Calcul de bornes de performances déterministes dans les systèmes embarqués. Cas de politique de service à priorités fixes

Aurore Junier

Université de Rennes 1, E.N.S Cachan, antenne de Ker Lann
Faculté d'informatique
Rapport de stage

Maître de stage : Anne Bouillard
Équipe : Distribcom, INRIA

Calcul de bornes de performances déterministes dans les systèmes embarqués. Cas de politique de service à priorités fixes.

Aurore Junier

Rennes, le 4 juin 2010

Table des matières

1 Introduction à la théorie du *Network Calculus*

Aujourd'hui les systèmes embarqués représentent une grande part du marché mondial des technologies. Nous les retrouvons dans les téléphones, les GPS et les avions...etc. L' AFDX [04] (*Avionic Full Duplex Switched Internet*) est le support de communication embarqué dans les avions. Il utilise des lignes de communication *full duplex* (communication dans les deux sens) reliées par des *switchs* (commutateurs réseaux). Un enjeu majeur réside dans le fait de savoir borner le délai d'attente dans les buffers des *switchs*. Il est également nécessaire de savoir calculer une borne maximale du délai subi par une donnée d'un bout à l'autre du réseau. En effet, dans le cas du système AFDX, on sait qu'il fonctionne correctement si les données traversent le réseau avant un certain laps de temps. Avant de placer ce système dans un avion il est nécessaire de s'assurer que cette condition est toujours vérifiée.

Le *Network Calculus* [05] et [08], est une théorie qui permet de calculer des bornes de performance dans les réseaux de communication, composés de flux de données traversant certains (ou tous) éléments du réseau. Un exemple est représenté sur la figure 1. Il comporte 2 serveurs et trois flux. Le premier est traversé par tous les flux mais le second uniquement par 2 d'entre eux. Les bornes

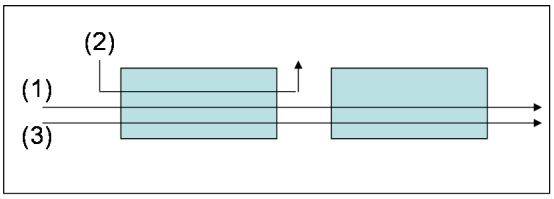

FIG. 1 – Exemple de réseau.

calculées grâce au *Network Calculus* sont des bornes supérieures subies par les données d'un ou plusieurs flux telles que le *délai* ou la *charge* maximale d'un système ou du réseau entier, ainsi que d'autres valeurs permettant de déterminer la qualité de service. Les calculs sont effectués par le biais de fonctions. Par exemple la *courbe des arrivées* est une fonction qui majore la quantité de données arrivant à travers un flux, la *courbe de service* délivre une borne minimale de la quantité de données traitées à chaque instant par le serveur étudié. Nous utiliserons une structure mathématique sur l'espace $\mathbb{R} \cup \{+\infty\}$ ayant pour lois internes le minimum et l'addition, deux opérations couramment utilisées sont la convolution et la déconvolution. Un peu plus loin, nous donnerons une définition formelle de ces opérations.

Exemple introductif Considérons le réseau de la figure 2, composé d'un système et d'un flux. Nous présentons ici, les systèmes *leacky bucket*, que l'on

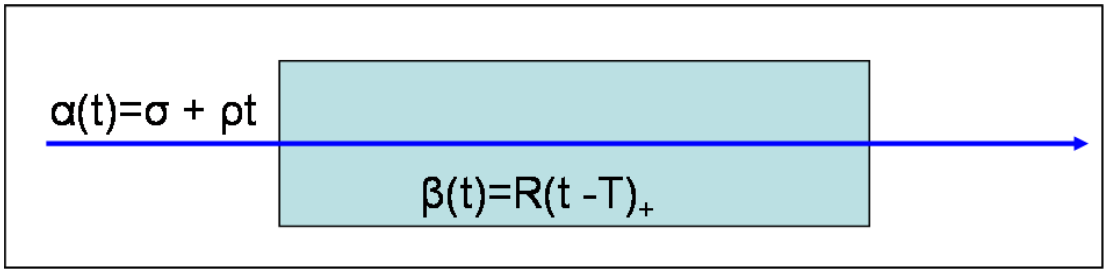

FIG. 2 – Exemple de système.

peut traduire par seau percé, qui sont très souvent utilisés pour la modélisation de systèmes contraints. Le livre [05] présente une étude complète de ces systèmes.

Soit un flux traversant un tel système. Les données y sont acheminées afin d'y être traitées. Nous allons introduire une notion : *la courbe d'arrivée* qui sera présenté formellement plus loin.

Les données à l'entrée d'un système arrivent avec un certain débit. Celui-ci peut varier au cours du temps. Pour se donner une image concrète on peut faire l'analogie avec la transmission de bits dans un câble allant d'un point A à un point B, lors d'une communication téléphonique.

La fonction de contrainte α des systèmes *leacky bucket* est définie par : $\forall t \in \mathbb{R},\ \alpha(t) = \sigma + \rho(t)$. Le paramètre σ définit la taille du buffer. Du point de vue des arrivées, cela signifie que la quantité maximale de données arrivant d'un coup dans le système est σ. Le paramètre ρ, quant à lui définit le taux d'arrivée des données : à chaque unité de temps, au maximum ρ données sont acheminées jusqu'au système.

Sur le schéma ci-dessous, nous avons représenté par la courbe bleue (R) les arrivées cumulées des données au système. Visuellement (R) est contrainte par α si sa courbe est toujours en dessous de cette dernière. On dira alors que le système est (σ, ρ)-contraint.

Nous avons cherché, durant ce stage, à fournir une méthode de calcul du délai subi par un flux dans un réseau de manière déterministe, lorsque la politique de service est à priorité fixe. Il existe bien sûr des méthodes, [10] et [02], fournissant des délais dans ce genre de situations. Notre but sera de fournir des résultats plus fins. Nous allons tout d'abord introduire la base théorique sur laquelle repose les calculs de délai, appelée *Network Calculus*, ainsi que les différentes méthodes qui en découlent et qui permettent ces calculs. Nous présentons ensuite un premier résultat sur le délai borné. Puis, nous énoncerons le travail que nous avons réalisé, avant de le comparer aux méthodes concurrentes. Enfin, nous exposerons une limite actuelle de notre travail, avant de conclure.

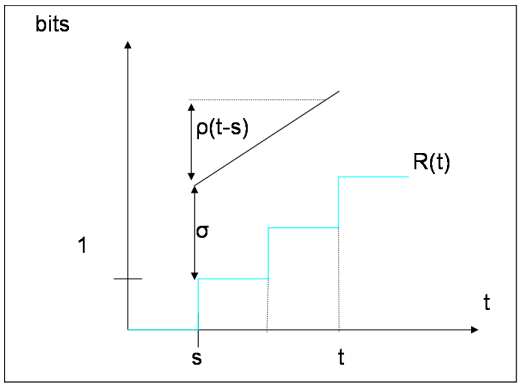

FIG. 3 – Courbe d'arrivée d'un système leacky bucket.

2 Le *Network Calculus*

2.1 Outils mathématiques : convolution et déconvolution

Nous commençons par introduire l'ensemble de fonctions suivant : $\mathcal{F} = \{f : \mathbb{R}_+ \to \mathbb{R}_+ \cup \{+\infty\} \mid f \text{ est croissante et } f(0) = 0\}$. Dans la suite, nous supposerons toujours que les fonctions appartiennent à cet ensemble.

Nous introduisons dans cette partie les opérations de convolution et de déconvolution dans un espace à temps discret dans le semi-anneau ($\mathcal{F}$, $\wedge$,$\otimes$), où $\wedge$ est la fonction minimum et $\otimes$ l'opérateur de convolution.

Définition 1 *Soit f et $g \in \mathcal{F}$ alors la convolution de f et g est :*

$$\forall t \in \mathbb{R}_+ \ (f \otimes g)(t) = \inf_{0 \leqslant s \leqslant t} \{f(t-s) + g(s)\}.$$

Ci-dessous sont listées les propriétés de la convolution dans le semi-anneau ($\mathcal{F}$, $\wedge$,$\otimes$).

Proposition 1 (Propriétés de la convolution) *Soient $f, g, h \in \mathcal{F}$.*

1. *Clôture : $f \otimes g \in \mathcal{F}$.*
2. *Associativité : $(f \otimes g) \otimes h = f \otimes (g \otimes h)$.*
3. *Élement neutre : soit δ_0 le Dirac en 0 : $f \otimes \delta_0 = f$.*
4. *Élement absorbant : l'élément absorbant est $\epsilon : t \mapsto +\infty$. On a $f \otimes \epsilon = \epsilon$.*
5. *Distributivité de la convolution par rapport au $\wedge$: $(f \wedge g) \otimes h = (f \otimes h) \wedge (g \otimes h)$.*
6. *Addition par une constante : $(f + K) \otimes g = f \otimes g + K$.*

Nous définissons l'opération de déconvolution ainsi que les principales propriétés qui en découlent :

Définition 2 *Soit f et $g \in \mathcal{F}$ alors la déconvolution de f et g est :*

$$\forall t \in \mathbb{R}_+ \ (f \oslash g)(t) = sup_{s \geqslant 0}\{f(t+s) - g(s)\}.$$

Proposition 2 (Propriétés de la déconvolution) *Soient $f, g, h \in \mathcal{F}$.*

1. $(f \oslash g) \oslash h = f \oslash (g \otimes h)$.
2. *Composition de $\oslash$ et de $\otimes$:* $(f \otimes g) \oslash h \leq f \otimes (g \oslash h)$.
3. *Dualité entre $\oslash$ et $\otimes$:* $f \oslash g \leq h$ *si et seulement si* $f \leq g \otimes h$.

Remarque : L'opération $\oslash$ est une pseudo-inverse pour $\otimes$. En effet, si l'on cherche à calculer $f \leq g \otimes h$, on sait que $f \oslash g$ est la plus petite fonction h vérifiant cette inégalité.

Dans la suite, nous verrons que ces deux opérations sont très utilisées dans le calcul de bornes caractéristiques de réseaux.

2.2 Courbes de contrôle des flux et des serveurs

Soit A la fonction des arrivées cumulées et B la fonction des sorties cumulées. Pour tout t, $A(t)$ est le nombre de données arrivées dans le système entre les instants 0 et t, et $B(t)$ est la quantité de données ayant quitté le réseau entre les instants 0 et t. De manière intuitive $B(t) \leqslant A(t)$, car on suppose que le serveur ni ne perd, ni n'ajoute de paquet.

Nous commençons par définir les courbes d'arrivée, fournissant des garanties sur les arrivées du système, puis nous introduisons les courbes de services qui définissent des garanties sur la vitesse de traitement des données du système.

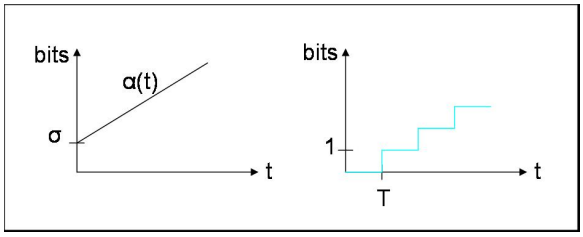

FIG. 4 – Courbes d'arrivées affine (à gauche) et étagée (à droite).

2.2.1 Courbe d'arrivée

On définit la courbe d'arrivée comme suit :

Définition 3 (Courbe d'arrivée) *Soit $\alpha \in \mathcal{F}$ une fonction croissante définie pour $t \in \mathbb{R}_+$.On dit que A est α-contrainte si et seulement si : $\forall s \leqslant t$,*

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

Ainsi, quels que soient deux instants s et t choisis, la quantité de données arrivée entre ces deux moments est inférieure à $\alpha(t-s)$. La fonction α va ainsi donner une borne sur la quantité de données pouvant arriver entre deux instants.

Parmi toutes les courbes d'arrivée, on distingue ici deux classes : les courbes d'arrivée affines (ou *Leacky Bucket*) ($\alpha(t) = \sigma + \rho t$) et les courbes d'arrivées à étages ($k\nu_{T,\tau}(t) = k\lfloor\frac{t+\tau}{T}\rfloor$). Ces courbes sont représentées sur la figure 4.

2.2.2 Courbe de service

Comme nous l'avons annoncé au début de cette section, les courbes de service donnent des garanties aux flots. En effet elles vont quantifier la vitesse à laquelle un nœud peut traiter des données. Nous noterons A (resp. B) le flot d'entrée (resp. de sortie) d'un système S. De manière formelle une courbe de service est définie comme suit :

Définition 4 (Courbe de service minimale) *Le système S offre au flot une courbe de service $\beta \in \mathcal{F}$ si et seulement si $B \geqslant A \otimes \beta$.*

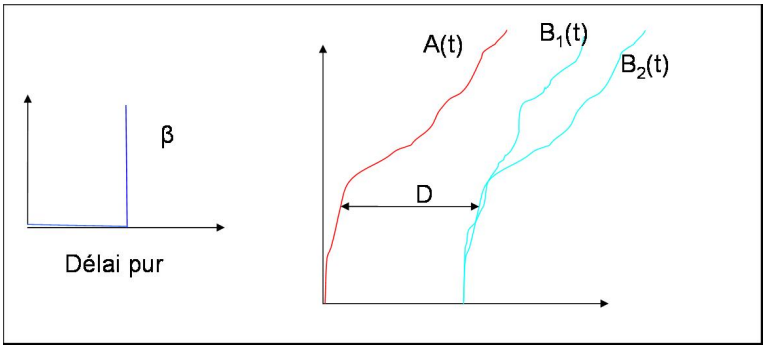

FIG. 5 – Courbes de service.

Dans l'article [02] sont répertoriés les types de courbes de service dont deux classes sont :

- courbe de service simple β : $A \geqslant B \geqslant A \otimes \beta$;
- courbe de service strict β : $A \geqslant B$ et pour toute période chargée, intervalle de temps pendant lequel le réseau n'est pas vide (définie formellement dans le paragraphe 1.3) [s,t],

$$B(t) - B(s) \geqslant \beta(t-s).$$

On remarque facilement que si S offre une courbe de service strict β, il offre aussi une courbe de service simple β.

La figure 5 illustre le principe des courbes de service simple. La figure de droite donne la courbe de service β d'un système S. Elle correspond à un simple

délai D pur. La figure à droite représente les arrivées cumulées de S (en rouge) ainsi que trois flux de sorties cumulées B_1 et B_2 représentent deux sorties possibles de S lorsque le système est muni d'une courbe de service simple. Les trajectoires de sortie sont obtenues à partir de A, en effet, il faut : $B(t) \geqslant A(t+D)$.

