines par morceaux) car $f * g$ est l'infimum d'une fonction convexe sur un ensemble convexe [Rockfellar 1996].

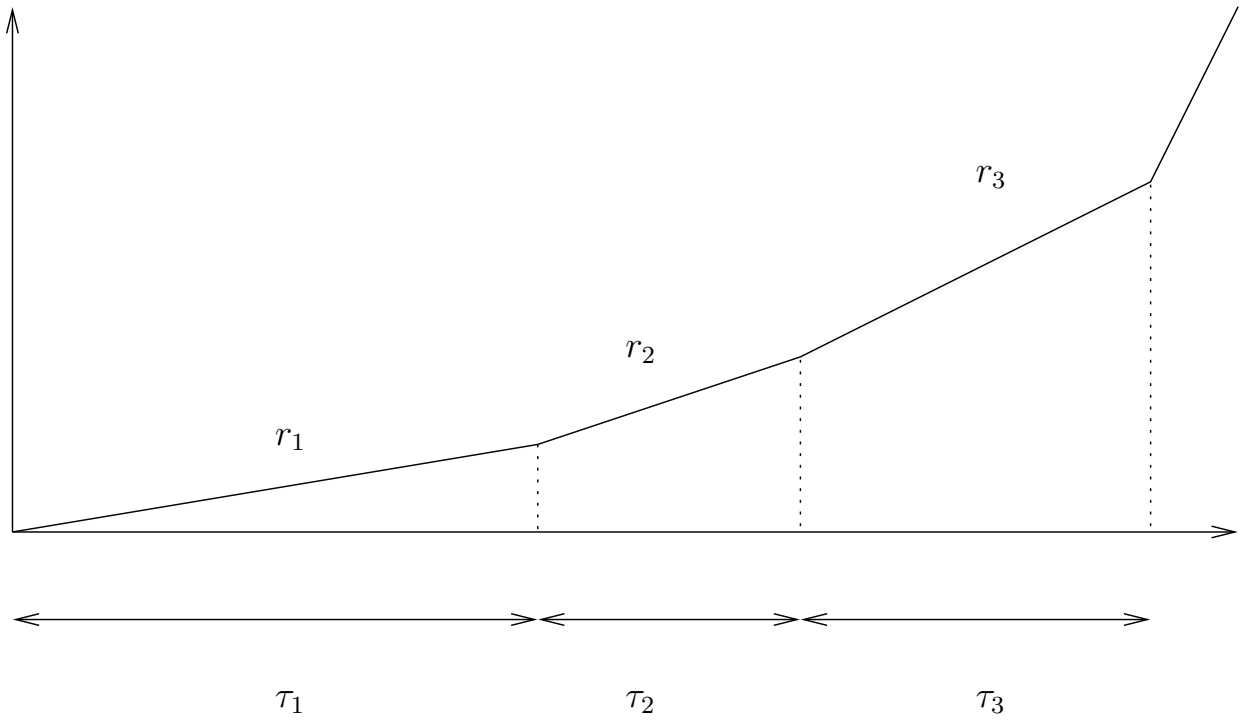


FIGURE 10.2 : Notations pour la transformée de Legendre-Fenchel

Nous introduisons à présent h la concaténation, à partir du point $(0,0)$, de tous les morceaux de f et g, triés par pente croissante (comme dans [Le Boudec 2001]). Si f ou g a un morceau semi-infini, c'est-à-dire $(\rho_f, +\infty) \in C_f$ ou $(\rho_g, +\infty) \in C_g$, alors nous ignorons les morceaux dont la pente est plus grande que ρ_f ou ρ_g et h se termine par un morceau semi-infini dont la pente est la plus petite des deux (notée ρ_h). Comme h est construit en triant les morceaux par pente croissante, h est convexe par construction, et le lemme 10.4 peut s'appliquer : pour tout $\lambda \in \mathbb{R}_+$, $\widehat{h}(\lambda) = \sum_{(r,\tau)\in C_h} \tau(\lambda - r)_+$ où C_h représente l'ensemble des couples (pente, distance horizontale) des morceaux de h .

Ainsi on a, pour tout $\lambda \in \mathbb{R}_+$, $\widehat{f * g}(\lambda) = \widehat{h}(\lambda)$. Si f et g n'ont pas de morceau semi-infini, $C_h = C_f \cup C_g$.

Sinon $C_h \subseteq C_f \cup C_g$. Mais certains termes, qui apparaissent *a priori* dans l'expression de $\widehat{f * g}(\lambda)$, ne jouent aucun rôle, et donc n'existent pas dans $\widehat{h}$: ils sont de la forme $\tau(\lambda - r)_+$ avec $r > \rho_h$, et apparaissent donc seulement pour $\lambda > \rho_h$, mais $\widehat{f * g}$ et $\widehat{h}$ contiennent le terme $+\infty \times (\lambda - \rho_h)_+$ qui vaut $+\infty$ pour ces valeurs de λ (ils sont "absorbés", dans la somme, par la valeur $+\infty$).

En conclusion, $f * g$ et h sont deux fonctions convexes de $\mathcal{F}^+$ et $\widehat{f * g} = \widehat{h}$. Comme la transformée de Legendre-Fenchel est injective sur $\mathcal{C}^+$, l'ensemble des fonctions convexes de $\mathcal{F}^+$, nous avons $f * g = h$.

■

Remarque 10.2. *Le théorème 8.4 est présenté dans [Le Boudec 2001, théorème 3.1.6 règle 9, pages 113-115].*

Deux fonctions y sont construites à partir de f et g : $h \stackrel{def}{=} f * g$ et h', la concaténation, à partir de $f(0) + g(0)$, de tous les morceaux de f et g triés par pente croissante.

En utilisant les épigraphes respectifs $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}$, $\mathcal{S}'$ de f, g, h, h' (et leur bordures $\partial\mathcal{S}_1$, $\partial\mathcal{S}_2$, $\partial\mathcal{S}$, $\partial\mathcal{S}'$), il est montré que $\partial\mathcal{S}' \subseteq \partial\mathcal{S}_1 + \partial\mathcal{S}_2$.

CHAPITRE 11

NOTATIONS

Remarque 11.1. *Ce chapitre traite des notations qui* pourraient *être utiles en Network Calculus. Les notations utilisées dans ce chapitre correspondent donc parfois à des notions qui ne sont pas abordées ailleurs dans ce mémoire ; nous utiliserons aussi des notations tirées de la littérature classique du NC. La plupart des notations présentées ici sont donc* spécifiques *à ce chapitre.*

Le calcul réseau manipule de nombreuses notions propres à la fois à son domaine de travail (flux, serveur, délai...), et à la modélisation (courbe d'arrivée, de service...). Il utilise le dioïde $(\min, +)$ comme support mathématique. Ce chapitre se propose de définir un ensemble de notations associées à ces notions. L'objectif des notations est double : non seulement de simplifier la manipulation (modélisation, preuves) et l'appropriation (diffusion, enseignement) du calcul réseau, mais aussi de redéfinir formellement certaines notions, et ce faisant, de les clarifier.

Introduction

Mais, alors que la recherche sur le calcul réseau s'active et que le besoin industriel est pressant, sa pénétration dans la communauté reste assez faible. Proposer des notations pour les principales notions aideraient à sa diffusion et sa manipulation.

Nous commençons par approfondir les enjeux liés aux notations, et présenter quelques notations mathématiques nécessaires pour cet article. La section 11.1 présente les notations que nous utilisons pour manipuler les principaux opérateurs du dioïde $(\min, +)$. C'est la section 11.2 qui présente le cœur de notre contribution, en présentant simultanément les principales notions de calcul réseau (courbe d'arrivée, de service) et les notations que nous proposons pour les désigner. La section 11.3 donne des précisions par rapport aux bornes calculées sur le modèle utilisé. Enfin, nous présentons notre conclusion.

Enjeux des notations

Définir des notations doit permettre d'aider à identifier et à manipuler les concepts. Cela se fait dans un contexte de notations déjà existantes en mathématiques, et il faut, autant que faire ce peut, rester cohérent avec l'existant.

Ainsi les notations de base utilisées en calcul réseau viennent de plusieurs domaines : principalement de la physique (la convolution) et des mathématiques (le dioïde min-plus). Les opérateurs sont souvent abordés de façons différentes dans ces différents domaines : parfois comme outils de calculs, parfois comme sujets d'étude à part entière. Il nous paraît nécessaire de clarifier les choses lorsque différentes notations sont utilisées pour le même objet. Il ne s'agit donc pas d'imposer nos propres notations, mais de faire la synthèse de celles existantes, et de comprendre quels sont les avantages et inconvénients de chacune, selon le contexte. Cela est particulièrement vrai pour tout ce qui concerne les opérateurs dans (min, plus) et la convolution.

D'autre part, plusieurs notions de calcul réseau (les courbes de service minimum ou strict par exemple) sont très proches les unes des autres, mais néanmoins distinctes ; et il n'existe pas, pour certaines d'entre elles, de notation qui permettrait de les distinguer facilement, ou de les manipuler formellement (sans pour autant nous dispenser d'une connaissance du sujet bien entendu). Nous proposons donc de nouvelles notations pour ces notions, tout on essayant de rester le plus proche possible de ce qui existe déjà dans la littérature du calcul réseau.

Enfin, il nous semble nécessaire de disposer de notations formelles pour exprimer la topologie d'un réseau de flux et de serveurs, mais aussi les politiques de service. Pour ce dernier point, il apparaît que les définitions seront forcément *ad hoc*, et qu'il est difficile de définir un cadre général pour toutes les politiques de service. Dans une partie sur les serveurs partagés nous abordons donc les problèmes de notations liées à la traversée d'un serveur par plusieurs flux. Nous traitons aussi le cas d'un flux traversant plusieurs serveurs.