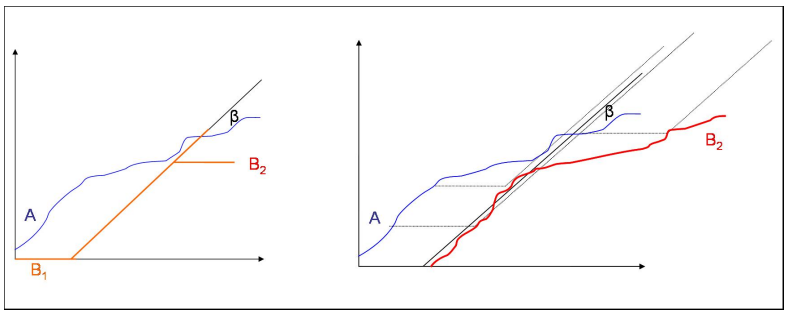

FIG. 6 – Second exemple de courbes de service et des flots d'entrée et de sortie du serveur.

Sur la figure 6, nous avons modélisé un comportement possible des flots d'entrée et de sortie, pour un serveur muni d'une courbe de service strict ou simple $\beta(t) = R(t-T)_+$. Sur le repère de droite, est représenté le flot de sortie minimal B_1 d'un serveur muni d'une courbe de service strict, et à gauche B_2 est une trajectoire issu d'une courbe de service simple. En général, les courbes de service des réseaux utilise cette fonction. Dans notre étude nous nous placerons toujours dans ce cas. Sur la figure 6 nous avons symbolisé un flux A, des entrées dans le serveur, ainsi que le flux de sortie minimal B. En effet B est nul pour tout t inférieur à T. Puis des données sortent exactement avec un taux R tant que le serveur n'est pas vide. Au moment où le serveur est vide, une nouvelle période de latence débute.

Définition 5 (Courbe de service maximale) *Soit S un système et A (resp. B) le flot d'entrée (resp. de sortie). Alors S offre au flot une courbe de service maximale $\beta' \in \mathcal{F}$ définie par : $B \leqslant A \otimes \beta$.*

2.3 Bornes caractéristiques

En introduction nous avons cité les bornes classiquement évaluée par le *Network Calculus* : *charge*, *délai*, et *flot de sortie/entrée*. Dans cette partie nous donnons des définitions précises permettant le calcul de ces grandeurs.

Définition 6 (Charge) *La charge de S à la date t est définie par $A(t) - B(t)$.*

La charge correspond en réalité à la quantité de données présentes dans le serveur étudié à chaque instant. La charge d'un réseau est une notion qui permet de définir un intervalle de temps spécial appelé *période chargée*, et qui constitue une notion importante pour la suite de notre travail.

Définition 7 (Période chargée) *La période chargée est un intervalle de temps I de $\mathbb{R}_+$ durant laquelle la charge du serveur S n'est pas nulle. Autrement dit :* $\forall u \in I, A(u) - B(u) > 0$.

À partir d'une date quelconque t, on peut retrouver son début de période chargée. Cette date est notée $start(t)$, et est définie par : $start(t) = \sup\{u \leqslant t | A(u) = B(u)\}$.

Le livre [08] nous fournit le résultat ci-dessous.

Théorème 3 (Propriété de la charge) *Soit α la contrainte sur A et soit β la courbe de service du système. Alors la charge vérifie :*

$$A(t) - B(t) \leqslant sup_{s \geqslant 0}\{\alpha(s) - \beta(s)\} = \alpha \oslash \beta(0).$$

La définition suivante concerne le délai subi par une donnée de l'entrée à la sortie du réseau. Cette information permet de définir si un réseau est surchargé ou non, et aussi, si une donnée est traitée assez vite (point important dans l'AFDX présenté en introduction).

Définition 8 (Délai) *Soit un flot traversant S, le délai subi par la donnée entrant dans le système à l'instant t est $d(t)$=*$\inf\{\tau \geqslant 0 | A(t) \leqslant B(t+\tau)\}$.

Théorème 4 (Propriété du délai) *Soit un flot traversant S α-contraint et β la courbe de service associée à S. Alors $d(t) \leqslant hDev(\alpha,\beta)$ où*

$$hDev = \sup_{s \geqslant 0}\{\inf_{\tau \geqslant 0}\{\alpha(s) \leqslant \beta(s+\tau)\}\}.$$

Connaissant α et β, on peut obtenir le délai et la charge maximale. En effet le délai maximal correspond à la plus grande déviation horizontale séparant α de β, et la charge maximale correspond à la plus grande déviation verticale (notée $vDev = \sup_{s \geqslant 0}\{\alpha(s) - \beta(s)\}$). Nous avons représenté ces bornes sur le schéma 7.

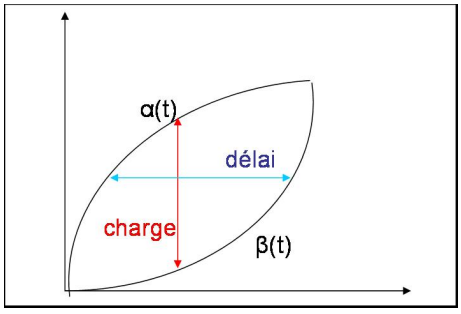

FIG. 7 – Représentation des délai et charge maximum d'un réseau à l'aide des contraintes d'arrivée et de service.

Exemple Nous allons calculer le délai maximal d_{max} subi par un flux contraint par la fonction $\alpha(t) = \sigma + \rho t$, passant dans un serveur muni d'une courbe de service β, telle que $\beta(t) = R(t-T)_+$. D'après la propriété ci-dessus,

$d_{max} \leqslant hDev(\alpha, \beta) = \sup_{t \geqslant 0}\{\inf_{d \geqslant 0}\{\alpha(t) \leqslant \beta(t+d)\}\}$. Commençons par calculer $\alpha(t) \leqslant \beta(t+d)$:

$$\begin{aligned}
\alpha(t) &\leqslant \beta(t+d) \\
\sigma + \rho t &\leqslant R(t+d-T)_+ \\
\sigma + \rho t &\leqslant R(t+d-T) \\
\frac{\sigma + \rho t - R(t-T)}{R} &\leqslant d \\
\frac{\sigma + RT}{R} + \frac{(\rho - R)t}{R} &\leqslant d.
\end{aligned}$$

Donc,

$$\begin{aligned}
d_{max} &\leqslant hDev(\alpha, \beta) \\
d_{max} &\leqslant \sup_{t \geqslant 0}\{\inf_{d \geqslant 0}\{\alpha(t) \leqslant \beta(t+d)\}\} \\
\text{Si } (\rho - R) < 0 \text{, alors } d_{max} &= \frac{\sigma + RT}{R}; \\
\text{Si non } d_{max} &= +\infty.
\end{aligned}$$

Théorème 5 (Flot de sortie) *Soit un flot passant dans S offrant une courbe d'arrivée α et une courbe de service β. Alors le flot sortant est α'-contraint avec :*

$$\alpha' = (\alpha \oslash \beta)_+.$$

Exemple Calculons le flot de sortie d'un flux de courbe d'arrivée α telle que $\alpha(t) = \sigma + \rho t$, traversant un serveur de courbe de service β, telle que $\beta(t) = R(t-T)_+$. Nous supposons ici que $R > \rho$. Cette hypothèse correspond au cas généralement considéré car le délai est borné, comme nous l'avons montré précédemment.

$$\begin{aligned}
\alpha'(t) &= (\alpha \oslash \beta)_+ \\
&= ((\sigma + \rho t) \oslash R(t-T)_+)_+ \\
&= \sup_{u \geqslant 0}\{\sigma + \rho(t+u) - R(u-T)_+\} \\
&= \sup_{0 \leqslant u \leqslant T}\{\sigma + \rho(t+u)\} \vee \sup_{u \geqslant T}\{\sigma + \rho(t+u) - R(u-T)\} \\
&= \sup_{0 \leqslant u \leqslant T}\{\sigma + \rho(t+u)\} \vee \sup_{u \geqslant T}\{\sigma + \rho t - (R-\rho)u + RT\} \\
&= \sigma + \rho(t+T) = \alpha(t+T).
\end{aligned}$$

Grâce à ce résultat, si l'on s'intéresse à la charge d'un serveur de courbe de service β traversé par un flux α-contraint est $\alpha(T)$.

Dans les calculs futurs, nous appliquerons directement ce résultat.

2.4 Premiers résultats sur les caractéristiques des réseaux

Deux questions sont très importantes lorsque l'on considère les systèmes comportant plusieurs flots et plusieurs composants à la suite : comment obtenir la capacité non utilisée à chaque instant ? Comment obtenir la courbe d'arrivée associée à un composant quelconque du réseau, ne connaissant que les courbes d'arrivées des flux à l'entrée du réseau ?

Définition 9 (Flux résiduel) *Considérons deux flux f_1 et f_2 tel que f_1 est α_1-contraint, et supposons que β soit la courbe de service strict associé à un système S. Alors une courbe de service associée à f_2 est $(\beta - \alpha_1)_+$.*

On peut ainsi calculer la capacité résiduelle qui est donc apportée aux flux les moins prioritaires. Grâce à ce résultat on peut définir la capacité de traitement qu'un serveur accorde à un flux quelconque. Sur la figure 8, si f_2 est moins prioritaire que f_1 on peut calculer la courbe de service résiduel associée et en déduire le délai. C'est ce que nous faisons dans l'exemple ci-dessous.

Exemple Nous examinons maintenant un serveur de courbe de service β_1, telle que $\beta_1(t) = R(t-T)_+$ traversé par deux flux f_1 et f_2 de courbe d'arrivée respective α_1, α_2 avec : $\alpha_1(t) = \sigma_1 + \rho_1 t$, et $\alpha_2(t) = \sigma_2 + \rho_2 t$. Ce réseau est représenté sur la figure 8 ci-dessus. Calculons β' le flux résiduel associé au second flux.

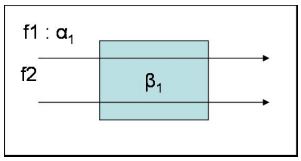

FIG. 8 – Représentation du réseau étudié.

$$\begin{aligned} \beta'(t) &= (\beta_1 - \alpha_1)_+(t) \\ &= ((R(t-T)_+ - \sigma_1 + \rho_1 t)_+ \\ &= (R-\rho_1)(t-T')_+, \end{aligned}$$

avec T' est la date à laquelle $(\beta - \alpha_1)$ s'annule :

$$\begin{aligned} \beta(T') &= \alpha_1(T') \\ R(T' - T) &= \sigma_1 + \rho_1 T' \\ T' &= \frac{\sigma_1 + RT}{R - \rho_1}. \end{aligned}$$

Définition 10 (Concaténation) *Soit un flot traversant un système s_1 et un système s_2. Supposons que le système s_i est une courbe de service β_i pour i=1,2. Alors la concaténation des deux systèmes offre une courbe de service $\beta_1 \otimes \beta_2$.*

Exemple Nous étudions un réseau composé de deux serveurs, s_1 et s_2 traversés par un flux, dont les courbes de services associés sont respectivement β_1 et β_2, telles que $\beta_1(t) = R_1(t-T_1)_+$ et $\beta_2(t) = R_2(t-T_2)_+$. Nous souhaitons obtenir la courbe de service β_{Tot} du réseau. Un résultat du le livre [08] nous dit que la convolution de fonctions convexes, affine par morceaux, résulte en la concaténation par pentes croissantes des segments affines. Donc :

$$\beta_{Tot}(t) = (\min(R_1, R_2)(t - T_1 - T_2))_+.$$

Ainsi, lorsqu'un flux traverse des serveurs les uns à la suite des autres, on peut imaginer qu'il franchit un unique serveur dont la courbe de service résulte en la convolution des courbes des serveurs traversés. Cependant ce principe n'est valable que si tous les flux traversent l'ensemble des serveurs dont on fait la concaténation. En effet, si ce n'est pas le cas, le problème devient plus complexe. Ce cas est présenté ultérieurement. Cette définition de la concaténation permet de calculer la courbe de service d'un flot passant à travers un ensemble de systèmes.

3 Méthodes de calcul du délai maximal d'un réseau

Dans cette partie du rapport nous faisons l'inventaire des méthodes existantes permettant de calculer le délai maximal d'un flux passant à travers un réseau. Nous commençons par présenter des travaux dont la politique de service ne considèrent pas de priorités sur les flux, [03] et [09]. C'est ce que l'on appelle des *politiques "arbitraires"* : tout les flux sont servis ensemble par le serveur dont on connait une borne minimale de la capacité de traitement. Enfin nous présentons une méthode de calcul prenant en compte des priorités, [10]. Après avoir présenté notre travail nous comparerons les délais obtenus par ces méthodes avec la nôtre.

3.1 Réseaux à flux multiplexés et regroupés

Nous nous intéressons dans cette partie aux réseaux dont les flux sont regroupés et où l'on ne peut pas supposer que la politique de service est FIFO

(politique aveugle). L'analyse [09] commence par calculer la quantité de données traitées en chaque nœud pour chaque flot considéré séparément. Autrement dit, elle calcule la bande passante associée à chaque nœud. Pour déduire le délai maximum, il reste à prendre la déviation horizontale (comme le montre la figure 7). Cette méthode s'appelle *Separate Flow Analysis* (SFA).

Prenons un exemple du calcul de délai grâce à cette méthode. La figure 9 présente un réseau composé de deux serveurs s_1 et s_2 de courbes de service respectives β_1, β_2. Deux flux, f_1 et f_2 de courbes d'arrivée respectives α_1, α_2 traversent l'ensemble du réseau. Alors le délai du flux f_1, noté d, est :

$$d = hDev(\alpha_1, [\beta_1 - \alpha_2]_+ \otimes [\beta_2 - (\alpha_2 \oslash (\beta_1 - \alpha_1)_+)_+]_+).$$

Cette méthode s'applique aux politiques arbitraires. En effet comme on ne sait pas quels sont les flux servis en premiers, on calcule les courbes de service résiduel en retirant tous les flux présents autres que f_ℓ, le flux dont on calcul le délai. Cependant cette approche concerne également les politiques de services à priorités fixes. Dans ce cas, il suffit de ne retrancher le service apporté aux flux plus prioritaires uniquement. Sur l'exemple précédent, si l'on suppose f_1 plus prioritaire, alors la formule devient : $d = hDev(\alpha_1, \beta_1 \otimes \beta_2)$, et le délai de f_2 est :

$$d = hDev(\alpha_2, [\beta_1 - \alpha_1]_+ \otimes [\beta_2 - (\alpha_1 \oslash \beta_1)_+]_+).$$

Dans le calcul fait par SFA le multiplexage des flots qui interfère est compté plusieurs fois. Ce n'est pas une vision très réaliste, car à partir du moment où une donnée est passée à travers un nœud avant une autre, le problème qui consiste à savoir qu'elle donnée passe en premier ne va plus se poser pour les serveurs suivant. Pour cela on compte les coûts de multiplexage une seule fois. Cette méthode se nomme *Pay Multiplexing Only Once SFA* (PMOO-SFA). Appliquons l'analyse PMOO-SFA à l'exemple de réseau de la figure 9, pour le calcul du délai du flux f_1. Il est calculé par $(\beta_1 \otimes \beta_2 - \alpha_2)_+$. Autrement dit, on effectue d'abord la concaténation des systèmes, puis on retranche la quantité de traitement nécessaire au premier flux. Cela donne donc la courbe de service associée au second flux, et grâce à la déviation horizontale maximale entre cette courbe et la courbe d'arrivée α_2 on peut calculer le délai maximal. La méthode PMOO-SFA, lorsqu'elle s'applique, fournit de meilleur résultat. Cependant, son utilisation est réduite. En effet, si l'on calcule le délai de f, il faudra que les autres flux effectuent le même chemin dans le réseau que celui emprunté par f.

3.2 Algorithme de programmation linéaire (LP)

L'article [03] apporte une nouvelle méthode de calcul des bornes de délai et de charge de ces réseaux, en utilisant la programmation linéaire, grâce au théorème suivant :

Théorème 6 *Soit S un système contenant n nœuds et f flots. Si S est acyclique alors pour un flot i(resp. un serveur j) il existe une collection finie λ de programmes linéaires avec pour valeur optimale $opt_\lambda \in \lambda$ tel que* $\max(opt_\lambda)$ *soit le pire délai d'un bout à l'autre du réseau (resp. la pire charge).*

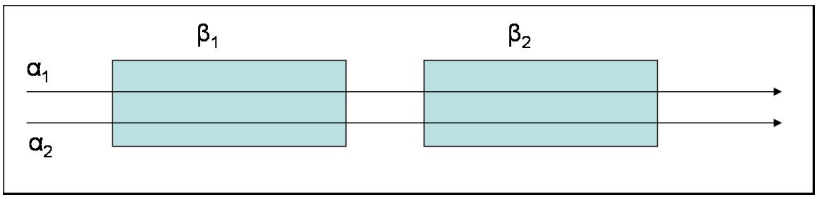

FIG. 9 – Représentation d'un réseau contenant deux flux et deux systèmes.

Grâce à cette technique on peut désormais calculer le pire délai et la pire charge d'un réseau ne contenant pas de cycle, sans connaissance préalable de la politique de service. Le problème c'est que le temps de calcul est NP-difficile. Cependant, si le réseau est en tandem ce calcul se fait en temps polynômial.

Exemple À titre d'exemple, nous énumérons les contraintes linéaires du réseau de la figure 9, composé de deux serveurs s_1 et s_2 et deux flux f_1 et f_2. On notera, pour tout $i \in \{1,2\}$, A_i les arrivées cumulées des données de f_i en s_1, B_i les arrivées cumulées des données de f_i en s_2, et C_i le flot de données de f_i sortant du réseau. On dispose de quatre dates importantes : t_0 début de période chargée de s_1, t_1 celle de s_2, t_2 (resp. u) date à laquelle la donnée de f_2 étudiée sort du réseau (resp. rentre dans le réseau).

- Objectif : max $t_2 - u$;
- Croissance des dates : $t_0 \leqslant t_1 \leqslant t_2$;
- Début de période chargée :
  - $A_1(t_0) = B_1(t_0)$;
  - $A_2(t_0) = B_2(t_0)$;
  - $B_1(t_1) = C_1(t_1)$;
  - $B_2(t_1) = C_2(t_1)$;
- Croissance du flux $f_i, i \in \{1,2\}$:
  - $A_i(t_0) \leqslant A_i(t_1) \leqslant A_i(t_2)$;
  - $B_i(t_0) \leqslant B_i(t_1) \leqslant B_i(t_2)$;
  - $C_i(t_0) \leqslant C_i(t_1) \leqslant C_i(t_2)$;
- Causalité :
  - $A_i(t_0) \geqslant B_i(t_0) \geqslant C_i(t_0)$;
  - $A_i(t_1) \geqslant B_i(t_1) \geqslant C_i(t_1)$;
  - $A_i(t_2) \geqslant B_i(t_2) \geqslant C_i(t_2)$;
- Contrainte α :
  - $A_i(t_1) - A_i(t_0) \leqslant \alpha_i(t_1 - t_0)$;
  - $A_i(t_2) - A_i(t_1) \leqslant \alpha_i(t_2 - t_1)$;
  - $A_i(t_2) - A_i(t_0) \leqslant \alpha_i(t_2 - t_0)$;
- Contrainte β :
  - $B_1(t_1) + B_2(t_1) - B_1(t_0) + B_2(t_0) \geqslant \beta_1(t_1 - t_0)$;
  - $C_1(t_2) + C_2(t_2) - C_1(t_1) + C_2(t_1) \geqslant \beta_2(t_2 - t_1)$.

3.3 Priorités fixes

Dans l'étude [10] on considère deux courbe des arrivées : α^{min} correspond au nombre minimum de données arrivées entre les instants 0 et t, et α^{max} au nombre maximum de données. De même on suppose avoir deux courbes services β^{min} et β^{max}. On calcule alors le délai maximal par :

$$d(t) = hDev(\alpha^{max}, \beta^{min}).$$

Cette technique de calcul basée sur plusieurs courbes de contraintes vise à apporter un ordonnancement efficace dans le cas de réseaux à taches périodiques et apériodiques, temps réel ou sans contrainte de temps, à l'aide de priorités fixes. On veut donc garantir l'arrivée des tâches temps-réel avant leurs échéances tout en permettant au système d'exécuter le plus de tâches non-contraintes possibles. Pour cela on assigne les priorités les plus fortes aux tâches temps-réel.

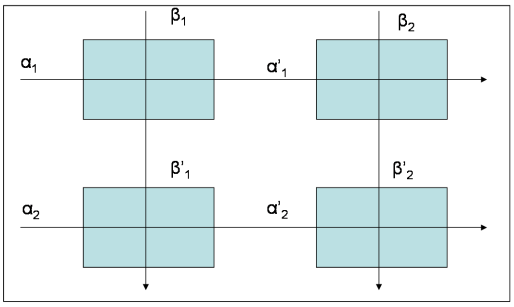

FIG. 10 – Représentation d'un réseau contenant deux flux et deux systèmes.

La figure 10, représente le même réseau que la figure 9, mais vu avec la méthode de l'article [10]. Nous avons sur ce schéma représenté simplement les courbes de service minimal, et d'arrivée maximale. Les flux α' et β' correspondent aux flux résiduels de α et β. Les deux flots sont représentés comme traités par différents systèmes. En réalité c'est le même et l'on donne au deuxième flot la courbe de service résiduel. Le problème de cette approche réside dans le fait que le multiplexage est réalisé au fur et à mesure de l'obtention des courbes résiduelles. Donc ils ne font pas le PMOO (Pay Multiplexing Only Once). Par conséquent, on risque de sur-estimer le délai.

Exemple Afin d'illustrer les propos précédent, nous allons effectuer les calculs des différentes fonctions α et β du réseau de la figure 10. Nous allons pour cela utiliser des résultats du paragraphe 1.4. Commençons par définir la courbe de service résiduel du serveur s_1. On trouve directement $\beta'_1 = (\beta_1 - \alpha_1)_+$. De plus, la courbe d'arrivée du premier flot est $\alpha'_1 = (\alpha_1 \oslash \beta_1)_+$. Intéressons nous

maintenant au second flux. Sa courbe d'arrivée au second serveur est :

$$\begin{aligned} \alpha_2' &= (\alpha_2 \oslash \beta_1')_+ \\ &= (\alpha_2 \oslash (\beta_1 - \alpha_1)_+). \end{aligned}$$

Il ne nous manque plus que la courbe de service résiduel du second serveur, que l'on obtient par : $\beta_2' = (\beta_2 - \alpha_1')_+ = (\beta_2 - (\alpha_1 \oslash \beta_1)_+)_+$. Pour obtenir le délai maximal de f_2, il suffit d'utiliser la fonction hDev(α_2', β_2'). On retrouve ainsi la même formule qu'au paragraphe 2.1, lorsque l'on calcule le délai de f_2 avec SFA.

3.4 Nœuds à capacités variables

L'article [02] présente une méthode alternative pour calculer le flot de sortie B d'un serveur. Il est vu comme un *nœud à capacité variable* et est alors calculé en fonction de son flux d'entrée A et sa capacité. Plus formellement le flot de sortie est défini par :

Définition 11 (Nœud à capacité variable) *Soient* $(A, B) \in \mathcal{F} \times \mathcal{F} \mid \exists C \in \mathcal{F}$, $\forall t \geqslant 0, B(t) = \inf_{0 \leqslant s \leqslant t}[A(s) + C(t) - C(s)]$ *avec* $C(t) - C(s) \geqslant \beta(t - s)$.

Il est montré dans [02] que : $B(t) = A(start(t)) + C(t) - C(start(t))$. Un second résultat de cet article est que l'ensemble des trajectoires admettant β comme courbe de service pour des serveurs définis comme des nœuds à capacité variable et celles admettant β comme courbe de service strict lorsque les serveurs respectent les contraintes du *Network Calculus* sont équivalentes si $\beta \oslash \beta < \infty$ et que β est convexe. Dans la suite nous choisissons des fonctions β de ce type, ainsi il nous sera possible d'utiliser les propriétés des nœuds à capacité variable ou bien celles du *Network Calculus*.

4 Étude des réseaux quelconques

On considère dans cette partie les réseaux quelconques dans lesquels les flux sont soumis à des priorités. Dans la suite, les flux sont indicés selon leur priorité : soient f_u, f_v tels que $u < v$ alors $prio(f_v) < prio(f_u)$. Nous considérons un réseau contenant n serveurs tel que s_k dispose d'une courbe de service minimale β_k telle que, $\beta_k(t) = R_k(t - T_k)_+$ et m flux tels que le flux f_i est contraint par $\alpha_i(t) = \sigma_i + \rho_i t$. On ne pose aucune contrainte sur la façon dont les flux traversent le réseau. Un exemple de réseau concerné est donné sur la figure 11. Le but est de formuler une condition nécessaire et suffisante pour obtenir un délai borné dans ce réseau.

Dans un premier temps, nous allons calculer la contrainte sur le flux de sortie du serveur s_k pour le flux f_i. Comme les flux sont soumis à des priorités dans ce réseau, pour obtenir la contrainte de sortie d'un flux f_i, il nous faut d'abord connaître les contraintes des flux plus prioritaires en ce serveur. Soit M_i^k l'ensemble des serveurs traversés par f_i avant d'arriver en s_k (s_k est inclu dans cet ensemble).

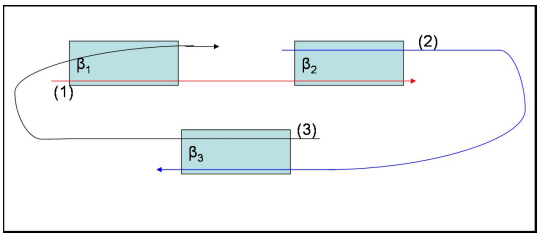

FIG. 11 – Exemple de réseau sur lequel on peut appliquer théorème.

Lemme 1 (Contrainte de sortie d'un flux en un serveur quelconque) *À la sortie du serveur s_k, la contrainte sur f_i est :*
$\alpha_i^{k+1}(t) = \sigma_i + \rho_i(t + \sum_{j \in M_i^k} T_j^i)$, *où* $T_j^i = \frac{\sum_{f_j \in s_k, j<i} \sigma_j + \rho_j(\sum_{\ell \in M_j^k} T_\ell^j) + R_k T_k}{R_k - \sum_{f_j \in s_k, j<i} \rho_j}$.

Démonstration : Pour prouver ce lemme, nous devons raisonner avec une double récurrence, sur le nombre de serveurs et de flux. Nous commençons par supposer qu'il n'y a que f_1 dans le réseau, flux le plus prioritaire.

Initialisation : Le réseau est composé d'un système s_1, de courbe de service β_1, telle que $\beta_1(t) = R_1(t - T_1)_+$, et f_1 dispose d'une contrainte sur ces arrivées α_1^1, alors $\alpha_1^2(t) = (\alpha_1^1 \oslash \beta_1)_+(t) = \sigma_1 + \rho_1(t + T_1)$, par le résultat sur le flot de sortie présenté au paragraphe 1.3.

Hérédité : Montrons que $\forall k \in \{1, ..., n\}$, $\alpha_1^{k+1}(t) = \sigma_1 + \rho_1(t + \sum_{j \in M_1^k} T_j)$. Supposons que $\alpha_1^k(t) = \sigma_1 + \rho_1(t + \sum_{j \in M_1^{k-1}} T_j)$ et montrons que :
$\alpha_1^{k+1}(t) = \alpha_1^k(t + T_{k+1})$.

$$\begin{aligned} \alpha_1^{k+1}(t) &= (\alpha_1^k \oslash \beta_{k+1})_+(t) \\ &= \sigma_1 + \rho_1(t + \sum_{j \in M_1^{k-1}} T_j + T_{k+1}), \\ &\quad car\ (R_{k+1} - \rho_1) < 0. \end{aligned}$$

On vient de montrer que le flux de sortie d'un serveur quelconque s_k pour f_1 d'un réseau en tandem est une courbe $\alpha_1^{k+1}(t) = \sigma_1 + \rho_1(t + \sum_{j \in M_1^k} T_j)$.

Maintenant, nous ne supposons plus que f_1 est le seul présent dans le réseau, et l'on considère un flux quelconque f_i de ce réseau.
On sait que $\alpha_i^{k+1} = \alpha_i^k \oslash [\beta_k - \sum_{j | f_j \in s_k, j<i} \alpha_j^k]_+$. Nous voulons en déduire que $\alpha_i^{k+1}(t) = \sigma_i + \rho_i(t + \sum_{j \in M_i^k} T_j)$, en supposant que la formule est vrai pour les flux de priorité supérieure.

Initialisation : Nous supposons à nouveau que le réseau est composé d'un seul serveur, sans perte de généralité, on suppose qu'il s'agit de s_1. Calculons

α_i^2. On désignera par la lettre ℓ, pour chaque flux, la position du serveur 1 dans leur chemin.

$$\begin{aligned}
\alpha_i^2(t) &= (\alpha_i^1 \oslash (\beta_1 - \sum_{j|f_j \in s_1, j<i} \alpha_j^\ell)_+)_+(t) \\
&= \sup_{s \geqslant 0}\{\alpha_i^1(t+s) - \\
&\quad (\beta_1(s) - \sum_{j|f_j \in s_1, j<i} (\sigma_j + \rho_j(s + \sum_{n \in M_j^1} T_n^j)))_+\} \\
&= \alpha_i^1(t + T_1^i),
\end{aligned}$$

avec T_1^i date à laquelle la fonction $(\beta_1 - \sum_{j|f_j \in s_1, j<i} \alpha_j^\ell)_+$ s'annule. Donc, $T_1^i = \frac{\sum_{j|f_j \in s_1, j<i} \sigma_j + \rho_j(\sum_{\ell \in M_j^1} T_\ell^j) + R_1 T_1}{R_1 - \sum_{j|f_j \in s_1, j<i} \rho_j}$.