Notations mathématiques spécifiques à ce chapitre

$\mathbb{R}_{\geq 0}$ désigne l'ensemble des réels positifs ou nuls. L'ensemble des intervalles de $\mathbb{R}_{\geq 0}$ est noté $\mathcal{I}_{\geq 0}$. La fonction de test sur un intervalle $A \subset \mathbb{R}$ est définie par $1_A(t) = 1$ si $t \in A$, 0 sinon. Par exemple, $1_{[0,2]}(t)$ vaut 1 si $t \leq 2$ et 0 sinon. La fonction $\delta_T(t)$ vaut 0 si $t \leq T$, $+\infty$ sinon. On note $[x]^+ = \max(x, 0)$.

Si $v \in V$ désigne un élément d'un ensemble, nous noterons $\mathbf{v}$ un vecteur de V^n, et v_i la ième composante du vecteur ($\mathbf{v} = (v_1, \ldots, v_n)$). Nous utiliserons la norme 1 d'un vecteur : $\|\mathbf{v}\| = \sum_{i=1}^{n} |v_i|$.

11.1. Dioïde min-plus

Un des intérêts du calcul réseau est que les notions qu'il manipule s'expriment très bien dans le dioïde dont la première loi est le minimum, et la seconde loi la somme.

Il nous faut donc des notations pour les opérateurs qui y sont utilisés. Nous notons $\wedge$ le minimum et $\vee$ le maximum, suivant les notations de [Le Boudec 2001] et, plus classiquement, + pour la somme.

Les flux de données sont représentés par leur fonction de cumul. Nous manipulons donc des fonctions croissantes ($\mathcal{G}$), nulles sur les négatifs ($\mathcal{F}$) et souvent nulles en 0 ($\mathcal{F}_0$).

$$\mathcal{G} \stackrel{\text{def}}{=} \{f : \mathbb{R} \to \mathbb{R} \cup \{+\infty\} \,|\, x < y \implies f(x) \leq f(y)\} \qquad \mathcal{F} \stackrel{\text{def}}{=} \{f \in \mathcal{G} \,|\, x < 0 \implies f(x) = 0\}$$

$$\mathcal{F}_0 \stackrel{\text{def}}{=} \{f \in \mathcal{F} \,|\, f(0) = 0\}$$

Trois opérateurs sur les fonctions sont utilisées, la convolution ($*$), la déconvolution ($\oslash$) et la clôture sous-additive ($\cdot^*$).

$$(f * g)(t) \stackrel{\text{def}}{=} \inf_{0 \leq s \leq t} f(t-s) + g(s)$$

$$(f \oslash g)(t) \stackrel{\text{def}}{=} \sup_{0 \leq s} f(t+s) - g(s)$$

$$f^* \stackrel{\text{def}}{=} \delta_0 \wedge f \wedge (f * f) \wedge (f * f * f) \wedge \cdots$$

Comparaisons avec d'autres auteurs

Nous avons repris à la fois des notations de [Le Boudec 2001] (pour la déconvolution $\oslash$) et de [Chang 2000] (pour la convolution $*$ et la clôture sous-additive $\cdot^*$). La notation de [Chang 2000] nous a semblé plus pertinente pour la convolution, car c'est la notation de la convolution dans l'algèbre usuelle. La notation $\otimes$ utilisée dans [Le Boudec 2001]

met plus l'accent sur le dioïde utilisé. De même pour la clôture sous-additive (notée $\bar{f}$ dans [Le Boudec 2001]), nous avons préféré mettre en évidence l'aspect auto-convolution. En ce qui concerne la déconvolution nous avons cette fois préféré la notation de [Le Boudec 2001] plutôt que celle de [Chang 2000] ($\div$), en la rapprochant plutôt de la théorie de la résiduation [Baccelli 1992] utilisée dans l'algèbre $(max, +)$ et qui peut être définie à droite ou à gauche (voir aussi [Le Corronc 2011]). Notre notation pourra facilement s'y adapter.

11.2. Calcul réseau

L'objectif du calcul réseau est de permettre de calculer des bornes supérieures garanties pour les délais subis par des flux dans des réseaux, ainsi que pour les quantités de mémoire utilisées par ces flux dans les éléments de réseau. Pour ce faire, il modélise un contrat de trafic par des « courbes d'arrivée », et modélise le serveur par une « courbe de service » [Cruz 1991a, Cruz 1991b, Le Boudec 2001, Chang 2000].

11.2.1. Courbes d'arrivée et de service

Courbes d'arrivée

Commençons par regarder comment noter qu'un flux $R \in \mathcal{F}_0$ admet une courbe d'arrivée $\alpha \in \mathcal{F}$. Soulignons pour commencer que le vocabulaire abonde pour désigner cette relation. On trouvera « R admet α comme courbe d'arrivée », « R est contrainte par α » ou « R est α-lisse [1] ». La définition formelle d'une courbe d'arrivée $\alpha \in \mathcal{F}$ pour un flux $R \in \mathcal{F}_0$ est :

$$\forall (t,s) \in \mathbb{R}^2_{\geq 0} : R(t+s) - R(t) \leq \alpha(s). \tag{11.1}$$

Intuitivement, le contrat de trafic stipule qu'il y aura au plus $\alpha(s)$ données produites sur tout intervalle de longueur s. On pourrait dire que $R \in \mathcal{F}_0$ a pour courbe d'arrivée α si R est plus petite qu'α quelle que soit l'origine du temps. On peut exprimer cela avec la définition équivalente en calcul réseau : $R \leq R * \alpha$. C'est en se basant sur cette notion de « plus petit » que nous proposons, comme [Schioler 2005], de noter $\prec$ la relation de courbe d'arrivée.

$$R \prec \alpha \overset{\text{def}}{\iff} R \leq R * \alpha \tag{11.2}$$

Cette notation permet de manipuler de façon assez naturelle des règles comme celles de l'équation (11.3), par analogie avec les réels ou les fonctions réelles ($r \leq a, r \leq a' \implies r \leq a \wedge a'$). Notons l'équation (11.4) qui permet de remplacer une courbe d'arrivée par sa clôture sous-additive, ce qui permet en pratique d'ajuster les courbes et d'avoir des

1. En anglais *has arrival curve*, *is contrained by* ou *is α-shaped*.

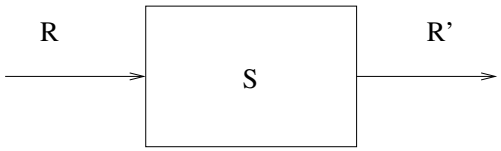


FIGURE 11.1 : Schéma de base du calcul réseau

résultats plus précis.

$$R \prec \alpha, \alpha \leq \alpha' \implies R \prec \alpha' \qquad R \prec \alpha, R \prec \alpha' \implies R \prec \alpha \wedge \alpha' \tag{11.3}$$

$$R \prec \alpha, R' \prec \alpha' \implies R + R' \prec \alpha + \alpha' \qquad R \prec \alpha \implies R \prec \alpha^* \tag{11.4}$$

Néanmoins, *il faut souligner que les analogies ne dispensent pas d'une connaissance du domaine, et que les notations peuvent induire en erreur*. En effet, dans les réels ou les fonctions réelles, on a $r \leq r', r' \leq a' \implies r \leq a'$, mais on n'a pas l'équivalent avec les courbes de service [2] : $R \leq R', R' \prec \alpha' \not\Rightarrow R \prec \alpha'$.

Courbes de service

C'est en cherchant à trouver des notations sur les courbes de service, et sur la relation entre flux d'entrée et flux de sortie que nous avons été amené à formaliser la notion même de serveur à laquelle nous cherchions une notation.

La définition de courbe de service est, par exemple dans [Le Boudec 2001], donnée ainsi : « Soit un serveur S, traversé par un flux de courbe d'entrée R et de sortie R'. Alors, S offre au flux un service β ssi $R' \geq R * \beta$. »

On dit que S possède une courbe de service β s'il offre un service β à tout flux. Cette notion de courbe de service pour tout flux apparaît sous le nom de *courbe de service universelle* dans [Chang 2000]. L'intuition est plus difficile à présenter que pour la notion de courbe d'arrivée. En laissant de côté des détails de continuité [3], l'équation $R' \geq R * \beta$ est équivalente à $\forall t, \exists s, R'(t) - R(t-s) \geq \beta(s)$. La nuance difficile à expliquer est dans le quantificateur existentiel de s, qui signifie qu'en regardant « assez longtemps » dans le passé, on a reçu au moins le service $\beta(s)$.