Hérédité : Par un raisonnement similaire, on trouvera :

$$\alpha_i^{k+1}(t) = \sigma_i + \rho_i(t + \sum_{j \in M_i^k} T_j^i), \text{ avec } T_j^i = \frac{\sum_{j|f_j \in s_1, j<i} \sigma_j + \rho_j(\sum_{\ell \in M_j^k} T_\ell^j) + R_1 T_1}{R_1 - \sum_{j|f_j \in s_1, j<i} \rho_j}.$$

Nous allons montrer la véracité du théorème suivant :

Théorème 7 *Propriété du délai*
Si pour tout flux f_ℓ ayant une priorité supérieure ou égale à f_i, traversant le serveur s_k, $\Sigma_{\ell \in M_i^k} \rho_\ell < R_k$, alors le délai maximal de f_i est borné au niveau du serveur s_k.

Démonstration :

Notons d le délai maximum atteint au serveur s_k pour le flux f_i. Nous allons montrer que ce délai est borné.
On a : $d = \text{hDev}(\alpha_i^k, (\beta_k - (\Sigma_{0 \leqslant \ell < i | f_\ell \in s_k} \alpha_\ell^k))_+)$.

$$\begin{aligned}
\beta_k(t+d) - (\sum_{0 \leqslant \ell < i | f_\ell \in s_k} \sigma_\ell^k + \rho_\ell^k(t+d)) &\geqslant \alpha_i^k(d) \\
R_k(t + d - T_k) &\geqslant \sum_{0 \leqslant \ell < i | f_\ell \in s_k} \sigma_\ell^k + \rho_\ell^k(t+d)) + \sigma_i^k + \rho_i^k(d) \\
d &\geqslant \frac{\sum_{0 \leqslant \ell < i | f_\ell \in s_k} \sigma_\ell^k + \sigma_i^k + R_k T_k}{R_k} \\
&\quad + \frac{((\sum_{0 \leqslant \ell < i | f_\ell \in s_k} \rho_\ell^k) - R_k)t}{R_k}.
\end{aligned}$$

En utilisant le même argument que celui du paragraphe 1.3 :

$$d < +\infty \Leftrightarrow R_k - \sum_{0 \leqslant \ell \leqslant i | f_\ell \in s_k} \rho_\ell^k > 0.$$

Ce qui est vrai par hypothèse, car $\rho_\ell^k = \rho_\ell$.

5 Étude du délai maximal pour des serveurs en tandem : politique de service à priorité fixe

5.1 Topologie des réseaux étudiés

Cette étude prend en compte une restriction de l'ensemble des réseaux, pour une topologie donnée. Il serait en effet complexe de considérer tous les types imaginables, qui ne se traitent pas tous de la même façon. Prenons l'exemple des réseaux cycliques : leur étude est complexe de part la dépendance de la charge à l'instant t par rapport au passé. Ainsi ces réseaux disposent d'une étude particulière. C'est pourquoi, dans notre travail nous nous restreignons à la classe des réseaux en tandem. De plus, les réseaux étudiés dans la suite respectent les hypothèses de stabilité énoncé dans le théorème 7.

5.2 Méthodologie

Dans le paragraphe 2.2 nous avons présenté une étude des réseaux acycliques dans le cas d'une politique de service arbitraire (*i.e.* à tout instant le serveur traite un des flux présents à son entrée choisi de manière arbitraire). Afin de déterminer le délai maximal subi par un flux f du réseau, ils définissent des contraintes linéaires traduisant le comportement du réseau pour une politique de service arbitraire et cherchent à maximiser la durée nécessaire à f pour traverser le réseau grâce à un solveur. Parmi ces contraintes, on repère notamment les α-contraintes sur les arrivées et β-contraintes sur les services.

Notre objectif est de définir un ensemble de contraintes assurant le système de priorité entre les flux choisis. Pour cela, nous allons reprendre le travail effectué dans l'article [03], et l'enrichir en ajoutant des contraintes de priorité.

Dans la suite, nous utiliserons les notations suivantes :

Notations :

- les flux du réseau sont annotés f_i tels que si f_u, f_v sont tels que $u < v$ alors $prio(f_v) > prio(f_u)$. Sur les figures de ce rapport, les priorités assignées aux flux, sont données entre parenthèses à côté de la flèche représentant le parcours du flux dans le réseau.
- le flux f_i est α_i-contraint tel que : $\alpha_i(t) = \sigma_i + \rho_i t$. Donc α_i sera toujours concave dans notre travail.
- le début de période chargée du serveur j (noté s_j) est noté t_{j-1}.
- le serveur j dispose d'une courbe de service strict β_j telle que : $\beta_j(t) = R_j(t - T_j)_+$. Le symbole R_j donne le taux de traitement des données par le serveur, et T donne le temps d'attente maximale des données à l'entrée du serveur au début de la période chargée. La fonction β_j est affine par morceaux et convexe.
- Lorsque l'on considère l'entrée et la sortie d'un flux quelconque au niveau d'un serveur quelconque, on utilisera le symbole A (resp. B) pour représenter la trajectoire d'entrée (resp. de sortie). Par contre si l'on étudie un flux

précis, par exemple f_i, en un serveur quelconque ; on utilisera les symboles A_i et B_i pour définir les flux d'entrée et de sortie du serveur. Enfin, si l'on considère un flux f_i au niveau du serveur s_j on prendra comme notation A_i^j et B_i^j.

5.3 Généralités

Si l'on s'intéresse au délai maximal subi par les données du flux f_ℓ alors il n'est pas nécessaire de considérer les flux moins prioritaires que lui. En effet, si à une date donnée, f_ℓ présente des paquets à traiter, alors f_ℓ sera toujours exécuté avant les flux moins prioritaires. Au contraire, la présence de flux moins prioritaires peut réduire le temps d'attente. En effet, supposons que f_ℓ s'exécute entièrement et que les autres flux plus prioritaires soient vides, alors le serveur se met en pause. Supposons qu'au bout d'un certain temps, des données du flux f_ℓ arrivent, elles peuvent alors attendre jusqu'à T unités de temps avant d'être traitées (dans le pire cas), comme le montre la figure 12. Cela correspond au temps de latence du serveur. Maintenant, si l'on ajoute des flux moins prioritaires que f_ℓ, alors ils seront traités après f_ℓ et le serveur ne sera pas en pause. Ainsi dès que des données de f_ℓ arrivent, elles seront traitées sans délai. C'est pourquoi, dans la suite on supposera disposer uniquement de ℓ flux lors de l'étude du délai maximal subit par f_ℓ.

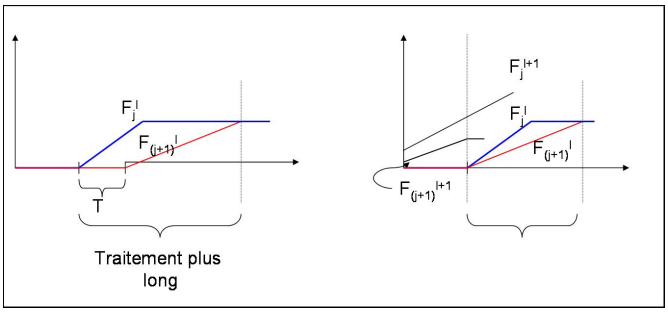

FIG. 12 – Explication schématique du phénomène introduit.

5.4 Programmes linéaires

Nous commençons par présenter les contraintes linéaires qui satisfont les priorités. Puis, nous allons montrer que ces contraintes suffisent à assurer les priorités sur les flux, pour le calcul du délai maximal subi par les données du flux f_ℓ. Pour ce faire, nous devons montrer que pour un serveur quelconque S placé en j^{eme} position dans un réseau entre t_{j-1} (début de période chargée de S) et t_j les contraintes assurent les priorités fixées. La contrainte d'arrivée du flux A_i^j est représentée par α_i, la contrainte sur le serveur s_j est β_j. Nous disposons d'un ensemble de dates :

– les débuts de période chargé de chaque serveur. Nous noterons ici $c_{j,0}$ le début de période chargée du serveur s_j, afin de fournir des formules générales.
– des dates intermédiaires introduites un peu plus loin, notées $c_{j,k} \forall k \in \{1, ..., \ell\}$.

5.4.1 Contraintes Linéaires

Soit λ l'ensemble des contraintes linéaires. Nous listons ci-dessous les types de contraintes présent dans notre programme ainsi que des exemples de ces contraintes appliqués à S sur lequel on souhaite calculer le délai subi par un flux f_ℓ. Nous indiquons par deb_i le premier serveur traversé par f_i. L'ensemble λ est composé des :

– ordres des dates :
$\{1, ..., \ell\}, \forall k > k' | (k, k') \in \{1, ..., \ell\}^2,\ c_{j,k} \geqslant c_{j,k'}$.
– α-contraintes pour tout flux, pour toute date introduite dans le programme :
$\forall k > k' | (k, k') \in \{1, ..., \ell\}^2, \forall j \in \{j | s_j \in S\}$,
$A_i^{deb_i}(c_{j,k}) - A_i^{deb_i}(c_{j',k'}) \leqslant \alpha_i(c_{j,k} - c_{j',k'})$;
et $\forall k, k' | (k, k') \in \{1, ..., \ell\}^2, \forall j > j' | (j, j') \in \{j | s_j \in S\}^2$,
$A_i^{deb_i}(c_{j,k}) - A_i^{deb_i}(c_{j',k'}) \leqslant \alpha_i(c_{j,k} - c_{j',k'})$;
– β-contraintes pour tout serveur sur leur période d'activité :
$\forall k > k' | (k, k') \in \{1, ..., \ell\}^2,\ \sum_{i | f_i \in s_j} B_i^j(c_{j,k}) - B_i^j(c_{j,k'}) \geqslant \beta_j(c_{j,k} - c_{j,k'})$;
– contraintes de croissance des flux pour toute date :
$\forall k > k' | (k, k') \in \{1, ..., \ell\}^2,\ A_i^j(c_{j,k}) \geqslant A_i^j(c_{j,k'})$;
– contraintes de causalité assurant que la quantité de donnée sortie à t est inférieure à la quantité de donnée entrée à t :
$\forall k \in \{1, ..., \ell\},\ A_i^j(c_{j,k}) \geqslant B_i^j(c_{j,k})$;
– contraintes de priorités (dont nous détaillons le principe ci-dessous).

Afin d'assurer les priorités pour S, nous définissons des dates intermédiaires notées $\forall i \in \{1, .., \ell - 1\}, c_{j,i}$, ainsi : $t_{j-1} \leqslant c_{j,i} \leqslant t_j$. Et telle que $\forall (i, k) \in \{1, .., \ell - 1\}^2$, avec $k > i, c_{j,i} \leqslant c_{j,k}$. La date $c_{j,1} = \sup\{t \in \mathbb{R}_+ | \forall i \in \{1, .., \ell - 1\}\ B_i^j(t) = A_i^j(t)\}$. Il s'agit donc de la dernière date à laquelle tous les flux de priorité supérieure à f_ℓ se sont vidés. On définit ensuite les autres dates intermédiaires par :

$$\begin{aligned} c_{j,i} \quad = \quad & \sup\{t \in \mathbb{R}_+ | \forall k \in \{1, .., \ell - 1 - i\},\ B_k^j(t) = A_k^j(t),\ \text{et} \\ & \forall k \in \{\ell - i + 1, .., \ell\},\ B_k^j(t) = B_k^j(c_{j,(i-1)})\}. \end{aligned}$$

Pour comprendre le principe de fonctionnement de ces dates, il faut bien noter que nous ne souhaitons pas assurer les priorités des flux à chaque instant d'exécution. En effet notre objectif est de calculer le pire délai de f_ℓ, et donc de s'assurer que la donnée étudiée de ce flux passe à travers le serveur une fois que tous les flux plus prioritaire sont servis.

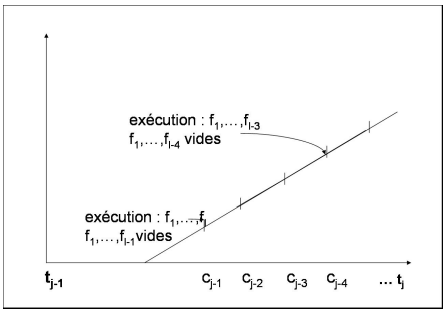

FIG. 13 – Explication schématique des contraintes de priorités.

Le principe de fonctionnement de ces dates intermédiaires est le suivant : pour que le flux le moins prioritaire puisse s'exécuté, il faut que tous les autres soient vides. Nous notons $c_{j,1}$ la dernière date à laquelle f_ℓ peut s'exécuter. De même le flux $f_{\ell-1}$ peut s'exécuter seulement quand $f_1, ...f_{\ell-2}$ sont vides. D'où la date $c_{j,1}$, et ainsi de suite. Sur la figure 13, nous avons représenté le flot de données minimal sortant d'un serveur quelconque s_j, sur l'intervalle de temps $[t_{j-1}, t_j]$, avec ces dates intermédiaires. Ainsi les priorités sont assurées. Après cette date, f_ℓ n'a plus la possibilité de s'exécuter et donc d'un point de vue de la donnée étudiée les priorités sont respectées. Dans le paragraphe 4.4.2, nous prouverons ce résultat.

Les contraintes de priorités sont alors définies en toutes dates :
$\forall j \in \{j | s_j \in S\}, \forall i \in \{1, .., \ell - 1\}, c_{j,i}$ par :
$\forall k \in \{1, .., \ell - 1 - i\},\ B_k^j(c_{j,i}) = A_k^j(c_{j,i})$
$\forall k \in \{\ell - i + 1, .., \ell\},\ B_k^j(c_{j,i}) = B_k^j(c_{j,(i-1)})$.

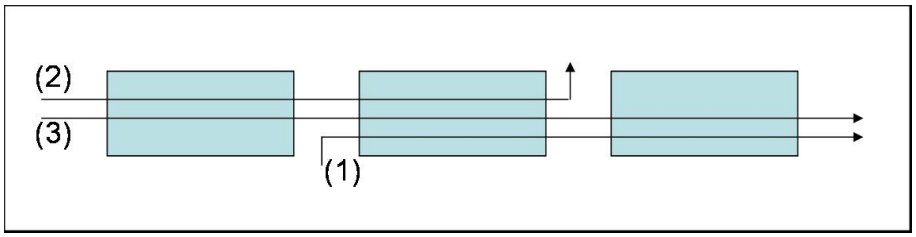

FIG. 14 – Schéma d'un réseau.

Exemple Nous prenons le réseau de la figure 14, pour en lister les contraintes linéaires décrivant le comportement du réseau, dans le cas d'une politique de service à priorités fixes. L'ensemble λ contient toutes les contraintes linéaires présentées présenté au paragraphe 2.2. Nous ajoutons ici uniquement les contraintes relatives aux priorités.

Les dates intermédiaires sont $c_{1,1},\ c_{2,1},\ c_{2,2}$ et $c_{3,1}$ ainsi que t_0, t_1, t_2, t_3 telles que $[t_0, t_1]$ est une période chargée pour s_1 (de même pour s_2 et s_3). Les trajectoires aux quelles on souhaite fournir les contraintes de priorités sont : A_2^1, A_3^1 flux d'entrée au premier serveur, B_2^1, B_3^1 flux de sortie au premier serveur et de même pour le second et troisième serveur. Il est à noter que $B_2^1 = A_2^2, B_3^1 = A_3^2$, et de même pour les autres serveurs. En effet, la trajectoire sortant du premier serveur est inchangée en entrant dans le second serveur, puisqu'entre ces deux points elle ne subi aucun traitement. Les contraintes de priorité sont alors :

Serveur 1
$A_2^1(c_{1,1}) = B_2^1(c_{1,1})$.
$B_3^1(t_1) = B_3^1(c_{1,1})$.
Serveur 2
$A_1^2(c_{2,1}) = B_1^2(c_{2,1})$.
$A_2^2(c_{2,1}) = B_2^2(c_{2,1})$.
$A_1^2(c_{2,2}) = B_1^2(c_{2,2})$.
$B_3^2(c_{2,2}) = B_3^2(c_{2,1})$.
$B_2^2(t_2) = B_2^2(c_{2,2})$.
$B_3^2(t_2) = B_3^2(c_{2,2})$.
Serveur 3
$A_2^1(c_{3,1}) = B_2^1(c_{3,1})$.
$B_3^1(t_1) = B_3^1(c_{1,1})$.

Afin de simplifier la rédaction des programmes linéaires répertoriant les contraintes des réseaux, nous avons créé un programme de génération de contraintes. Il suffit alors de lui transmettre un simple fichier texte contenant les caractéristiques clefs du réseau (nombre de flux, de serveurs, α et β contraintes de chacun), pour obtenir le programme linéaire complet correspondant.

Exemple Si l'on veut obtenir les contraintes linéaires associées au réseau 14, il suffit d'écrire ceci :

nservers 3 // (commentaire : il y a 2 serveurs dans le réseau.) nflows 3 // (et 2 flux) objective d 3 // (l'objectif est le calcul du délai pour le flux 2.) // (la partie qui suit donne les fonctions β associées à chaque serveur.)
beginservers server 1 $[R_1, T_1]$ server 2 $[R_2, T_2]$ server 3 $[R_3, T_3]$ endservers
// (la partie qui suit donne la fonction α associée au flux // et les serveurs par lesquels il passe.) beginflows flow 1 (2,3) $[\sigma_1, \rho_1]$ flow 2 (1,2) $[\sigma_2, \rho_2]$ flow 3 (1,3) $[\sigma_3, \rho_3]$ endflows

5.4.2 Application des nœuds à capacités variables

Nous souhaitons tout d'abord montrer qu'entre deux dates intermédiaires u et v (définies dans le lemme suivant) les priorités sont assurées par la seule connaissance de ces dates. Nous allons appliquer les résultats sur les nœuds à capacité variable pour prouver que les priorités sont assurées entre u et v.

Dans cette partie, nous considérons le serveur s_1, premier serveur d'un réseau quelconque. Soit $A(t) = \Sigma_i A_i(t), A_i(t) \in \mathcal{F}$ la fonction comptant la quantité de données arrivées en s_1 entre $start(t)$ et t. Soit $C(t) \in \mathcal{F}$ la quantité de données traitées à l'instant t par S. La date u est telle que $A_i(u) = B_i(u), \forall i \in \{1, ..., \ell - 1 - n\}$ et v telle que : $A_i(v) = B_i(v), \forall i \in \{1, ..., \ell - 2 - n\}$. Soit $t \in [u, v]$, alors $B(t) = A(u) + C(t) - C(u)$.

Lemme 2 *Soit $\overrightarrow{B}_i(t)$ une trajectoire construite à partir des $A_i(t)$ telle que :*

$$\overrightarrow{B}_i(t) = \inf_{t1 \leqslant s \leqslant t} \{\overrightarrow{A}_i(s) + C^{(i)}(t) - C^{(i)}(s)\},$$

où $\forall 1 \leqslant i \leqslant \ell, C^{(i)}(t) = C(t) - \Sigma_{k=1, k \leqslant i-1} \overrightarrow{B}_k(t)$ et $\overrightarrow{A}_i = A_i$.

Alors $\overrightarrow{B}_i(t)$ est une trajectoire qui respecte les priorités et qui satisfait les contraintes de service β.

Démonstration :
De manière évidente $\overrightarrow{B}_i(t)$ est une trajectoire qui respecte les priorités. Montrons que : $\overrightarrow{B}_i(t) \leqslant \overrightarrow{A}_i(t)$, que $\overrightarrow{B}(t) = B(t)$, et que $\overrightarrow{B}_i(t)$ est croissante.

– $\overrightarrow{B}_i(t) \leqslant \overrightarrow{A}_i(t)$:

$$\begin{aligned} \text{On sait que } \forall 1 \leqslant i \leqslant \ell,\ \overrightarrow{B}_i(t) &= \inf_{t1 \leqslant s \leqslant t} \{\overrightarrow{A}_i(s) + C^{(i)}(t) - C^{(i)}(s)\} \\ \text{donc } \overrightarrow{B}_i(t) &\leq \overrightarrow{A}_i(t) + C(t) - C(t) \\ &= \overrightarrow{A}_i(t). \end{aligned}$$

Ainsi $\forall 1 \leqslant i \leqslant \ell, \overrightarrow{B}_i(t) \leq \overrightarrow{A}_i(t)$.

– $\overrightarrow{B}(t) = B(t)$:

$$\begin{aligned} \text{on a } B(t) &= C(t) - C(u) + A(u), \\ \text{et } \overrightarrow{B}_\ell(t) &= A_\ell(u) + C^{(\ell)}(t) - C^{(\ell)}(u), \\ \text{or } A_\ell(u) &= \overrightarrow{B}_\ell(u), \\ \text{donc } \overrightarrow{B}_\ell(t) &= \overrightarrow{B}_\ell(u) + C(t) - \Sigma_{1 \leqslant i \leq \ell-1} \overrightarrow{B}_i(t) - C(u) + \Sigma_{1 \leqslant i \leq \ell-1} \overrightarrow{B}_i(u), \\ D'où\ \Sigma_{i=1, i \leq \ell} \overrightarrow{B}_i(t) &= \Sigma_{i=1, i \leq \ell} \overrightarrow{B}_i(u) + C(t) - C(u). \end{aligned}$$

Finalement $\overrightarrow{B}(t) = B(t)$. Ainsi $\overrightarrow{B}_i(t)$ est une trajectoire qui satisfait les contraintes de service β.

– $\overrightarrow{B}_i(t)$ est croissante :
On a : $\overrightarrow{B}_i(t) = \inf_{u \leqslant s \leqslant t} \{\overrightarrow{A}_i(s) + C^{(i)}(t) - C^{(i)}(s)\}$
où $C^{(i)}(t) = C^{(i-1)}(t) - \overrightarrow{B}_{i-1}(t)$ et $C^1 = C$.

Commençons tout d'abord par montrer que $C^{(i)}$ est une fonction croissante quelle que soit la valeur de i. Nous allons montrer cela par récurrence. Initialisation : $C^{(1)}(t) = C(t)$. Or, on sait que C est une fonction croissante. Donc l'initialisation est vérifiée.

Supposons maintenant que $C^{(i)}$ est croissante et montrons que $C^{(i+1)}$ l'est aussi.

$$\begin{aligned} C^{(i+1)}(t) &= C^{(i)}(t) - \overrightarrow{B}_i(t), \\ C^{(i+1)}(t) &= C^{(i)}(t) - \inf_{u \leq s \leq t} \{\overrightarrow{A}_i(s) + C^{(i)}(t) - C^{(i)}(s)\}, \\ C^{(i+1)}(t) &= \sup_{u \leq s \leq t} \{C^{(i)}(s) - \overrightarrow{A}_i(s)\}. \end{aligned}$$

Soit $t' < t$ alors

$$C^{(i+1)}(t) \geq \sup_{u \leq s \leq t'} \{C^{(i)}(s) - \overrightarrow{A}_{i+1}(s)\} = C^{(i+1)}(t'),$$

donc $C^{(i+1)}$ est croissante.

Montrons maintenant que $\overrightarrow{B}_i(t)$ est croissante : $\overrightarrow{B}_i(t) = \inf_{u \leq s \leq t}\{\overrightarrow{A}_i(s) + C^{(i)}(t) - C^{(i)}(s)\}$. Posons $t' < t$, alors :
$\overrightarrow{B}_i(t) = \inf_{u \leq s \leq t'}\{\overrightarrow{A}_i(s) + C^{(i)}(t) - C^{(i)}(s)\} \wedge \inf_{t' \leq s \leq t}\{\overrightarrow{A}_i(s) + C^{(i)}(t) - C^{(i)}(s)\}$
Or $\inf_{u \leq s \leq t'}\{\overrightarrow{A}_i(s) + C^{(i)}(t) - C^{(i)}(s)\} \geqslant \inf_{u \leq s \leq t'}\{\overrightarrow{A}_i(s) + C^{(i)}(t') - C^{(i)}(s)\} = B_i(t')$ (car (C^i) est croissante)
$\inf_{t' \leq s \leq t}\{\overrightarrow{A}_i(s) + C^{(i)}(t) - C^{(i)}(s)\} \geqslant \inf_{s=t}\{\overrightarrow{A}_i(t) + C^{(i)}(t) - C^{(i)}(t)\} = \overrightarrow{A}_i(t)$
Or $A_i(t) \geqslant \overrightarrow{A}_i(t')$, (par hypothèse)
et $A_i(t') \geqslant \overrightarrow{B}_i(t')$, (par le premier point de la démonstration)
Donc $\overrightarrow{B}_i(t) \geqslant \overrightarrow{B}_i(t')$.

Le lemme que nous venons de prouver permet de montrer que les trajectoires à la sortie du premier serveur respecte les priorités et satisfait les contraintes de service β. Le lemme suivant permet la généralisation de ce résultat à la sortie de s_j.

Lemme 3 *Soit $\overrightarrow{B}_i^j(t)$ une trajectoire de sortie d'un serveur quelconque d'un réseau, construite à partir des $\overrightarrow{A}_i^j$ telle que :*

$$\overrightarrow{B}_i^j(t) = \inf_{t1 \leqslant s \leqslant t}\{\overrightarrow{A}_i^j(s) + C^{(i)}(t) - C^{(i)}(s)\},$$

où $\forall 1 \leqslant i \leqslant \ell, C^{(i)}(t) = C(t) - \Sigma_{k=1,k\leqslant i-1}\overrightarrow{B}_k^j(t)$ et sachant que $\overrightarrow{A}_i^j = \overrightarrow{B}_i^{j-1}(t)$, $\overrightarrow{A}_i^j$ est croissante, que $\overrightarrow{A}_i^j \leqslant A_i$ et que $\overrightarrow{A}^j = A$.

Alors $\overrightarrow{B}_i^j(t)$ est une trajectoire qui respecte les priorités et qui satisfait les contraintes de service β.

Démonstration : La preuve de ce lemme s'obtient en effectuant la preuve du lemme précédent par récurrence.

Nous venons de prouver que nous sommes capables de construire une trajectoire qui respecte les priorités à partir des contraintes du *Network Calculus* et des contraintes linéaire. Ainsi, nous savons que les contraintes définies fournissent une borne maximale du délai dans le cas d'une politique de service à priorité. Il nous faut maintenant montrer qu'à partir des contraintes linéaires transmises au solveur, il fournit des trajectoires respectant les contraintes du *Network Calculus*. C'est l'objectif du paragraphe suivant.

5.4.3 Reconstruction des trajectoires

Les lemmes 3 et 4, nous permettent directement de constater que l'on a besoin d'ajouter des contraintes seulement aux dates t_{j-1}, $c_{j,i}$, t_j afin de garantir les priorités qu'un point de vue du délai maximal du flux f_ℓ. Nous devons maintenant montrer que pour tout f, flux quelconque respectant les contraintes

linéaires définies, il vérifie les contraintes du *Network Calculus*. Nous commençons par montrer que la fonction d'arrivées est $\alpha - contrainte$, puis nous montrons que la courbe de sortie de S est $\beta - contrainte$. On rappel que α est affine par morceaux, concave et β affine par morceaux, convexe.

Lemme 4 *Soit* $c_n > t > c_{n-1},\ f(t) = \min\{\min_{m<n}\{\alpha_i(t-c_m)+A_i^j(c_m)\}; A_i^j(c_n)\}$. *Alors, la fonction* f *construite à partir des valeurs* A_i^j *est* α*-contrainte.*

Démonstration

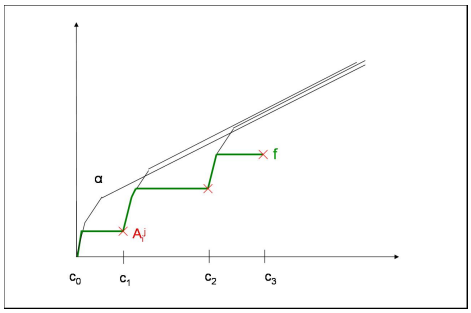

FIG. 15 – Schéma représentant la fonction f construite à partir des contraintes α et des valeurs déterminées par le solveur.

Nous avons représenté sur la figure 15, une fonction alpha convexe, affine par morceau, ainsi que la fonction f construite à partir des points A_i^j et d'α. Soient u, v telles que : $u > v$, $c_n > u > c_{n-1}$, $\forall ktelquek \leqslant n,\ c_k > v > c_{k-1}$.

Par définition de f si v appartient à α_i alors $f(v) - f(c_{k-1}) = \alpha_i(v - c_{k-1})$.
Sinon, par définition $f(v) - f(c_{k-1}) = f(c_k) - f(c_{k-1}) = \alpha_i(c_k - c_{k-1})$.
On sait, par définition de f, que : $f(u) - f(c_{k-1}) \leq \alpha_i(u - c_{k-1})$.
$f(u)-f(v) = f(u)-f(ck) = f(u)-f(c_{k-1})-[f(c_k)-f(c_{k-1})] \leq \alpha_i(u-c_{k-1})-\alpha_i(c_k - c_{k-1})$
Or α est sous-additive donc $\alpha_i(u-c_{k-1})-\alpha_i(c_k-c_{k-1}) \leq \alpha_i(u-c_k) \leq \alpha_i(u-v)$ car $u - c_k < u - v$

Lemme 5 *Soit* $c_k > t \geqslant c_{k-1},\ f^j(t) = \max\{\Sigma_{i|f_i \in S} B_i^j(c_k), \max_{n|c_n \leqslant t}\{\beta(t - c_n) + \Sigma_{i|f_i \in S} B_i^j(c_n)\}\}$. *Alors la fonction* f^j *construite à partir des valeurs* B_i^j *respecte les* β*-contraintes.*

Démonstration

On sait que $\forall c_n, c_{n'} f^j(c_n) \geq \beta(c_n - c_{n'}) + \Sigma B_i^j(c_{n'})$
Soit $t \in [c_{n'}, c_n]$, alors $f^j(c_n) \geq \beta(c_n - t) + f^j(t)$

Car soit $\beta(c_n - c_{n'}) + \Sigma B_i^j(c_{n'}) \geq \beta(c_n - t) + f^j(t)$ où $\exists c_{n''} > c_n$ tel que $\beta(c_{n''} - c_n) + \Sigma B_i^j(c_n) \geq \beta(c_{n''} - t) + f^j(t)$ dû au fait que β est affine (par morceaux) croissante.