Pour donner une définition formelle à cette notion de courbe de service, nous avons été amené à repenser la notion de serveur. En effet, intuitivement, un serveur crée un flux en sortie à partir d'un flux en entrée. Un serveur serait donc une fonction de $\mathcal{F}_0$ vers $\mathcal{F}_0$. Mais ceci correspond à un fonctionnement *déterministe*. Et si, en pratique, on manipule plutôt des serveurs déterministes, lorsque l'on travaille sur les aspects théoriques du calcul réseau (justesse des bornes, pire ordonnancement), il faut considérer tous les serveurs

2. On pourra s'en convaincre avec le contre-exemple $R'(t) = t, R(t) = R'.1_{t \geq 1}, \alpha' = R$. On a bien $R \leq R'$, mais R' est très lisse (débit constant), alors que R possède une rafale, à la date $t = 1$.
3. Voir [Le Boudec 2001, prop. 1.3.1] pour les détails.

possibles, y compris les serveurs non déterministes. Un serveur associe donc à un flux un ensemble de flux de sortie. Nous notons $\mathcal{S}_1$ l'ensemble des serveurs à une entrée et une sortie [4]. Bien sûr, les bits sortant après être entrés, on a toujours $R' \leq R$. Nous notons $R \xrightarrow{S} R'$ pour signifier que R' est une sortie possible de R par S.

$$\mathcal{S}_1 \stackrel{\text{def}}{=} \{S : \mathcal{F}_0 \to \mathcal{P}(\mathcal{F}_0) \,|\, R' \in S(R) \implies R' \leq R\} \tag{11.5}$$

$$R \xrightarrow{S} R' \stackrel{\text{def}}{=} R' \in S(R) \tag{11.6}$$

Quant à la relation de « courbe de service », nous décidons de la noter $\trianglelefteq$ (Le symbole $\triangleleft$ sera utilisé pour les courbes de service strict).

$$S \trianglerighteq \beta \iff \forall R, \forall R' \text{ tel que } R \xrightarrow{S} R',\ R' \geq R * \beta \tag{11.7}$$

Les principaux résultats de calcul réseau se décrivent assez simplement dans ce cadre formel. Posons $R, R' \in \mathcal{F}_0$, $S, S' \in \mathcal{S}_1$, $\alpha, \beta \in \mathcal{F}$. La règle (11.8) est fondamentale en calcul réseau, car elle montre l'on peut calculer la courbe d'arrivée en sortie d'un flux à partir de la courbe d'arrivée en entrée et de la courbe de service, et donc propager les calculs. L'équation (11.9), elle, nous dit que l'on peut toujours sous-appproximer une courbe de service.

$$S \trianglerighteq \beta, R \prec \alpha, R \xrightarrow{S} R' \implies R' \prec \alpha \oslash \beta \tag{11.8}$$

$$S \trianglerighteq \beta, \beta \geq \beta' \implies S \trianglerighteq \beta' \tag{11.9}$$

11.2.2. Délais et occupation mémoire

L'expérience montre qu'il existe une confusion chez les débutants en calcul réseau entre les bornes réelles, expérimentées dans un système, et les bornes calculées en calcul réseau sur la modélisation faite du système.

Là encore, nous espérons qu'en définissant formellement les notions, avec des notations distinctes, les confusions soient levées.

Définitions des délais et occupation mémoire

Commençons par définir la fonction d de délai subi par un flux dans un serveur, et b la fonction d'occupation mémoire. Nous avons besoin pour cela de d'abord définir deux objets mathématiques : la déviation horizontale et la déviation verticale entre deux courbes

4. Nous verrons au paragraphe 11.2.5 les serveurs à plusieurs entrées et plusieurs sorties.

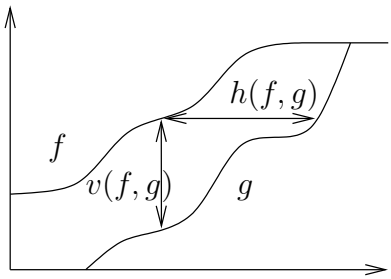


FIGURE 11.2 : Déviations horizontales et verticales.

f et g.

$$h(f,g) \stackrel{\text{def}}{=} \inf\{d \mid \forall t, f(t) \leq g(t+d)\} \qquad v(f,g) \stackrel{\text{def}}{=} \sup_{t\geq 0}\{f(t)-g(t)\}$$

Comment définir le temps d'attente à partir des courbes d'entrée R et de sortie R′ ? Faisons une analogie avec une file de supermarché. Le 50e client à entrer dans la file sortira quand le compte des clients sortis atteindra 50. C'est-à-dire que le client arrivé à la date t sortira lorsque la courbe de sortie $\mathrm{R}'(t)$ aura rattrapé la courbe d'entrée $\mathrm{R}(t)$. Le temps d'attente sera donc d tel que $\mathrm{R}(t) = \mathrm{R}'(t+d)$. En généralisant cette idée à un modèle fluide et à des fonctions non continues, cela devient $\mathrm{R}(t) \leq \mathrm{R}'(t+d)$. Et on définit le délai (maximal) entre R et R′ comme le plus grand délai individuel, c'est-à-dire $h(\mathrm{R},\mathrm{R}')$.

Soulignons que la présentation informelle que nous avons faite (50e entré, 50e sorti) suppose une politique FIFO et une vision individualisée des bits de données. Nous garderons cette définition même dans un cadre non FIFO en considérant que dans un flux de données, les bits ne peuvent pas être individualisés (puisque nous ne prenons en compte que leur nombre cumulé $\mathrm{R}(t)$), et que le délai est donc le temps nécessaire pour que le cumul en sortie rattrape le cumul en entrée. Dans [Chang 2000], on parle dans ce cas de *délai virtuel*. Des résultats sur les serveurs partagés suivant une politique de service seront présentés au paragraphe 11.2.5.

Une fois défini le délai pour un couple (R,R'), on peut définir le délai (maximal) $d(\mathrm{R},\mathrm{S})$ induit par un serveur S sur un flux en entrée R comme le maximum pour toutes les sorties possibles du serveur.

On définit de même l'occupation mémoire $b(\mathrm{R},\mathrm{S})$ (de l'anglais *backlog*) à partir de la déviation verticale.

$$d(\mathrm{R},\mathrm{R}') \stackrel{\text{def}}{=} h(\mathrm{R},\mathrm{R}') \qquad d(\mathrm{R},\mathrm{S}) \stackrel{\text{def}}{=} \sup\left\{d(\mathrm{R},\mathrm{R}') \,\middle|\, \exists \mathrm{R}',\ \mathrm{R} \xrightarrow{\mathrm{S}} \mathrm{R}'\right\}$$

$$b(\mathrm{R},\mathrm{R}') \stackrel{\text{def}}{=} v(\mathrm{R},\mathrm{R}') \qquad b(\mathrm{R},\mathrm{S}) \stackrel{\text{def}}{=} \sup\left\{b(\mathrm{R},\mathrm{R}') \,\middle|\, \exists \mathrm{R}',\ \mathrm{R} \xrightarrow{\mathrm{S}} \mathrm{R}'\right\}$$

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

Les résultats du calcul réseau sont que, connaissant une courbe d'arrivée et une courbe de service, on peut calculer une borne maximale sur le délai d et l'occupation mémoire b.

$$\mathrm{S} \trianglerighteq \beta, \mathrm{R} \prec \alpha \implies d(\mathrm{R},\mathrm{S}) \leq h(\alpha,\beta) \tag{11.10}$$

$$\mathrm{S} \trianglerighteq \beta, \mathrm{R} \prec \alpha \implies b(\mathrm{R},\mathrm{S}) \leq v(\alpha,\beta) \tag{11.11}$$

Il est connu que les bornes calculées ainsi sont justes (*tight* en anglais), c'est-à dire-atteignables [Le Boudec 2001, sec. 1.4.2] mais pas forcément atteintes. Le paragraphe 11.3 traite plus précisément de ce point. Pour atteindre les bornes prédites par un couple (α,β), on construit un couple (flux, serveur) qui atteint les bornes.

$$\forall\alpha, \forall\beta, \exists \mathrm{S}_\beta, \exists \mathrm{R}_\alpha, d(\mathrm{R}_\alpha, \mathrm{S}_\beta) = h(\alpha,\beta) \tag{11.12}$$

$$\forall\alpha, \forall\beta, \exists \mathrm{S}_\beta, \exists \mathrm{R}_\alpha, x(\mathrm{R}_\alpha, \mathrm{S}_\beta) = v(\alpha,\beta) \tag{11.13}$$

11.2.3. Courbes de service strict, courbes de service maximal

À partir de ces principes de base, nous pouvons introduire d'autres notions, comme les courbes de service strict ($\triangleright$), les courbes de service maximal ($\beta_\mathrm{U} \trianglerighteq_\mathrm{U} \mathrm{S} \trianglerighteq_\mathrm{L} \beta_\mathrm{L}$).

Courbe de service stricte

La notion de courbe de service stricte se révélera importante lors que nous étudierons la question des serveurs partagés. On dit qu'un serveur offre une courbe de service strict si, sur toute période chargée, la sortie du serveur est au moins égale à la courbe de service strict. On peut définir formellement un intervalle I, ouvert ou fermé à chaque bord, comme une période chargée (*backlogged period*) si et seulement si $\forall t \in \mathrm{I},\ \mathrm{R}'(t) - \mathrm{R}(t) > 0$. Notons $\mathcal{BP}$ l'ensemble des périodes chargées. On peut alors définir une courbe de service strict $(\mathrm{S} \triangleright \beta)$ comme le fait que pour deux instants dans une période chargée $(t, t' \in \mathrm{I})$, la sortie est au moins égale à la courbe de service sur cette durée $(\mathrm{R}'(t') - \mathrm{R}'(t) \geq \beta(t'-t))$.

$$\mathcal{BP}_{\mathrm{R},\mathrm{R}'} \stackrel{\text{def}}{=} \{\mathrm{I} \in \mathcal{I}_{\geq 0} \,|\, \forall t \in \mathrm{I},\ \mathrm{R}(t) > \mathrm{R}'(t)\} \tag{11.14}$$

$$\begin{aligned} \mathrm{S} \triangleright \beta \stackrel{\text{def}}{\iff} \forall \mathrm{R}, \mathrm{R}' \in \mathcal{F}_0 \text{ tels que } \mathrm{R} \stackrel{\mathrm{S}}{\to} \mathrm{R}',\ \forall \mathrm{I} \in \mathcal{BP}_{\mathrm{R},\mathrm{R}'}, \\ \forall t, t' \in \mathrm{I},\ \mathrm{R}'(t') - \mathrm{R}'(t) \geq \beta(t'-t) \end{aligned} \tag{11.15}$$