Soit $t \in [c_{n'}, c_n]$,
Alors $\exists t'$ tel que $f^j(t) \in \beta(t - t') + B_i^j(t')$.
Donc $\forall t'', f^j(t) \geq \beta(t - t'') + B_i^j(t'')$.

En conclusion :

Théorème 8 *L'ensemble λ des contraintes linéaires assure les priorités assignés aux flux entre t_{j-1} et t_j pour le calcul du délai maximal. De plus, toutes les trajectoires possibles obtenues par le solveur à partir de Λ vérifient les contraintes α et β.*

Si l'on considère un réseau à n serveurs et que l'on calcule le délai subit par le flux f_ℓ, notre programme génère un nombre de contrainte de l'ordre de $\ell^3 \times n^2$.

On dispose le $\ell + 1$ dates intermédiaires pour chacun des n serveurs. Ce qui fait donc ℓn contraintes sur l'ordre des dates. Les α-contraintes sont exprimés pour chaque duo de dates et pour tout les flux, ce qui implique un ajout de dates de l'ordre de $(\ell n) \times (\ell n)\ell$. Les contraintes de causalité sont exprimées pour tout flux, par date voisines et en chaque serveur, donc on ajoute ℓn contraintes. De même, les contraintes de croissance sont exprimées par doublon de dates voisines, en chaque serveur, donc : ℓn contraintes. Le service est exprimé $\ell^2 n$ fois, car on exprime les β-contraintes pour chaque dates de la période d'activité en chaque serveur. Les début de période chargée sont exprimées une fois par serveur ce qui donne n contraintes, enfin on génère $\ell^2 n$ contraintes de priorité qui correpond au contraint sur la période d'activité pour chaque flux et en chaque serveur.

5.4.4 Contraintes linéaires

Un problème demeure : actuellement, pour obtenir ce délai, il faut considérer quels sont les flux auxquels il reste des données non traitées après t_j, et donc faire une disjonction de cas, ce qui nous oblige à travailler avec plusieurs programmes. Le but de la prochaine partie est de prouver que cette disjonction est inutile.

Grâce au lemme suivant, nous allons montrer qu'il n'est pas nécessaire de faire plusieurs programmes linéaires en dissociant le nombre de flux présents dans le système à la date t_j. En effet, nous montrons que considérer le cas où tous les flux peuvent être présent après t_j permet de retrouver les cas où il y en a moins, et donc de calculer le délai maximal.

Lemme 6 *Soit une trajectoire T telle qu'après t_j les flux $f_1, ..., f_{\ell-k-2}$ ne sont pas vides, $f_{\ell-k-1}, ..., f_\ell$ sont vides. Soit $c_1, ..., c_{k-2}$ les dates intermédiaires associées à T. Soit T' une seconde trajectoire telle qu'après t_j les flux $f_1, ..., f_{\ell-k-1}$*

ne sont pas vides, $f_{\ell-k}, ..., f_\ell$ sont vides. Soit $c_1, ..., c_{k-1}$ les dates intermédiaires associées à T'. Alors les contraintes associées à T' expriment également T.

Démonstration :

Considérons d'abord T. Soit $\forall n \in \{1, ..., k-2\}$ c_n ces dates intermédiaires. Les contraintes en ces dates sont alors :

$$\begin{aligned}
A_i(c_n) &= B_i(c_n)\ \forall i \in \{1, ..., \ell - n - 1\}, \\
B_i(c_n) &= B_i(c_{n-1})\ \forall i \in \{\ell - n, ..., \ell\}, \\
\Sigma_{0 \leqslant i \leqslant \ell} B_i(c_n) &\geqslant \Sigma_{0 \leqslant i \leqslant \ell} B_i(c_{n'}) + \beta(c_n - c_{n'}),\ \forall n' < n, \\
\Sigma_{0 \leqslant i \leqslant \ell} B_i(c_n) &\geqslant \Sigma_{0 \leqslant i \leqslant \ell} B_i(t_{j-1}) + \beta(c_n - t_{j-1}).
\end{aligned}$$

Considérons maintenant la trajectoire T'. Soit $\forall n \in \{1, ..., k-1\}$ c_n. Les contraintes sont :

$$\begin{aligned}
A_i(c_n) &= B_i(c_n)\ \forall i \in \{1, ..., \ell - n - 1\}, && (1) \\
B_i(c_n) &= B_i(c_{n-1})\ \forall i \in \{\ell - n, ..., \ell\}, && (2) \\
\Sigma_{0 \leqslant i \leqslant \ell} B_i(c_n) &\geqslant \Sigma_{0 \leqslant i \leqslant \ell} B_i(c_{n'}) + \beta(c_n - c_{n'}),\ \forall n' < n, && (3) \\
\Sigma_{0 \leqslant i \leqslant \ell} B_i(c_n) &\geqslant \Sigma_{0 \leqslant i \leqslant \ell} B_i(t_{j-1}) + \beta(c_n - t_{j-1}). && (4)
\end{aligned}$$

Autrement dit, $\forall n \in \{1, ..., k-2\}$ les contraintes sont les mêmes. Reste à considérer la date intermédiaire c_{k-1}. A cette date on a :
$A_i(c_{k-1}) = B_i(c_{k-1})\ \forall i \in \{1, ..., \ell - k - 2\}$ (1).
Or en c_{k-2} on a $A_i(c_{k-2}) = B_i(c_{k-2})\ \forall i \in \{1, ..., \ell - k - 3\}$.
Ainsi en posant $c_{k-1} = c_{k-2}$ (1) est vérifié. Et de même pour (2,3).

Conclusion : On a montré qu'en posant $c_{k-1} = c_{k-2}$ on peut exprimer T avec les contraintes de T'.

En itérant ce procédé, on montre qu'en exprimant uniquement le cas où tous les flux sont tous présents après t_j est nécessaire. Ainsi, lors des tests, nous ne fournirons qu'un seul programme de contraintes linéaires et le solveur trouvera par lui-même les flux devant être présents après t_j afin de fournir le pire cas.

6 Comparaison des méthodes de calcul de délais

6.1 Étude d'un premier type de réseau

Le réseau dessiné ci-dessous est composé de $n+2$ flux et $2n+1$ serveurs en tandem. Le flux f_{n+2-i} apparait dans le réseau au serveur i et en sort au serveur $n+2+(n+2-i)$. La priorité de ce flux est i. Dans cette partie, on considérera que tous les serveurs ont la même courbe de service minimal : β telle que, $\beta(t) = R(t-T)_+$ et les flux, la même courbe d'arrivée : $\alpha(t) = \sigma + \rho(t)$.

Nous allons étudier le délai du flux le moins prioritaire de ce réseau en utilisant les différentes méthodes présentées précédemment. Nous commencerons

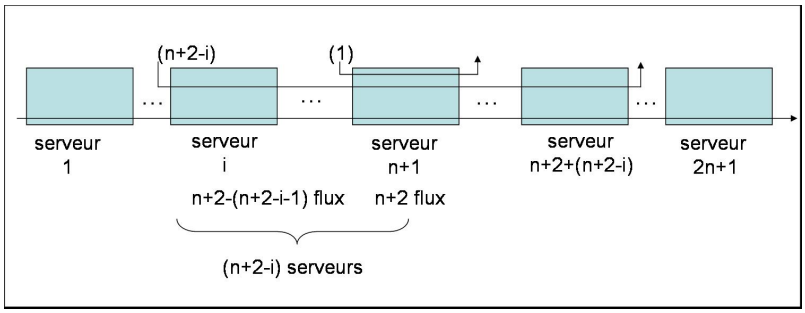

FIG. 16 – Représentation du système étudié.

par appliquer le résultat de l'article [02], puis nous adopterons la méthode SFA de l'article [09], enfin, nous exécuterons notre programme linéaire et comparerons les valeurs ainsi obtenues.

6.1.1 Méthode de composition des courbes de services résiduelles

Cette méthode évalue la courbe de service résiduelle associée au réseau en concaténant les courbes de service résiduel des serveurs traversés par le flux dont on calcule le délai. Cette méthode s'applique uniquement lorsque les priorités sont fixes avec des flux imbriqués, tels que f_k passe au moins à travers les serveurs de $f_{k'}$ si $k > k'$. La courbe de service résiduel du flux f_i est β'_i. Avec $\beta'_i = \bigotimes_{j\in N_i} \beta_j \bigotimes_{k\in M_i} (\beta'_k - \alpha_k)_+$ où N_i est l'ensemble des serveurs traversés par f_i et qui ne sont pas traversés par des flux plus prioritaires, et M_i est l'ensemble des serveurs traversés par f_i ayant des flux plus prioritaires.

Lemme 7 *Le délai maximal subi par le flux f_{n+2} est $d_m = \frac{\sigma+(R-(n+1)\rho)t'_{2n+1}}{R-(n+2)\rho}$.*

Démonstration
Le délai maximal subi par le flux f_{n+2} défini par : $d_m = hDev(\alpha, \beta_{Tot})$ où β_{Tot} est la courbe de service résiduelle du réseau pour f_{n+2} :
$\beta_{Tot} = \beta^*_{2n+1} = ((\beta \otimes \beta_{2n-1})_+ \otimes \beta)_+$ où $\beta_1 = (\beta - \alpha)_+$.

Calculons une expression de β_{Tot} en fonction des paramètres R, T,σ et ρ par récurrence.
Initialisation : Par un résultat précédent on sait que :
$\beta_1(t) = (\beta - \alpha)_+(t) = (R - \rho)(t - t'_1)_+$, où $t'_1 = \frac{\sigma+RT}{R-\rho}$ est la date pour laquelle la fonction $(\beta - \alpha)_+$ s'annule.

Soit $i < n$, supposons que $\beta_{2i-1}(t) = (R - i\rho)(t - t'_{2i-1})_+$. Montrons que cette formule est vraie pour β_{2i+1}.
Soit $\beta^*_{2i+1} = ((\beta \otimes \beta_{2i-1})_+ \otimes \beta)_+$.
On applique un résultat pour la convolution prouvée dans le livre [08] : La convolution de fonctions convexes, affine par morceaux, résulte en la concaténation par pentes croissantes des segments affines. Donc :

$\beta^*_{2i+1}(t) = (R - i\rho)(t - (t'_{2i-1} + 2T))_+$.
Et on pose : $\beta_{2i+1} = (\beta^*_{2i+1} - \alpha)_+$.
Donc : $\beta_{2i+1}(t) = (R - (i+1)\rho)(t - t'_{2i+1})_+$, avec t'_{2i+1} la date à laquelle $(\beta^*_{2i+1} - \alpha_+)$ s'annule :

$$\begin{aligned}
\beta^*_{2i+1}(t'_{2i+1}) &= \alpha(t'_{2i+1}) \\
(R - i\rho)(t'_{2i+1} - (t'_{2i-1} + 2T)) &= \sigma + \rho(t'_{2i+1}) \\
(R - (i+1)\rho)t'_{2i+1} &= (t'_{2i-1} + 2T)(R - i\rho) + \sigma \\
t'_{2i+1} &= \frac{(t'_{2i-1} + 2T)(R - i\rho) + \sigma}{(R - (i+1)\rho)}.
\end{aligned}$$

Le calcul du délai maximal se fait ensuite par un calcul similaire à celui présenté en partie 1.3.

6.1.2 Méthode SFA

Le principe de la méthode SFA consiste également à calculer la courbe de service résiduel minimal du réseau puis à calculer la déviation horizontale maximale avec la courbe des arrivées du flux étudié. Le principe est que la courbe de service résiduel du serveur k est β_k telle que, $\beta_k = (\beta - \sum_{i \in M_i} \alpha_i)_+$ où M_i est l'ensemble des flux du serveur k plus prioritaire que f_i.

Lemme 8 *Le délai maximal subi par le flux f_{n+2} est :*

$$d = hDev(\alpha(t), (R - (n+1)\rho)(t - \sum_{j \in \{1,\dots,n+1\}} m_j - \sum_{j \in \{1,\dots,n+1\}} m'_j)_+),$$

où : $m_j = \frac{(j+1)*\sigma + RT + \sum_{\ell=0}^{j} \rho(\sum_{k=0}^{\ell-1} t'_k)}{R - (j+1)\rho}$
et $m'_j = \frac{(n+1-j-2)*\sigma + RT + \sum_{\ell=j+2}^{n+1} \rho(\sum_{k=0}^{\ell-1} t'_k + \sum_{k=1}^{j} t_{k,j})}{R - (n+1-j-2)\rho}$
Et avec $t'_k = \frac{k\sigma + RT + \sum_{i=1}^{k-1} \rho(\sum_{j=0}^{i-1} t'_j)}{R - k\rho}$, *et*
$t_{k,j} = \frac{(k-j-1)\sigma + \sum_{i=j+1}^{k-1} \rho(\sum_{\ell=0}^{k-2} t'_\ell + \sum_{\ell=i}^{k-1} t_{\ell,j})}{R - (k-j)\rho}$ $\forall j > 1$
Et $t_{i,1} = \frac{(i-2)\sigma + \sum_{\ell=2}^{i-1} \rho(T + \sum_{k=1}^{\ell-1} t'_k) + RT}{R - (i-2)\rho}$

Démonstration La preuve est présentée en annexe.

6.1.3 Comparaison des résultats

Sur la figure 17, nous avons regroupé les résultats obtenus pour les différentes méthodes présentées précédemment. Ici, nous avons fait varier le nombre de serveurs (et donc le nombre de flux). Chaque serveur du réseau est contraint par la même courbe de service : $\beta(t) = 28(t-5)_+$. Aux flux est associée une même contrainte : $\alpha(t) = 2 + 3t$.

Les barres bleues représentent les délais obtenus par notre méthode. Les violettes désignent les bornes calculés par la méthode introduite dans l'article

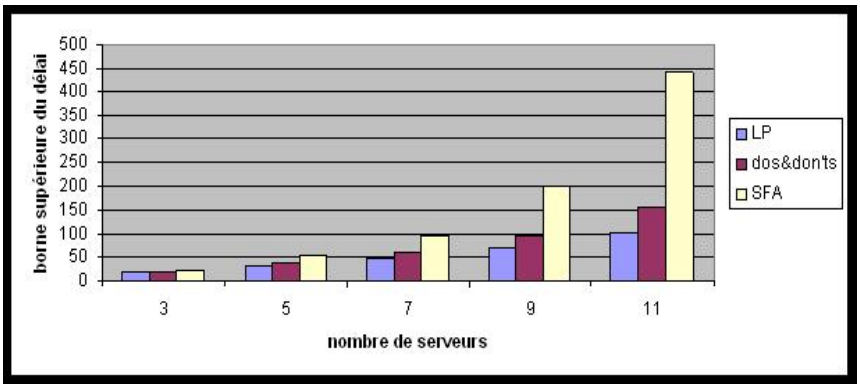

FIG. 17 – Diagramme représentant les résultats obtenus pour le réseau étudié avec différentes méthodes.

[02], pour qu'une donnée du flux le moins prioritaire traverse le réseau. Nous avons obtenu les résultats grâce au lemme 5 énoncé précédemment. Enfin le jaune donne ce délai par la méthode SFA. Ces derniers ont été calculés à partir des formules définies au lemme 6.

On constate que les résultats de la méthode de programmation linéaire avec priorités et celle de l'article [02] fournissent des résultats assez proche, bien que l'écart tende à augmenter en même temps que le nombre de serveurs. Cependant que la méthode SFA fournit des valeurs très pessimistes. Hélas pour que la méthode de l'article [02] soit applicable il faut pouvoir isoler les serveurs traversés par le flux le plus prioritaire, afin de calculer sa courbe de service résiduelle, puis faire de même pour le second flux et ainsi de suite. En l'occurrence la méthode est applicable car le flux f_{i+1} passe au moins à travers les serveurs par lesquels passe f_i, mais pour le réseau de la figure 19 ce n'est plus le cas. Ainsi, en plus de fournir des résultats plus fins des délais, notre méthode s'applique à un plus grand spectre de réseaux.

La démarche utilisée par SFA, elle aussi, s'applique à tous les réseaux, cependant ces résultats sont bien plus pessimistes. En moyenne, sur cet exemple, les délais calculés sont 50% à 100% plus élevés que notre approche.

Nous souhaitons étudier sur cet exemple l'impact que la variation des paramètres des fonctions α et β. Nous avons effectué cette étude, en considérant un réseau à 9 serveurs, sur le paramètre R qui correspond au taux de données traitées par les serveurs à chaque instant. Nous avons représenté les résultats sur la figure 18 ci-dessous, pour les méthodes de calcul de programmation linéaire et l'approche de l'article [03].

L'approche de l'article [03] procure des résultats supérieurs à ceux de notre travail. De plus, plus le taux d'utilisation du réseau est important, moins les résultats de cette méthode sont proches des notres, et sont donc moins fiables.

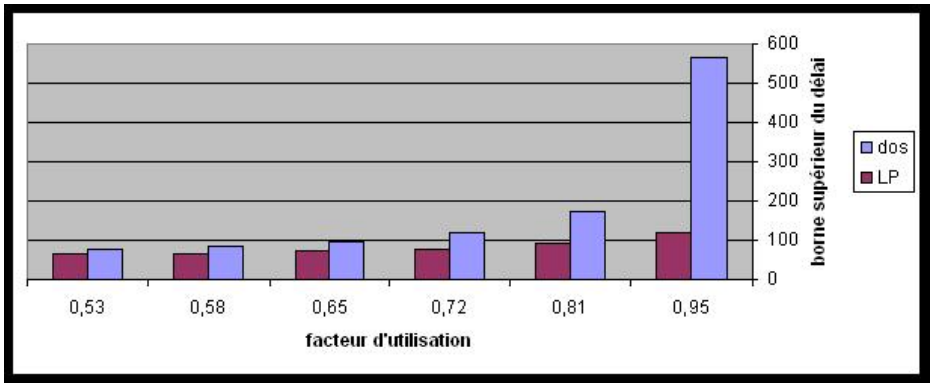

FIG. 18 – Diagramme représentant les résultats obtenus pour le réseau étudié avec différentes méthodes en faisant varier R.

6.2 Étude d'une seconde topologie de réseau

Le réseau dessiné ci-dessous est composé de $n+2$ flux et n serveurs en tandem. Le flux f_{n+1-i} apparaît dans le réseau au serveur i et en sort au serveur $i+2$. La priorité de se flux est i. Dans cette partie, on considérera que le serveur s_j dispose d'une courbe de service minimale : $\beta_j(t) = R_j(t - T_j)_+$ et le flux f_i est muni d'une courbe d'arrivée : $\alpha_i(t) = \sigma_i + \rho_i t$.

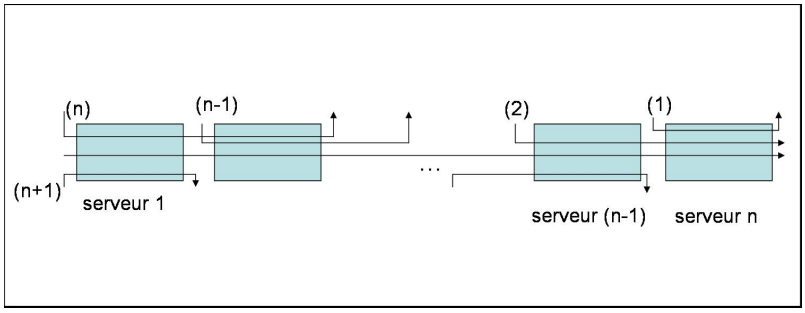

FIG. 19 – Représentation du système étudié.

Nous allons calculer le délai de ce réseau pour le flux f_{n+2} par la méthode SFA puis par nôtre méthode, afin d'en comparer les résultats.

6.2.1 Méthode SFA

Lemme 9 *Le délai maximal subi par le flux f_{n+2} est défini par : $d_m = hDev(\alpha_{n+2}, \beta_{Tot})$, où $\beta_{Tot}(t) = \min_{1 \leqslant i \leqslant n}\{(R - \rho_{n+2-i} - \rho_{n+1-i})\}(t - \sum_{1 \leqslant i \leqslant n} t'_i)_+$,*
et $t'_i = \frac{\sigma_{n+2-i} + \sigma_{n+1-i} + R_i T_i + \rho_{n+2-i-1} * T_{i-1}}{R_i - \rho_{n+2-i} - \rho_{n+1-i}} \forall i \in \{2, ..., n\}$, $t'_1 = \frac{\sigma_{n-2} + \sigma_{n-1} + RT}{R - \rho_{n-2} - \rho_{n-1}}$.

Démonstration

La service résiduel du réseau pour le flux f_{n+2} est : $\beta_{Tot} = (\beta - \alpha_{n+1} - \alpha_n)_+ \bigotimes (\beta - \alpha_{n+1-i} - \alpha_{n+2-i} \oslash \beta)_+$. En reprenant le même raisonnement que dans la preuve du lemme précédent, on retrouve les valeurs annoncées dans le lemme.

6.2.2 Comparaison des résultats

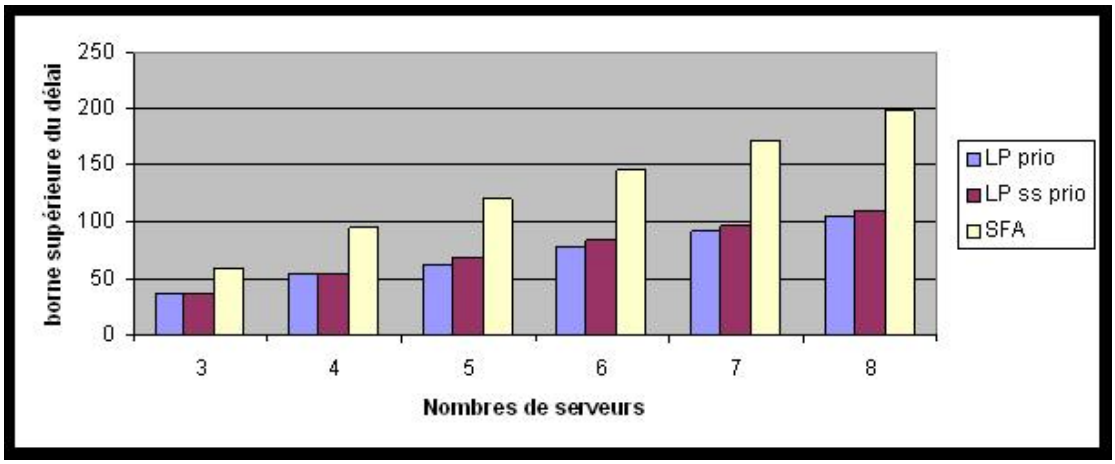

FIG. 20 – Diagramme représentant les résultats obtenus pour le réseau étudié avec différentes méthodes.

Sur la figure 20 représentée ci-dessus, nous avons regroupé les résultats obtenus pour les différentes méthodes présentées précédemment. Ici, nous avons fait varier le nombre de serveurs (et donc le nombre de flux). Chaque serveur du réseau est contraint par la même courbe de service β telle que, $\beta(t) = 10(t-5)_+$. Aux flux f_1, f_2, f_3 ainsi qu'aux avant-dernier et dernier flux du réseau, on associe toujours la même courbe d'arrivée : $\alpha(t) = 3 + 4t$. Puis les flux ajoutés au fur et à mesure que l'on augmente la taille du réseau, partagent une même courbe des arrivées. Nous avons fait varier le nombre de serveur dans le réseau de 3 à 8.

Les barres jaunes représentent les délais obtenus par notre méthode. Ensuite, les bandes violettes désignent le temps calculé par le solveur de programme linéaire à politique arbitraire, pour qu'une donnée du flux le moins prioritaire traverse le réseau. Enfin le bleu donne ce délai par la méthode SFA. Ces derniers ont été calculés à partir des formules définies au lemme 7.

On constate que cette dernière méthode fournit des résultats bien plus pessimistes que les autres. En effet, la différence entre SFA et notre méthode est 50% à 100% plus élevée. La méthode de calcul par programmation linéaire pour une

politique aveugle génère des valeurs plus hautes que notre méthode. Ce résultat est tout à fait prévisible puisqu'elle est moins restrictive que la politique à priorité fixe. On peut néanmoins remarquer que l'écart entre ces programmations linéaires est assez faible. En effet, nos délais sont 3% à 4% plus faibles que la politique aveugle.

Conclusion Nous venons d'étudier deux topologies de réseaux. Nous avons remarqué que la méthode SFA fournissait des résultats bien plus élevés que les autres méthodes et que de plus, elle est assez complexe à calculer. L'approche présentée dans l'article [02] fournit des résultats plus fins. Cependant elle s'applique à un sous-ensemble plutôt restrictif des réseaux acycliques, quand notre démarche s'applique à tous. Enfin nous avons pu vérifier que nos résultats étaient inférieurs à la méthode de programmation linéaire pour la politique arbitraire.

6.3 Limite du calcul de délai pour les politiques à priorité fixe à l'aide des contraintes linéaires

Jusqu'ici nous avons montré que nos contraintes permettaient d'assurer les priorités sur la première période d'activité du serveur étudié et que les contraintes du *Network Calculus* (α et β contraintes, croissance, causalité) étaient satisfaite sur cet intervalle. Une question reste en suspens : que se passe-t-il dans les serveurs en-dehors de la période d'activité étudiée ? Intuitivement, on pourrait penser que s'il se passe des phénomènes bizarres sur les autres intervalles de temps, ce n'est pas un problème puisque la donnée, du flux le moins prioritaire, dont on étudie le délai est passée. Évidement ce n'est pas aussi simple.

Nous avons exprimé les contraintes α entre toutes les dates intermédiaires existantes. Donc on sait que ces contraintes seront toujours assurées. Il en est de même pour la croissance et la causalité. Ainsi seules les contraintes sur le service peuvent nous faire défaut. Nous avons recherché un exemple dans lequel une telle contrainte ne serait pas assurée.

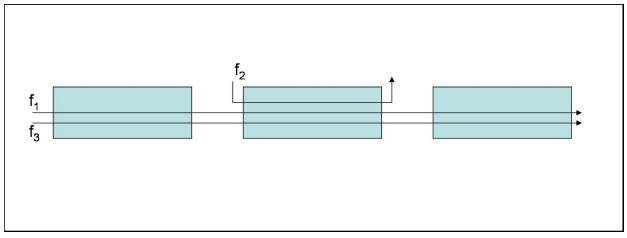

FIG. 21 – Représentation du réseau dans lequel une contrainte de service n'est pas contrôlée.

Nous avons étudié le comportement du réseau représenté en figure 21. Nous avions l'intuition que les priorités ainsi disposées pourraient permettre le genre

de comportement recherché. Nous avons choisi de prendre : $\alpha_1(t) = 2 + 2t$, $\alpha_2(t) = 3 + 3t$, $\alpha_3(t) = 2$ et $\beta_1 = 4(t-5)_+$, $\beta_2 = 8(t-4)_+$, $\beta_3 = 3(t-4)_+$.

Sur le schéma 22, nous avons tracé les trajectoires A_1, B_1, A_3 et B_3, ainsi que la contrainte de service du premier serveur dans l'intervalle de temps $[t_0, t_2]$. Il montre que la trajectoire B_1 est en-dessous de β_1 dans l'intervalle de temps $[t_1, t_2]$.

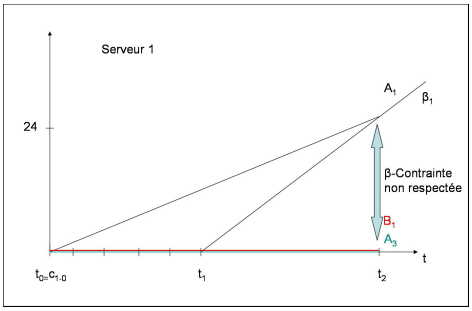

FIG. 22 – Schéma représentant les trajectoires des flux d'entrée et de sortie présent dans le serveur s_1 entre t_0 et t_2.

Il est à noter qu'aucune donnée n'est servie pour f_1 dans l'intervalle $[t_0, t_2]$ ce qui, intuitivement, paraît anormal puisque c'est le flux le plus prioritaire. Avant t_1 c'est normal puisque β_1 y est nulle. Mais passée cette date ce n'est plus le cas. La trajectoire B_1 est nulle jusqu'à t_3. Ainsi f_1 n'est servi qu'à partir de cette date alors que des données de ce dernier attendent à l'entrée du serveur dès t_0. Quel est l'impact de ce phénomène sur le délai calculé ?

Nous avons simulé ce réseau par programmation linéaire avec une politique de service à priorité fixe ou arbitraire, puis nous avons comparé les valeurs obtenues. Quelque soit la politique de service appliquée le délai calculé est le même. Il est évident que l'on ne peut obtenir une valeur plus élevée avec une politique de priorité fixe qu'avec une politique aveugle. En effet, l'ensemble de contraintes de nos programmes inclu toute les contraintes de la politique arbitraire. Ainsi, les trajectoires possibles dans notre cas le sont également pour la politique arbitraire. Comme le délai maximal est égale au pire délai calculé sur l'ensemble des trajectoires possibles, le délai que nous obtenons ne peut qu'être plus faible.

Nous voulons maintenant comparer notre délai avec celui obtenu par la méthode SFA.