Courbes de service minimal et maximal

La notion de courbe de service minimal peut se compléter par celle de courbe de service maximal, qui donne une borne sur le meilleur service que peut rendre un serveur. La définition est symétrique de celle de la courbe de service définie en (11.7), qui devient dès lors « minimale ». Le même symbole $\trianglelefteq$ représentant deux relations distinctes, de même domaine similaire ($\mathcal{F}_0 \times \mathcal{S}_1$ ou $\mathcal{S}_1 \times \mathcal{F}_0$), on distingue ces notations : $\trianglelefteq_U$ sera utilisé pour le service maximal (*upper*) tandis que $\trianglerighteq_L$ sera utilisé pour le service minimal (*lower*). Notons que quand l'indice est omis, il s'agit toujours de la courbe de service minimal (en particulier, quand seul le service minimal est considéré). Cette introduction est intéressante car elle permet de préciser la courbe en sortie d'un serveur (règle (11.18)) : si on connaît la courbe de service maximal, on peut calculer une courbe de sortie plus précise [5].

$$S \trianglelefteq_U \beta \overset{\text{def}}{\iff} \forall R, \forall R', \text{ tel que } R \xrightarrow{S} R', R' \leq R * \beta \tag{11.16}$$

$$\forall S,\ S \trianglelefteq_U \delta_0 \tag{11.17}$$

$$\implies R' \prec_L (\alpha * \beta_U) \oslash \beta_L \tag{11.18}$$

11.2.4. Séquence de serveurs

Une fois formalisées les notions de serveur et de délai, nous pouvons formaliser la notion de mise en séquence de serveurs. Un célèbre résultat de calcul réseau, dit *pay bursts only once*, permet de remplacer une séquence de deux serveurs par un pseudo-serveur équivalent. La compréhension de ce qu'est ce « pseudo-serveur », qui n'existe pas, mais que l'on utilise pour améliorer des calculs, est parfois difficile. Avec notre formalisation, qui modélise un serveur comme une relation entre courbe d'entrée et de sortie, la mise en séquence de serveurs est tout simplement la composition des deux relations. Nous pouvons définir un opérateur de mise en séquence sur $\mathcal{S}_1 \times \mathcal{S}_1$, noté $\circ$. Le pseudo-serveur est donc un serveur qui entre dans la même classe $\mathcal{S}_1$ que les autres.

$$(S \circ S')(R) \overset{\text{def}}{=} \left\{ R'' \,\middle|\, \exists R', R \xrightarrow{S} R' \xrightarrow{S'} R'' \right\} \tag{11.19}$$

Le résultat *pay bursts only once* s'énonce alors simplement : on sait calculer une courbe de service $\beta * \beta'$ pour la mise en séquence de deux serveurs à partir de leurs courbes de service respectives β et β'.

$$S \trianglerighteq \beta, S' \trianglerighteq \beta' \implies (S \circ S') \trianglerighteq \beta * \beta' \tag{11.20}$$

5. En effet $\alpha * \beta_U \leq \alpha$, ce qui donne $(\alpha * \beta_U) \oslash \beta_L \leq \alpha \oslash \beta_L$.

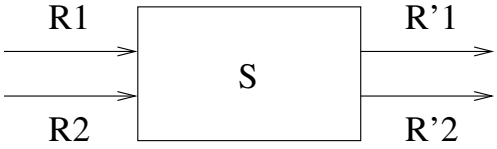


FIGURE 11.3 : Serveur partagé par deux flux

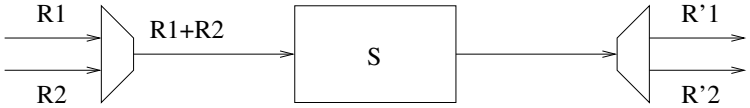


FIGURE 11.4 : Usage de multiplexeur/démultiplexeur

Ce résultat est important car le délai de bout en bout $d(R, S \circ S')$ est inférieur à la somme des délais individuels $d(R, S \circ S') \leq d(R, S) + d(R', S')$, et en pratique, le gain peut être important.

11.2.5. Serveur partagé

Dans la réalité, on n'utilise pas (ou pas souvent) un élément pour traiter un unique flux. Un serveur est partagé par plusieurs flux, et sa puissance est partagée suivant des politiques de service.

Nous allons dans ce paragraphe montrer que les notions de multiplexeurs/démultiplexeurs, naturelles en réseau, se modélisent mal dans ce cadre théorique, et sont en fait inutiles.

Multiplexer ou pas ?

Le calcul réseau possède des résultats pour traiter ces configurations. Par exemple, pour la configuration présentée dans la figure 11.3, on sait que si le serveur S offre une courbe de service strict β, en l'absence de toute autre information sur la politique d'ordonnancement (*blind multiplexing*), chaque flux i se voit offrir le service individuel $\beta_i = [\beta - \alpha_j]^+$, avec $i \neq j$, sous réserve que $\beta_i \in \mathcal{F}$. Le serveur S n'a donc pas une seule entrée, comme dans le cas présenté à la figure 11.1 et dans le paragraphe 11.2.1, et ne peut donc pas être un élément de $\mathcal{S}_1$. On peut essayer de se ramener à une entré unique en utilisant un multiplexeur et un démultiplexeur, comme sur la figure 11.4. Mais si la modélisation du multiplexeur est assez simple (il fait la somme des courbes de trafic), c'est le démultiplexeur qui pose problème. Comment reconstruire R'_2 et R'_1 uniquement à partir de leur somme ? C'est impossible.

D'un point de vue formel, un tel multiplexeur n'a pas d'intérêt. Bien sûr, dans le cadre d'un outil graphique qui permettrait de concevoir des réseaux, il faut offrir ce type d'élément, mais d'un point de vue formel, cela se ramène à la figure 11.5, et sa formalisation ne ferait que compliquer le cadre formel.

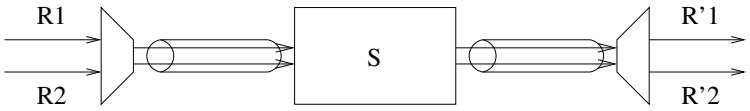


FIGURE 11.5 : Autre point de vue sur multiplexeur/démultiplexeur

Un serveur devient donc élément avec n entrées, et autant de sorties. C'est donc une fonction de l'ensemble $\mathcal{F}_0^n \to \mathcal{P}(\mathcal{F}_0^n)$, que nous noterons $\mathcal{S}_n$ qui relie un vecteur de courbes d'entrée $\mathbf{R}$ à un autre vecteur $\mathbf{R}'$ de courbes de sortie.

Là où la formalisation s'oppose au sens physique, c'est qu'un composant réseau est généralement une boîte avec n ports physique en entrée, et ce nombre est intrinsèque à la boîte, et ne dépend pas de son usage. Dans notre formalisation, le nombre d'entrée d'un composant est le nombre de flux qu'il manipule, et il est donc dépendant de la topologie.

L'ensemble des serveurs est donc $\mathcal{S} \stackrel{\text{def}}{=} \bigcup_{n>0} \mathcal{S}_n$, et $\mathcal{S}_1$ n'est qu'un cas particulier intéressant.

Nous pouvons alors formellement définir dans la règle (11.21) la notion de courbe de service pour un élément de $\mathcal{S}_n$ de la même façon que pour $\mathcal{S}_1$ (cf (11.7)), comme étant le service sur la somme des flux $\|\mathbf{R}\|$, et de même pour la courbe de service strict (11.22). Rappelons que pour tout n, si $\mathbf{R} \in \mathcal{F}_0^n$, alors $\|\mathbf{R}\| \in \mathcal{F}_0$.

$$\mathrm{S} \trianglerighteq \beta \iff^{\text{def}} \forall \mathbf{R}, \forall \mathbf{R}', \text{ tel que } \mathbf{R} \stackrel{\mathrm{S}}{\to} \mathbf{R}', \|\mathbf{R}'\| \geq \|\mathbf{R}\| * \beta \tag{11.21}$$

$$\begin{aligned} \mathrm{S} \triangleright \beta \iff^{\text{def}} \forall \mathbf{R}, \mathbf{R}', \text{ tel que } \mathbf{R} \stackrel{\mathrm{S}}{\to} \mathbf{R}', \forall \mathrm{I} \in \mathcal{BP}_{\|\mathbf{R}\|,\|\mathbf{R}'\|}, \forall t, t' \in \mathrm{I}, \\ \|\mathbf{R}'\|(t') - \|\mathbf{R}'\|(t) \geq \beta(t'-t) \end{aligned} \tag{11.22}$$

Politiques de services et réduction

Nous venons de définir le service global d'un serveur partagé par plusieurs flux. Il faut maintenant étudier la façon dont la politique de service permet de regarder le service offert à chaque flux individuel, et comment formaliser les résultats déjà présents dans la littérature et aider au développement de nouveaux.

Pour illustrer le problème, nous allons reprendre deux résultats (liés entre eux) sur les politiques de priorités fixes et sur l'ordonnancement aveugle. Soit un serveur $\mathrm{S} \in \mathcal{S}_2$ partagé par deux flux R_1 et R_2 ($(\mathrm{R}_1, \mathrm{R}_2) \stackrel{\mathrm{S}}{\to} (\mathrm{R}'_1, \mathrm{R}'_2)$), et de courbe de service stricte β ($\mathrm{S} \triangleright \beta$). Alors, sans hypothèse sur la politique de service (*blind multiplexing*) on sait [Le Boudec 2001, théorème 6.2.1] que chaque flux R_i reçoit le service résiduel $[\beta - \alpha_j]^+$ (avec $i \neq j$, cf (11.23)). Le second dit que dans le cas d'un partage par une priorité stricte (non préemptif), le flux le plus prioritaire reçoit le service $[\beta - l_j^m]^+$, où l_j^m est la taille du plus grand message du flux le moins prioritaire, et le moins prioritaire reçoit le même

service résiduel que dans le cas d'un ordonnancement aveugle (équation 11.24).

$$\left.\begin{array}{c}(\mathrm{R}_1,\mathrm{R}_2)\xrightarrow{\mathrm{S}}(\mathrm{R}_1',\mathrm{R}_2')\\ \mathrm{S}\triangleright\beta\\ \{i,j\}=\{1,2\}\\ [\beta-\alpha_i]^+\in\mathcal{F}\end{array}\right\}\Longrightarrow\exists\mathrm{S}_i,\mathrm{R}_i\xrightarrow{\mathrm{S}_i}\mathrm{R}_i',\mathrm{S}_i\trianglerighteq[\beta-\alpha_i]^+ \tag{11.23}$$

$$\left.\begin{array}{c}(\mathrm{R}_1,\mathrm{R}_2)\xrightarrow{\mathrm{S}}(\mathrm{R}_1',\mathrm{R}_2')\\ \mathrm{S}\triangleright\beta\\ \mathrm{R}_1\text{ plus prioritaire que }\mathrm{R}_2\\ \text{Politique SP, non préemptif}\\ [\beta-\alpha_1]^+\in\mathcal{F}\end{array}\right\}\Longrightarrow\begin{cases}\exists\mathrm{S}_1,\mathrm{R}_1\xrightarrow{\mathrm{S}_1}\mathrm{R}_1',\mathrm{S}_1\trianglerighteq[\beta-l_2^{\mathrm{M}}]^+\\ \exists\mathrm{S}_2,\mathrm{R}_2\xrightarrow{\mathrm{S}_2}\mathrm{R}_2',\mathrm{S}_2\trianglerighteq[\beta-\alpha_1]^+\end{cases} \tag{11.24}$$

avec l_2^m la taille maximale de trame du second flux.

La question que nous nous posons est : comment passer formellement à un serveur à n entrées ? Peut-il exister un méta-théorème qui permette de décrire toutes les politiques de service avec un minimum de flux (deux dans le cas de (11.23) et (11.24)), et de ramener le cas général à n entrées à un système réduit ? Ou faut-il, pour chaque politique de service, écrire la version à n entrées ?

Nous allons montrer avec des exemples que cette envie de méta-théorème multi-politique de service semble naturelle quand on ne considère que des cas simples, mais est difficilement généralisable.

Si un serveur S possède trois flux $\mathrm{R}_a,\mathrm{R}_b,\mathrm{R}_c$ en entrée de courbes d'arrivée respectives $\alpha_a,\alpha_b,\alpha_c$, on peut appliquer (11.23) au flux R_a en considérant que c'est « comme si » il partageait S avec un seul autre flux agrégé $\mathrm{R}_b+\mathrm{R}_c$ (de courbe d'arrivée $\alpha_b+\alpha_c$). Mais ce « comme si » qui semble naturel reste informel.

Si le serveur applique une politique à priorités fixes, on peut vouloir appliquer (11.24). Regardons différents cas : si R_a est le plus prioritaire, et R_b et R_c se partagent le niveau le moins prioritaire, on peut vouloir agréger $\mathrm{R}_b+\mathrm{R}_c$ et se ramener au cas à deux entrées. Pour le faire formellement, il faudrait soit ré-écrire (11.24) dans le cas général à n entrées, soit un méta-théorème qui dise que l'on peut regrouper les flux d'une même classe de service, avec une définition formelle de « classe de service » et se ramener d'un serveur de S_n à S_{n-1}, jusqu'au S_2 utilisé dans (11.24).

Mais on peut vouloir faire plus : si R_b et R_c ont des niveaux de priorité différents mais restent tout deux moins prioritaires que R_a, on peut aussi vouloir les agréger alors qu'ils sont dans deux classes de service différentes. Le méta-théorème devrait alors agréger des classes de service « équivalentes » du point de vue de celui que l'on cherche à étudier de façon à réduire le nombre d'entrées.

Mais toutes les politiques de services ne peuvent pas ainsi réduire leur nombre d'entrées. Considérons par exemple la politique d'arbitrage FTDMA entre priorités utilisée dans

le segment dynamique de FlexRay [FlexRay 2005, 5.1.4]. Chaque flux dans cette partie se voit attribuer statiquement un mini-slot, qu'il peut transformer en slot de message. Les mini-slots sont ordonnés, ce qui induit une priorité entre flux. La particularité de ce protocole dans notre contexte est qu'un système à trois niveaux de priorités ne peut pas, du point de vue du moins prioritaire, se réduire à un système à deux priorités où les deux priorités les plus hautes seraient agrégées, car le flux le moins prioritaire est retardé d'au moins deux mini-slots dans le système à trois niveaux, et seulement d'un mini mini-slot dans le système réduit, ce qui n'est pas équivalent. Nous n'affirmons pas par là qu'on ne peut pas étudier un système réduit (il suffit dans notre exemple de doubler le mini-slot agrégé), mais que la méthode sera différente dans les détails de celle utilisable pour l'ordonnancement aveugle ou les priorités fixes.

En conclusion, alors que la simplicité d'usage amène à étudier chaque politique de service avec un minimum de flux d'entrées (un par classe de service), il ne peut pas exister de méta-théorème formel qui permette pour toute politique de service de se ramener à un cas réel avec de nombreuses entrées à un système minimal. Il faut donc soit ré-écrire les théorèmes s'appliquant à une politique de service de façon à ce qu'ils traitent un nombre quelconque d'entrées, soit définir ces résultats de réduction politique par politique.

Ceci étant dit, le mieux étant parfois l'ennemi du bien, nous ne militons pas forcément pour une formalisation systématique de ce type de résultats : il n'est pas sûr que la formalisation de l'agrégation de flux dans le cas de politiques de services à priorité stricte aide sa manipulation ou sa compréhension. Pour des politiques plus complexes, cela pourrait l'être.

11.3. Précisions sur les bornes calculées

Nous avons mentionné au paragraphe 11.2.2 que les bornes calculées étaient atteignables mais pas nécessairement atteintes. Nous donnons ici quelques explications à ce fait.

Trois phénomènes entrent en jeu pour que les bornes soient effectivement atteintes : la modélisation manque de précision, les résultats du calcul réseau ne donnent pas des bornes atteignables et certains comportements ne peuvent pas être modélisés.

Premièrement, il ne faut pas oublier que les courbes calculées le sont par rapport à un système réel, et que les courbes ne sont donc qu'une modélisation de la réalité. Comme on veut calculer des performances pire-cas, les contraintes calculées ne sont pas toujours exactes mais sont des approximations supérieures pour le processus des arrivées et des approximations inférieures pour le service garanti. Par exemple, pour pouvoir calculer des performances assez rapidement, les courbes d'arrivées choisies sont en général des fonctions affines, ou le minimum de deux fonctions affines. Modéliser un comportement périodique par une courbe affine entraîne nécessairement une perte de précision, comme

le montre la figure 11.6. De même pour modéliser un serveur, on utilisera principalement des courbes de type *rate-latency* $\beta_{R,T} : t \mapsto R(t-T)^+$ ou le maximum de deux — voire trois — telles courbes.

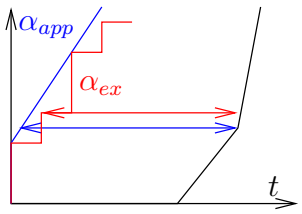


FIGURE 11.6 : Approximation de la courbe d'arrivée. Le processus des arrivées est périodique, et il arrive à chaque unité de temps alternativement un paquet de taille 2 et un paquet de taille 1. La courbe d'arrivée exacte α_{ex} est périodique (en rouge) et peut être approximée supérieurement par la bleue α_{app}, qui est affine. Les bornes obtenues pour le délai (distance horizontale à courbe de service) ne sont pas les mêmes

Une deuxième cause est intrinsèque au calcul réseau et vient des résultats disponibles eux-mêmes. Les courbes d'arrivée et de service peuvent avoir été calculées via les techniques connues du calcul réseau. Hormis les cas très simples (un seul serveur ou un seul flux), les bornes calculées ne sont pas atteignables. L'exemple le plus simple de calcul de borne non juste est le cas de deux serveurs en tandem traversées par deux flux présenté dans [Schmitt 2008b]. Il existe deux manières de calculer le service disponible pour l'un des deux flux dans ce système, et aucune de ces méthodes ne permet un calcul exact des bornes dans tous les cas. La question de savoir s'il existe une manière de calculer une courbe résiduelle qui donne toujours des bornes exactes est encore ouverte. Le même problème se pose si la politique de service est FIFO [Bisti 2008].

La troisième cause de borne non atteinte est le modèle de serveur choisi : nous calculons les bornes à partir des courbes de service. Mais soit une courbe $\beta \in \mathcal{F}_0$, on peut construire le serveur S définit par $S(R) = \{R' \mid \exists \varepsilon > 0 \text{ tel que } R' \geq (\beta * R + \varepsilon) \wedge R\}$. On a bien sûr $S \trianglerighteq \beta$. Comme S(R) ne contient pas *toutes* les courbes R′ telles que $R' \geq \beta * R$, mais un sous-ensemble strict infiniment proche, les bornes calculées $d(R,S)$ et $b(R,S)$ ne seront jamais atteintes. Mais surtout, *on ne peut pas modéliser plus finement* : il ne peut pas exister de courbe β′ plus grande que β qui soit aussi une courbe de service pour S.

Conclusion

De notre point de vue, les notations et formalisations des principaux objets (courbe de trafic, courbe d'arrivée, courbe de service simple et strict, délai, bornes) permettent de clarifier les notions fondamentales du calcul réseau. Par contre la gestion de topologies complexes, avec de multiples serveurs manipulant de multiples flux, reste insatisfaisante.

De manière générale, nous souhaitons par la diffusion de cette proposition solliciter des réactions et discussions avec le reste de la communauté.