Calcul du délai par la méthode SFA Nous devons donc commencer par le calcul de service résiduel total du réseau :

$$\beta_{Tot} = ((\beta_1 - \alpha_1)_+ \otimes (\beta_2 - \alpha_2 - (\alpha_1 \oslash \beta_1)_+)_+ \otimes (\beta_3 - ((\alpha_1 \oslash \beta_1)_+ \oslash \beta_2)_+)_+$$

$$\begin{aligned}
\beta'_1(t) &= (\beta_1(t) - \alpha_1(t))_+ \\
&= (R_1 - \rho_1)(t - t'_1)_+ \\
t'_1 &= \frac{\sigma_1 + R_1 T_1}{R_1 - \rho_1} \\
&= 11.
\end{aligned}$$

De même on calcule le service résiduel du second serveur :

$$\begin{aligned}
\beta'_2(t) &= (\beta_2(t) - \alpha_2(t) - (\alpha_1 \oslash \beta_1)(t)_+)_+ \\
&= \beta_2(t) - \alpha_2(t) - \alpha_1(t + T_1) \\
&= (R_2 - \rho_1 - \rho_2)(t - t'_2)_+ \\
t'_2 &= \frac{\sigma_1 + \sigma_2 + \rho_1 T_1 + R_2 T_2}{R_2 - \rho_2 - \rho_1} \\
&= 15,7.
\end{aligned}$$

Reste à calculer β'_3 :

$$\begin{aligned}
\beta'_3(t) &= (\beta_3(t) - \alpha_1(t + T_1 + T_2))_+ \\
&= (R_3 - \rho_1)(t - t'_1)_+ \\
t'_3 &= \frac{\sigma_1 + \rho_1(T_1 + T_2) + R_3 T_3}{R_3 - \rho_1} \\
&= 32.
\end{aligned}$$

Finalement, $\beta_{Tot} = 2(t - 58,7)_+$ et $d = hDev(\alpha_3, \beta_{Tot}) = 119.4$. Avec notre programme linéaire nous avons obtenu un délai de 52. Ainsi, malgré l'égalité des résultats avec la politique arbitraire, nous avons tout de même amélioreé, de manière significative, les résultats obtenus par rapport à la meilleure méthode existante jusque là.

Une prochaine étape dans nos recherches sera de déterminer sous quelles hypothèses le délai obtenu correspond exactement au pire délai du flux calculé sur le réseau. Nous aimerions également savoir dans quelles circonstances les priorités ne sont pas assurées, et tenter de résoudre ces problèmes.

Conclusion

L'enjeu de ce stage était de déterminer des bornes de performances déterministes telles que le délai ou la charge d'un réseau dans le cas de politique de service à priorités fixes. Nous avons découvert des méthodes basées sur l'application directe du *Network Calculus* effectuant déjà ce genre de calculs (tel que SFA). Nous avons tenté une autre approche pour fournir des résultats plus fins. Pour cela, nous nous sommes intéressés à la programmation linéaire. Alors, nous avons forgé un ensemble de contraintes linéaires qui permettaient de décrire le comportement d'un tel réseau (en créant un programme de génération automatique

de contraintes). Ainsi, nous avons mis les contraintes nécessaires au bon fonctionnement du réseau (contraintes du *Network calculus*) que l'on retrouvait déjà dans la programmation linéaire pour une politique arbitraire. Puis nous avons ajouté les contraintes définissant les priorités. Nous avons trouvé des résultats toujours inférieurs à la politique arbitraire. Mais, plus intéressant encore, nous avons montré que la méthode SFA génère des délais pessimistes.

Hélas, dans certains cas, les priorités ne sont pas assurées en-dehors des périodes chargées des serveurs. Dans ce cas, notre délai est égal à celui obtenu par programmation linéaire avec la politique arbitraire, mais est toujours inférieur aux résultats de SFA.

Désormais notre enjeu consiste à trouver des hypothèses sous lesquelles le délai obtenu est exactement le pire délai du réseau. De plus, nous souhaitons établir dans quel cas les priorités ne sont pas respectées, pour tenter d'y remédier.

7 Annexe A : Preuve du lemme 7

Le délai maximal subit par le flux $f_{(n+2)}$ est défini par : $d_m = hDev(\alpha, \beta_{Tot})$.

$\beta_{Tot} = \bigotimes_{j=0}^{n+2}(\beta - \sum_{k=0}^{j}\alpha_k^0)_+ \bigotimes_{j=1}^{n}(\beta - \sum_{k=j+2}^{n+1}\alpha_k^j)_+$,
où α_k^0 est la courbe d'arrivée du $k_{ième}$ flux le plus prioritaire du serveur j, et α_k^j est la courbe d'arrivée de fk au serveur $n+1+j$.

Calcul des α_k^0 :
La figure 24 présente l'arrivée des trois flux les plus prioritaires du réseau accompagnés des différentes fonctions alpha qui leur sont associés au cours de leur traversée du réseau, jusqu'au serveur $n+2$.

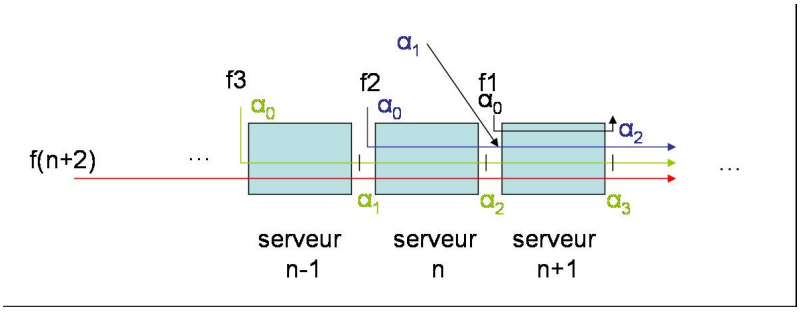

FIG. 23 – Représentation d'une partie de la première moitié du réseau étudié.

$\alpha_0^0(t) = \alpha(t)$,
$\alpha_1^0(t) = (\alpha \oslash \beta)_+ = \alpha(t+T) = \alpha(t+t'_0)$, (prouvé dans le livre [08] à la page 148)
$\alpha_2^0(t) = ((\alpha \oslash \beta)_+ \oslash (\beta - \alpha)_+)_+(t) = (\alpha_1^0 \oslash (\beta - \alpha_0^0)_+)_+(t)$ etc.
Et donc $\alpha_k^0(t) = \alpha_{k-1}^0 \oslash (\beta - \sum_{i=0}^{k-2}\alpha_i^0)(t),\ \forall k \in \{0, ..., n\}$.

Nous allons montrer par récurence que $\alpha_k^0(t) = \alpha(t + \sum_{i=0}^{k-2} t'_i)$.
On trouve directement que α_1^0 vérifie la formule. Supposons maintenant que cette expression est vraie à l'ordre $(k-1)$ et montrons qu'elle l'est alors à l'ordre k.
$\alpha_k^0 = (\alpha_{k-1}^0 \oslash (\beta - \sum_{i=0}^{k-2}\alpha_i^0)_+)_+$,
et $\beta - \sum_{i=0}^{k-2}\alpha_i^0 = (R - (k-1)\rho)(t - t'_{k-1}))_+$,
avec t'_{k-1} la date à laquelle la fonction $(\beta - \sum_{i=0}^{k-2}\alpha_i^0)_+$ s'annule. On trouve $t'_{k-1} = \frac{(k-1)\sigma + \sum_{i=0}^{k-2}\rho(\sum_{j=0}^{j-1}t'_i) + RT}{R-(k-1)\rho}$.

$$\alpha_k^0(t) = \sup_{u\geqslant 0}\{\alpha_{k-1}^0(t+u) - (\beta - \sum_{i=1}^{k-2}\alpha_i^0)_+(u)\}$$
$$= \sup_{0\leqslant u\leqslant t'_{k-1}}\{\sigma+\rho(t+\sum_{i=0}^{k-2}t'_i+u)\} \vee \sup_{u\geqslant t'_{k-1}}\{\sigma+\rho(t+\sum_{i=0}^{k-2}t'_i+u) - (R-(k-1)\rho)(u-t'_{k-1})\}$$
$$= \{\sigma+\rho(t+\sum_{i=0}^{k-2}t'_i+t_{k-1})\} \vee \sup_{u\geqslant t'_{k-1}}\{\sigma+\rho(t+\sum_{i=0}^{k-2}t'_i) - (R-k\rho)u + t'_{k-1}(R-(k-1)\rho)\}$$

Or $(R-k\rho) > 0$,
donc $\{\sigma+\rho(t+\sum_{i=0}^{k-2}t'_i+t_{k-1})\} \geqslant \sup_{u\geqslant t_{k-1}}\{\sigma+\rho(t+\sum_{i=0}^{k-2}t'_i) - (R-k\rho)u + t'_{k-1}(R-(k-1)\rho)\}$.
Finalement : $\alpha_k^0 = \{\sigma+\rho(t+\sum_{i=0}^{k-2}t'_i+t'_{k-1})\}$.

Calcul des α_k^j : Sur la figure ci-dessous nous avons représenté les premiers serveurs après le milieu du réseau. A partir de ce moment, les flux vont quitter le réseau un à un en commençant par le plus prioritaire. Nous avons associé les courbes des arrivées de ces flux en fonction de leur progression dans le réseau, de façon à montrer l'évolution de ces données.

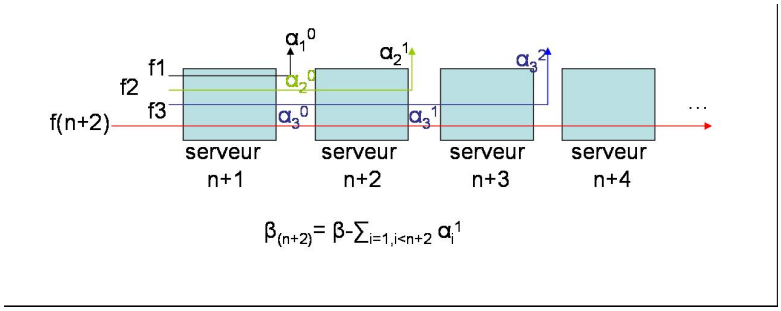

FIG. 24 – Représentation d'une partie de la deuxième moitié du réseau étudié.

La fonction α_k^j est la courbe des arrivées associée au flux f_k à l'entrée du serveur $(n+1-j)$. Comme le calcul de ces fonctions est très similaire à celui fait pour les α_k^0, nous présenterons uniquement le cas initial (i.e. $j=1$).
On a $\alpha_i^1 = (\alpha_i^0 \oslash (\beta - \sum_{\ell=2}^{i-1}\alpha_\ell^0)_+)_+$,
et $(\beta - \sum_{\ell=2}^{i-1}\alpha_\ell^0)_+(t) = (R-(i-2)\rho)(t-t_{i,1}))_+$,
avec $t_{i,1}$ la date à laquelle la fonction $(\beta - \sum_{\ell=1}^{i-2}\alpha_\ell^0)_+$ s'annule.
Alors $t_{i,1} = \frac{(i-2)\sigma+\sum_{\ell=2}^{i-1}\rho(T+\sum_{k=1}^{\ell-1}t'_k)+RT}{R-(i-2)\rho}$.

Ainsi,

$$\begin{aligned}
\alpha_i^1(t) &= \sup_{u\geqslant 0}\{\alpha_i^0(t+u) - (\beta - \sum_{\ell=2}^{i-1}\alpha_\ell^0)_+(u)\} \\
&= \sup_{0\leqslant u\leqslant t_{i,1}}\{\sigma+\rho(t+\sum_{\ell=0}^{i-1}t'_\ell+u)\} \vee \sup_{u\geqslant t_{i,1}}\{\sigma+\rho(t+\sum_{\ell=0}^{i-2}t'_\ell+u) - (R-(i-1)\rho)(u-t_{i,1})\} \\
&= \{\sigma+\rho(t+T+\sum_{\ell=1}^{i-1}t'_\ell+t_{i,1})\}
\end{aligned}$$

Donc $\alpha_i^j(t) = \alpha(t+\sum_{\ell=0}^{i-1}t'_\ell+\sum_{\ell=J}^{n+2}t_{i,\ell})$, où $t_{i,j} = \frac{(i-j)\sigma+\sum_{\ell=j+1}^{i-1}\rho(t+\sum_{k=0}^{i-1}t'_k+\sum_{k=1}^{j}t_{\ell,k})}{R-(i-j)\rho}$.

Il ne nous reste plus qu'à déterminer une expression de β_{Tot} est fonction des paramètres : R, T, ρ, σ :

$$\begin{aligned}
\beta_{Tot}(t) &= \bigotimes_{j=0}^{n+1}(\beta(t) - \sum_{\ell=0}^{j}\alpha(t+\sum_{k=0}^{\ell-1}t'_k))_+ \\
&\quad \bigotimes_{j=1}^{n}(\beta(t) - \sum_{\ell=j}^{n+2}\alpha(t+\sum_{k=0}^{\ell-1}t'_k+\sum_{k=1}^{j}t_{\ell,k}))_+, \\
\beta_{Tot}(t) &= (R-(n+1)\rho)(t-\sum_{j=0}^{n+2}m_j+\sum_{j=1}^{n}m'_j)_+.
\end{aligned}$$

Soit m_j la date à laquelle $(\beta(t)-\sum_{\ell=0}^{j}\alpha(t+T+\sum_{k=1}^{\ell-1}t'_k))_+$ s'annule et m'_j celle à laquelle l'autre fonction s'annule. Par un simple calcul (identique à celui fait pour déterminer les t'_k et t_k), on retrouve les formules énoncées dans le lemme. De plus, en appliquant le concaténation des pentes par coefficient directeur croissant, on obtient la formule de β_{Tot} énoncée dans le lemme.

Références

[01] Luca Bisti, Luciano Lenzini, Enzo Mingozzi, Giovani Stea, *Estimating the Worst Case Delay in FIFO Tandems Using Network Calculus*, Proceedings of Valuetools'2008, 2008

[02] Anne Bouillard, Laurent Jouhet, Eric Thierry, *Service curves in Network Calculus : dos and don'ts*, Rapport de collaboration scientifique avec l'ONERA, numéro 7094, Juin 2009

[03] Annne Bouillard, Laurent Jouhet, Eric Thierry, *Tight Performance Bounds in the Worst Case Analysis of Feed Forward Networks*, INFOCOM 2010

[04] Marc Boyer, Christian Fraboul, *Thightening the end to end delay bound for AFDX network calculus with rate latency FIFO servers using network calculus*, Proceedings of the 7th IEEE International Workshop on Factory Communication Systems Communication in Automation (WFCS'2008), 2008

[05] Cheng-Shang Chang, *Performance Garantees in communication networks*, TNCS, Springer-Verlag, 2000

[06] Rene L. Cruz, *A Calculus for Network Delay Part I : Network Elements in Isolation*, IEEE Transactions on Information Theory, pages : 114-131, 1991

[07] Rene L. Cruz *: A Calculus for Network Delay Part II :Network Analysis*, IEEE Transactions on Information Theory, pages : 132-14, 1991

[08] Jean Yves Le Boudec, Patrick Thiran *: Network Calculus : A theory of Deterministic Queuing System for the Internet*, LNCS 2050, Springer, 1998

[09] Jens B. Shmitt, Frank A. Zdarsky *: Delay Bounds under Arbitrary Multiplexing : When Network Calculus Leaves you in a Lurch...*, In INFOCOM 2008, 2008

[10] Lothar Thiele, Samarjit Chakraborty, Mattias Gries, Alexander Mexiaguine, Jonas Greutert *: Embedded Sorftware in Network Processors - Models and Algorithms*,Proceedings of EMSOFT'2001, 2001