CHAPITRE 12

PERSPECTIVES

En me replongeant dans la bibliographie traitant du Network Calculus, j'ai découvert ou redécouvert avec plaisir les nombreuses avancées qui ont eu lieu en deux ans, en particulier :

- l'impressionnant *survey* de Markus Fidler, qui est une référence à part entière ;
- le Network Calculus Stochastique, qui a été utilisé avec succès par plusieurs groupes de recherche ;
- voir le NC utilisé comme un outil pour étudier des systèmes, ce qui le rapproche de la théorie de la résiduation, dont nous avons encore beaucoup à apprendre.

Il me semble cependant important de pouvoir modéliser les politiques de services dans le cadre du Network Calculus, mais aussi de prendre en compte chaque bit « individuellement ». Évidement, on ne souhaite pas, en NC, étudier toutes les trajectoires possibles pour un bit donné.

On pourrait pour ce faire essayer d'étudier non pas des courbes « quantité de données en fonction du temps » mais par exemple des enveloppes représentant — pour tout bit arrivant dans un noeud — un encadrement du « retard » qu'il peut subir en fonction de la charge actuelle du serveur :

- soit un retard en termes de temps (un délai donc, avec un modèle fluide) ;
- soit un retard en termes de nombre de paquets (modèle discret, qui modélise d'une certaine manière la politique de service).

De telles courbes pourraient, peut-être, s'étendre au cas MIMO. Au niveau de chaque flux, on aurait un encadrement (minimum et maximum) sur le temps perdu par un bit qui arrive, en fonction de la charge pour chacun des flux du serveur. Ces objets mathématiques seraient sûrement plus compliqués à manipuler, mais c'est ce qui est déjà fait parfois avec des matrices (attention, les propriétés mathématiques des objets étudiés peuvent ne plus être les mêmes en dimension n, par exemple le dioïde $((\mathbb{R} \cup \{+\infty\})^n, min, +)$ n'est plus sélectif).

Ce serait une manière d'essayer de modéliser (très grossièrement) la politique de service directement avec des outils algébriques. En effet, un bit arrivant à un instant t dans un serveur peut voir son temps de traversée énormément varier en fonction de la quantité de données dans sa file d'attente (ou dans la file globale dans le cas SISO), mais surtout d'après la politique de service (qu'on a du mal à modéliser en MIMO, sans agréger les flux). Dans un tout autre contexte, certains auteurs étudient pour des réseaux déterministes les écarts de temps de traversée pour deux bits arrivant au même instant [Raz 2008].

Il me semble en effet qu'il manque au Network Calculus la possibilité de prendre en compte de manière générique et directement dans le formalisme — même très sommairement — les politiques de service. Actuellement, les politiques de service sont en effet traitées soit de manière *ad hoc* (FIFO, priorités fixes), soit de manière générique (multiplexage aveugle) mais sans prendre en compte les spécificités des politiques de service.

Remarque 12.1. *Encore un gros MERCI à Éric Thierry, Anne Bouillard et Marc Boyer ; ainsi qu'à mes amis et ma famille : en particulier à Monique Rouillon et Anne Reverdy.*

BIBLIOGRAPHIE

[Andrews 1996] M. Andrews, B. Awerbuch, A. Fernández, T. Leighton, Z. Liu & J. M. Kleinberg. *Universal-stability results for greedy contention-resolution protocols*. In Proc. of FOCS'96, pages 380–389, 1996. 41

[Andrews 2000] M. Andrews. *Instability of FIFO in session-oriented networks*. In Proceedings of SODA'00, 2000. 41

[Andrews 2001] M. Andrews, B. Awerbuch, A. Fernández, T. Leighton, Z. Liu & J. M. Kleinberg. *Universal-stability results and performance bounds for greedy contention-resolution protocols*. J. ACM, vol. 48, no. 1, pages 39–69, 2001. 41

[Andrews 2007] M. Andrews. *Instability of FIFO in the permanent sessions model at arbitrarily small network loads*. In Proceedings of SODA'07, 2007. 17, 41

[Baccelli 1992] F. Baccelli, G. Cohen, G. Y. Olsder & J. P. Quadrat. Synchronization and linearity. Wiley, 1992. 16, 22, 23, 24, 41, 160

[Bauer 2010] H. Bauer, J.-L. Scharbarg & C. Fraboul. *Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach*. IEEE Transactions on Industrial Informatics, vol. 6, no. 4, pages 521–533, 2010. 40

[Bisti 2008] L. Bisti, L. Lenzini, E. Mingozzi & G. Stea. *Estimating the Worst-case Delay in FIFO Tandems Using Network Calculus*. In Proceedings of Valuetools'2008, 2008. 17, 170

[Bisti 2010] L. Bisti, L. Lenzini, E. Mingozzi & G. Stea. *DEBORAH: A Tool for Worst-Case Analysis of FIFO Tandems*. In ISoLA, pages 152–168, 2010. 42

[Boissonnat 2001] J.-D. Boissonnat & M. Yvinec. Algorithmic geometry. Cambridge University Press, 2001. 119

[Borodin 2001] A. Borodin, J. M. Kleinberg, P. Raghavan, M. Sudan & D. P. Williamson. *Adversarial queuing theory*. J. ACM, vol. 48, no. 1, pages 13–38, 2001. 17

[Boudec 2012] J.-Y. Le Boudec. *The System Theory of Network Calculus*. http://icawww1.epfl.ch/PS_files/woneca2012.pdf, 2012. WoNeCa 2012, Kaiserslautern, March 21, 2012. 41

[Bouillard 2007a] A. Bouillard, B. Gaujal, S. Lagrange & E. Thierry. *Optimal routing for end-to-end guarantees: the price of multiplexing*. In Proceedings of Valuetools'07, 2007. 88, 115, 116, 117, 121, 149

[Bouillard 2007b] A. Bouillard & E. Thierry. *An algorithmic toolbox for network calculus.* Rapport technique, Technical 6094, IRISA, 2007. 58, 59, 62, 64, 65, 71, 75

[Bouillard 2007c] A. Bouillard & E. Thierry. *Some examples and counterexamples for* (min, +) *filtering operations.* Rapport technique, Technical 6095, IRISA, 2007. 43, 49, 50

[Bouillard 2008a] A. Bouillard, B. Gaujal, S. Lagrange & E. Thierry. *Optimal routing for end-to-end guarantees using Network Calculus*. Rapport technique, INRIA, 2008. 88, 115, 121, 123

[Bouillard 2008c] A. Bouillard & E. Thierry. *An Algorithmic Toolbox for Network Calculus*. Discrete Event Dynamic Systems, vol. 18, no. 1, pages 3–49, 2008. 43, 49, 116, 149

[Bouillard 2009a] A. Bouillard, B. Cottenceau, B. Gaujal, L. Hardouin, S. Lagrange & M. Lhommeau. *COINC library: a toolbox for the network calculus: invited presentation, extended abstract*. In VALUETOOLS, page 35, 2009. 41

[Bouillard 2009c] A. Bouillard, L. Jouhet & E. Thierry. *Tight performance bounds in the worst-case analysis of feed-forward networks*. Rapport technique 7012, INRIA, August 2009. 141

[Bouillard 2010] A. Bouillard, L. Jouhet & E. Thierry. *Tight performance bounds in the worst-case analysis of feed-forward networks*. In Proceedings of Infocom'2010, 2010. 95, 124, 145

[Boyer 2008] M. Boyer & C. Fraboul. *Tightening end to end delay upper bound for AFDX network calculus with rate latency FCFS servers using network calculus*. In Proceedings of the 7th IEEE International Workshop on Factory Communication Systems Communication in Automation (WFCS'2008), 2008. 16

[Boyer 2010a] M. Boyer. *NC-maude: a rewriting tool to play with network calculus*. Leveraging Applications of Formal Methods, Verification, and Validation, pages 137–151, 2010. 43

[Boyer 2010b] M. Boyer, N. Navet, X. Olive & E. Thierry. *The PEGASE project: precise and scalable temporal analysis for aerospace communication systems with Network Calculus*. Leveraging Applications of Formal Methods, Verification, and Validation, pages 122–136, 2010. 38, 42, 50

[Boyer 2011] Marc Boyer, Jörn Migge & 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 '11. ACM, 2011. 50, 65

[Chakraborty 2003] S. Chakraborty, S. Künzli, L. Thiele, A. Herkersdorf & P. Sagmeister. *Performance Evaluation of Network Processor Architectures: Combining Simulation with Analytical Estimation*. Computer Networks, vol. 41, no. 5, pages 641–665, 2003. 16, 78

[Chang 1997] C. S. Chang. *A filtering theory for deterministic traffic regulation*. In Proc. of INFOCOM'97, pages 436–443, 1997. 16

[Chang 1999] C. S. Chang. *Deterministic traffic specifications via projections under the min-plus algebra*. In Proceedings of INFOCOM'99, pages 43–50, 1999. 153

[Chang 2000] C. S. Chang. Performance guarantees in communication networks. Telecommunication Networks and Computer Systems. Springer, 2000. 15, 25, 30, 32, 37, 77, 95, 98, 159, 160, 161, 163

[Charara 2005] H. Charara & C. Fraboul. *Modelling and simulation of an avionics full duplex switched ethernet*. In Telecommunications, 2005. advanced industrial conference on telecommunications/service assurance with partial and intermittent resources conference/e-learning on telecommunications workshop. aict/sapir/elete 2005. proceedings, pages 207–212. IEEE, 2005. 40, 43

[Charny 2000] A. Charny & J.-Y. Le Boudec. *Delay bounds in a network with aggregate scheduling*. In Proceedings of QoFIS'00, volume LNCS 1922, pages 1–13. Springer-Verlag, 2000. 41

[Cholvi 2007] V. Cholvi & J. Echagüe. *Stability of FIFO networks under adversarial models: State of the art*. Computer Networks, vol. 51, no. 15, pages 4460–4474, 2007. 41

[Ciucu 2005] F. Ciucu, A. Burchard & J. Liebeherr. *A network service curve approach for the stochastic analysis of networks*. In Proceedings of SIGMETRICS'2005, 2005. 40, 77

[COINC] COINC. *COmputational Issues in Network Calculus*. http://perso.bretagne.ens-cachan.fr/~bouillar/coinc. 41, 78

[Cormen 2001] T. H. Cormen, C. E. Leiserson, R. L. Rivest & C. Stein. Introduction to algorithms. MIT Press and McGraw-Hill, 2001. 122

[Cottenceau 2000] B. Cottenceau, L. Hardouin, M. Lhommeau & J.-L. Boimond. *Data processing tool for calculation in dioid*. In WODES 2000, Gand, 2000. http://www.istia.univ-angers.fr/~hardouin/outils.html. 42

[Cottenceau 2001] B. Cottenceau, B. Gruet, L. Hardouin & M. Lhommeau. *Modèles et Systèmes Dynamiques*. http://www.istia.univ-angers.fr/~hardouin/outils.html, 2001. 42

[Cruz 1991a] R. L. Cruz. *A calculus for network delay, Part I: Network elements in isolation*. IEEE Transactions on Information Theory, vol. 37, no. 1, pages 114–131, 1991. 15, 16, 21, 160

[Cruz 1991b] R. L. Cruz. *A calculus for network delay, Part II: Network analysis*. IEEE Transactions on Information Theory, vol. 37, no. 1, pages 132–141, 1991. 15, 16, 17, 160

[CyNC] CyNC. *A Tool for Performance Analysis of Complex Real Time Systems*. http://www.control.auc.dk/~henrik/CyNC. 41

[Daniel-Cavalcante 2006] M. Daniel-Cavalcante, M. F. Magalhaes & R. Santos-Mendes. *Max-Plus: A Network Algebra*. In C. Commault & N. Marchand,

editeurs, Positive Systems, volume 341 of *LNCIS*, pages 375–382. Springer-Verlag, 2006. 16

[Daniel-Cavalcante 2008] M. Daniel-Cavalcante & R. Santos-Mendes. *Diod of Formal Power Series for the Determination of Performance Parameters of Communication Networks*. In Proceedings of WODES'2008, 2008. 16

[Diaz 2004] J. L. Diaz, J. M. Lopez, M. Garcia, A.M. Campos, K. Kim & L.L. Bello. *Pessimism in the stochastic analysis of real-time systems: Concept and applications*. In Real-Time Systems Symposium, 2004. Proceedings. 25th IEEE International, pages 197–207. IEEE, 2004. 40

[DISCO] DISCO. *The DISCO Network Calculator*. http://disco.informatik.uni-kl.de/content/Downloads. 41

[Fidler 2003] M. Fidler. *Extending the network calculus pay bursts only once principle to aggregate scheduling*. Quality of Service in Multiservice IP Networks, pages 19–34, 2003. 16, 35, 37

[Fidler 2006a] M. Fidler. *An end-to-end probabilistic network calculus with moment generating functions*. In IWQoS 2006. 14th IEEE International Workshop on Quality of Service, pages 261–270. IEEE, 2006. 40

[Fidler 2006b] M. Fidler. *A Network Calculus Approach to Probabilistic Quality of Service Analysis of Fading Channels*. In Proceedings of GLOBECOM'2006, 2006. 77, 124

[Fidler 2006c] M. Fidler & S. Recker. *Conjugate Network Calculus: A Dual Approach Applying the Legendre Transform*. Computer Networks, vol. 50, no. 8, pages 1026–1039, 2006. 153

[Fidler 2010] M. Fidler. *A Survey of Deterministic and Stochastic Service Curve Models in the Network Calculus*. IEEE Communications Surveys & Tutorials, vol. 12, no. 1, 2010. 15, 40

[Firoiu 2002] V. Firoiu, J.-Y. Le Boudec, D. Towsley & Zhi-Li Zhang. *Theories and models for Internet quality of service*. Proceedings of the IEEE, vol. 90, no. 9, pages 1565–1591, 2002. 16

[FlexRay 2005] FlexRay. *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. Rapport technique, FlexRay consortium, December 2005. 40, 169

[Frances 2009] F. Frances, C. Fraboul & J. Grieu. *Using network calculus to optimize the AFDX network*. In European Congress on Embedded Real-Time Software (ERTS), Toulouse France, 25/01/06-27/01/06, page (electronic medium). SIA/3AF/SEE, 2009. 40

[Gaubert] S. Gaubert. *The Maxplus toolbox of Scilab*. http://www.cmap.polytechnique.fr/~gaubert/MAXPLUSTOOLBOX.html. 42

[Gaubert 1992] S. Gaubert. *Théorie Linéaire des Systèmes dans les Dioıdes*. PhD thesis, École des mines de Paris, 1992. 22, 41

[Gaubert 1999] S. Gaubert. Introduction aux systèmes dynamiques à événements discrets. Notes de cours, ENSMP, Option Automatique et M2 ATSI, Université d'Orsay, 1999. 22

[Georges 2002] J.-P. Georges, E. Rondeau & T. Divoux. *Evaluation of switched Ethernet in an industrial context by using the Network Calculus*. In Proceedings of 4th IEEE Workshop on Factory Communication Systems, pages 19–26, 2002. 40

[Gollan 2008] N. Gollan, F. A. Zdarsky, I. Martinovic & J. B. Schmitt. *The DISCO Network Calculator*. In Proceedings of MMB'2008, 2008. 78

[Gondran 2002] M. Gondran & M. Minoux. *Graphes, dioıdes et semi-anneaux*. Tec & Doc, 2002. 22

[Grieu 2004] J. Grieu. *Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques*. PhD thesis, INPT, Toulouse, 2004. 16, 40

[Hendriks 2006] M. Hendriks & M. Verhoef. *Timed automata based analysis of embedded system architectures*. In Proc. of IPDPS, 2006. 16

[IntServ] IntServ. *Integrated Services IETF Working Group*. http://datatracker.ietf.org/wg/intserv/charter/. 40

[Jiang 2006] Y. Jiang. *A Basic Stochastic Network Calculus*. In Proceedings of ACM SIGCOMM'2006, 2006. 77

[Jiang 2008] Y. Jiang & Y. Liu. Stochastic network calculus. Springer, 2008. 77, 124

[Junier 2010] A. Junier. *Calcul de bornes de performances déterministes dans les systèmes embarqués. Cas de politique de service à priorités fixes.*, 2010. 145

[Kim 2004] H. Kim & J. C. Hou. *Network Calculus Based Simulation for TCP Congestion Control: Theorems, Implementation and Evaluation*. In Proceedings of INFOCOM'2004, 2004. 149

[Kuczma 2009] M. Kuczma & A. Gilányi. An introduction to the theory of functional equations and inequalities: Cauchy's equation and jensen's inequality. Birkhauser, 2009. 72

[Le Boudec 1998] J.-Y. Le Boudec. *Application of Network Calculus to Guaranteed Service Networks*. IEEE Transactions on Information Theory, vol. 44, no. 3, pages 1087–1096, 1998. 40

[Le Boudec 2001] J.-Y. Le Boudec & P. Thiran. Network calculus: A theory of deterministic queuing systems for the internet, volume LNCS 2050. Springer-Verlag, 2001. 15, 20, 21, 22, 24, 25, 28, 29, 30, 32, 37, 40, 92, 95, 98, 103, 121, 136, 140, 149, 154, 159, 160, 161, 164, 167

[Le Corronc 2010] E. Le Corronc, B. Cottenceau & L. Hardouin. *Flow control with (min,+) algebra*. In 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation, 2010, Heraklion (Crète). ISOLA'10, 2010. 23

[Le Corronc 2011] E. Le Corronc. *Modèles et calculs garantis pour les systèmes (min,+)-linéaires*. PhD thesis, ISTIA, 2011. 23, 26, 41, 160

[Le Corronc 2012a] E. Le Corronc, , B. Cottenceau & L. Hardouin. *Container of (min, +)-linear systems*. Discrete Event Dynamic Systems, 2012. DOI : http://dx.doi.org/10.1007/s10626-012-0148-9. 41, 42

[Lenzini 2007] L. Lenzini, E. Mingozzi & G. Stea. *End-to-end Delay Bounds in FIFO-multiplexing Tandems*. In Proceedings of Valuetools'07, 2007. 34

[Lenzini 2008] L. Lenzini, E. Mingozzi & G. Stea. *A methodology for computing end-to-end delay bounds in FIFO-multiplexing tandems*. Performance Evaluation, vol. 65, no. 11-12, pages 922–943, 2008. 34, 136, 140

[Lucet 2010] Y. Lucet. *What shape is your conjugate? A survey of computational convex analysis and its applications*. SIAM Journal on Optimization, vol. 20, no. 1, page 216, 2010. 153

[Martin 2005] S. Martin, P. Minet & L. George. *End-to-end response time with fixed priority scheduling: trajectory approach versus holistic approach*. Int. J. Communication Systems, vol. 18, no. 1, pages 37–56, 2005. 16, 40

[Mavronicolas 2001] M. Mavronicolas. *Stability in Routing: Networks and Protocols*. Bulletin of the EATCS, vol. 74, pages 119–134, 2001. 41

['Max Plus'] 'Max Plus'. http://www.maxplus.org. The MaxPlus Algebra Home Page. 22

[Nemeth 2005] F. Nemeth, P. Barta, R. Szabo & J. Biro. *Network internal traffic characterization and end-to-end delay bound calculus for generalized processor sharing scheduling discipline*. Computer Networks, vol. 48, no. 6, pages 910–940, 2005. 149

[Pandit 2006a] K. Pandit. *Quality of Service Performance Analysis based on Network Calculus*. PhD thesis, Technische Universität Darmstadt, 2006. 149

[Pandit 2006b] K. Pandit, C. Kirchner, J. B. Schmitt & R. Steinmetz. *A Transform for Network Calculus and Its Application to Multimedia Networking*. In Proceddings of SPIE's Multimedia Computing and Networking Conference 2006 (MMCN'06), 2006. 149

[Perathoner 2007] S. Perathoner, E. Wandeler, L. Thiele, A. Hamann, S. Schliecker, R. Henia, R. Racu, R. Ernst & M. G. Harbour. *Influence of different system abstractions on the performance analysis of distributed real-time systems*. In Proceedings of EMSOFT'2007, 2007. 16

[Pruesse 1994] G. Pruesse & F. Ruskey. *Generating Linear Extensions Fast*. SIAM Journal on Computing, vol. 23, no. 2, pages 373–386, 1994. 128

[Raz 2008] D. Raz, H. Levy & B. Avi-Itzhak. *On the twin measure and queueing systems predictability*. Performance Evaluation, vol. 65, no. 11-12, pages 839–853, 2008. 174

[Rizzo 2005] G. Rizzo & J.-Y. Le Boudec. *"Pay bursts only once" does not hold for non-FIFO guaranteed rate nodes*. Performance Evaluation, vol. 62, no. 1-4, pages 366–381, 2005. 36

[Rizzo 2007] G. Rizzo & J.-Y. Le Boudec. *Generalization of the RIN Result to Heterogeneous Networks of Aggregate Schedulers and Leaky Bucket Constrained Flows*. In Proceedings of ICON'2007, 2007. 17

[Rizzo 2008a] G. Rizzo. *Stability and bounds in aggregate scheduling networks*. PhD thesis, EPFL Lausanne, 2008. 17

[Rizzo 2008b] G. Rizzo & J.-Y. Le Boudec. *Stability and Delay Bounds in Heterogeneous Networks of Aggregate Schedulers*. In Proceedings of INFOCOM'2008, 2008. 17

[Rockfellar 1996] R. T. Rockfellar. Convex analysis. Princeton University Press, 1996. 118, 153

[Scharbarg 2009] J.-L. Scharbarg, F. Ridouard & C. Fraboul. *A probabilistic analysis of end-to-end delays on an AFDX avionic network*. IEEE Transactions on Industrial Informatics, vol. 5, no. 1, pages 38–49, 2009. 40

[Schioler 2005] H. Schioler, J. J. Jessen, J. Dalsgaard & K. G. Larsen. *Network Calculus for Real Time Analysis of Embedded Systems with Cyclic Task Dependencies*. In Proceedings of Computers and Their Applications, 2005. 78, 160

[Schioler 2006] H. Schioler, J. D. Nielsen, K. G. Larsen & J. J. Jessen. *CyNC - a method for Real Time Analysis of Systems with Cyclic Data Flows.* In In Proceedings of 13th RTS Conference on Embedded Systems '05., 2006. 41

[Schioler 2007] H. Schioler, H.-P. Schwefel & M. B. Hansen. *CyNC: a MATLAB/SimuLink toolbox for network calculus*. In Proceedings of Valuetools'2007, 2007. 78

[Schmitt 2005] J. B. Schmitt & U. Roedig. *Sensor Network Calculus: A Framework for Worst Case Analysis*. In Proceedings of 1st International Conference on Distributed Computing in Sensor Systems, 2005. 40

[Schmitt 2006a] J. B. Schmitt & F. A. Zdarsky. *The DISCO network calculator: a toolbox for worst case analysis*. In Proceedings of Valuetools'2006, 2006. 29, 37, 78, 88, 115, 116, 121, 123, 124, 136, 137, 140, 149

[Schmitt 2006b] J. B. Schmitt, F. A. Zdarsky & I. Martinovic. *Performance Bounds in Feed-Forward Networks under Blind Multiplexing*. Technical Report 349/06, University of Kaiserslautern, Germany, 2006. 88, 115, 116, 121, 149

[Schmitt 2007] J. B. Schmitt, F. A. Zdarsky & M. Fidler. *Delay Bounds under Arbitrary Multiplexing*. Rapport technique, University of Kaiserslautern, 2007. 121, 124, 140, 141

[Schmitt 2008b] J. B. Schmitt, F. Zdarsky & M. Fidler. *Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves you on the Lurch...* In INFOCOM, 2008. 123, 124, 125, 136, 137, 140, 141, 170

[Schmitt 2011] J. B. Schmitt, N. Gollan, S. Bondorf & I. Martinovic. *Pay bursts only once holds for (some) non-fifo systems*. In INFOCOM, 2011 Proceedings IEEE, pages 1970–1978. IEEE, 2011. 29, 30

[Schrijver 1998] A. Schrijver. Theory of linear and integer programming. Wiley, 1998. 119

[Skeie 2006] T. Skeie, S. Johannessen & O. Holmeide. *Timeliness of real-time ip communication in switched industrial ethernet networks*. IEEE Tansactions on Industrial Informatics, vol. 2, pages 25–39, 2006. 16

[Sofack 2011] W.M. Sofack & M. Boyer. *Non preemptive static priority with network calculus*. In Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on, pages 1–8. IEEE, 2011. 36

[Tassiulas 1996] L. Tassiulas & L. Georgiadis. *Any Work-Conserving Policy Stabilizes the Ring with Spatial Re-Use*. ACM Transactions on Networking, vol. 4, no. 2, pages 205–208, 1996. 17

[Thiele 2000] L. Thiele, S. Chakraborty & M. Naedele. *Real-time calculus for scheduling hard real-time systems*. In Proceedings of ISCAS'2000, 2000. 86, 145

[Thiele 2001] L. Thiele, S. Chakraborty, M. Gries, A. Maxiaguine & J. Greutert. *Embedded Software in Network Processors Models and Algorithms*. In Proceedings of EMSOFT'2001, 2001. 78, 86, 124

[Thiele 2006] L. Thiele & E. Wandeler. *Real-time calculus Research Project*. Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland, 2006. 42

[Tindell 1994] K. Tindell & J. Clark. *Holistic schedulability analysis for distributed hard real-time systems*. Microprocess. Microprogram, vol. 40, no. 2-3, pages 117–134, 1994. 16

[Touchette 2005] H. Touchette. *Legendre-Fenchel transforms in a nutshell*. Rapport technique, School of Mathematical Sciences, University of London, 2005. 153

[Vanderbei 2000] R. J. Vanderbei. Linear programming, foundations and extensions. Kluwer Academic Publisher, 2000. 139

[Wandeler 2006a] E. Wandeler. *Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems*. PhD thesis, ETH Zurich, 2006. 30, 78, 80, 85, 86, 92, 95

[Wandeler 2006b] E. Wandeler & L. Thiele. *Real-Time Calculus (RTC) Toolbox*. http://www.mpa.ethz.ch/Rtctoolbox, 2006. 42

[Wandeler 2006c] E. Wandeler, L. Thiele, M. Verhoef & P. Lieverse. *System Architecture Evaluation Using Modular Performance Analysis - A Case Study*. Software Tools for Technology Transfer, vol. 8, no. 6, pages 649–667, 2006. 78, 79, 80

INDEX

ALGORITHMIQUE DU NETWORK CALCULUS

English — Abstract Network Calculus is a theory aiming at computing worst-case bounds on performances in communication networks. The network is usually modelled by a digraph : the servers are located on the nodes and the flows must follow path in the digraph. There are constraints on the trafic curves (how much data have been through a given point since the activation of the network) and on the service curves (how much work each server may provide). To derive bounds on the worst-case performances, as the backlog or the end-to-end delay, these envelopes are combined thanks to tropical algebra operators: min, +, convolution... This thesis focuses on Network Calculus algorithmics, that is how effective is this formalism. This work led us to compare various models in the litterature, and to show expressiveness equivalence between Real-Time Calculus and Network Calculus. Then, we suggested a new $(min, +)$ operator to compute performances bounds in networks with agregated flows and we studied feed-forward networks under blind multiplexing. We showed the difficulty to compute these bounds, but we gave an heuristic, which is polynomial for interesting cases.

Français — Résumé Le Network Calculus est une théorie visant à calculer des bornes pire-cas sur les performances des réseaux de communication. Le réseau est modélisé par un graphe orienté où les nœuds représentent des serveurs, et les flux traversant le réseau doivent suivre les arcs. S'ajoutent à cela des contraintes sur les courbes de trafic (la quantité de données passées par un point depuis la mise en route du réseau) et sur les courbes de service (la quantité de travail fourni par chaque serveur). Pour borner les performances pire-cas, comme la charge en différents points ou les délais de bout en bout, ces enveloppes sont combinées à l'aide d'opérateurs issus notamment des algèbres tropicales : min, +, convolution-$(min, +)$... Cette thèse est centrée sur l'algorithmique du Network Calculus, à savoir comment rendre effectif ce formalisme. Ce travail nous a amené d'abord à comparer les variations présentes dans la littérature sur les modèles utilisés, révélant des équivalences d'expressivité comme entre le Real-Time Calculus et le Network Calculus. Dans un deuxième temps, nous avons proposé un nouvel opérateur $(min, +)$ pour traiter le calcul de performances en présence d'agrégation de flux, et nous avons étudié le cas des réseaux sans dépendances cycliques sur les flux et avec politique de service quelconque. Nous avons montré la difficulté algorithmique d'obtenir précisément les pires cas, mais nous avons aussi fourni une nouvelle heuristique pour les calculer. Elle s'avère de complexité polynomiale dans des cas intéressants.