Approches (m,k)-firm pour la gestion de la qualité de service temps réel

Ye-Qiong Song

Approches (m,k)-firm pour la gestion de la qualité de service temps réel

YeQiong SONG
LORIA - INPL
Campus Scientifique, B.P. 239
54506 Vandoeuvre-Lès-Nancy, France
song@loria.fr

Résumé

Cet article présente d'abord un état de l'art sur les principaux algorithmes d'ordonnancement développés pour la garantie temps réel (m,k)-firm, puis explique comment les approches (m,k)-firm peuvent être utiles pour une meilleure gestion de la qualité de service avec dégradation contrôlée (graceful degradation) dans les réseaux et systèmes temps réel. Un algorithme appelé (m,k)-WFQ est détaillé pour illustrer l'intérêt de l'approche (m,k)-firm dans l'ordonnancement des paquets de flux MPEG dans des réseaux. Le problème fondamental de la garantie déterministe de (m,k)-firm est également approfondi à travers l'élaboration de la condition suffisante d'ordonnançabilité de l'algorithme DBP.

1. Introduction

Aujourd'hui, de plus en plus d'applications temps réel sont déployées au-dessus de réseaux comme l'Internet. Ceci signifie que ces réseaux doivent fournir des garanties en termes de respect de contraintes temporelles sur les communications. L'augmentation constante de débit des réseaux qui composent l'Internet (Ethernet, réseaux mobiles sans fil, réseaux courants porteurs par exemple) n'apporte une solution que temporaire. En effet, le surplus de bande passante apportée par toute augmentation de débit est pratiquement immédiatement comblée par de nouvelles applications multimédias. La technique de réservation qui consiste à sur-dimensionner les ressources (over-provisioning) n'est, donc, pas une solution viable à l'avenir. Ce constat oblige donc à spécifier des techniques et méthodes pour garantir une Qualité de Service (QdS) temporelle sous des contraintes de ressources limitées. De plus, la gestion de la QdS temporelle dans l'Internet soulève un problème de passage à l'échelle. Pour y faire face, l'IETF recommande d'appliquer l'architecture « Diffserv » plutôt que « Intserv ». Mais le problème de « Diffserv » est que la garantie est vis à vis de classes de trafics et non pour une application. Une exigence de garantie déterministe de QdS pour une application dans une classe d'applications oblige à un dimensionnement selon la contrainte la plus stricte imposée aux applications considérées, conduisant de nouveau à un surdimensionnement de ressources. Notons que des solutions pour supporter des applications sous contraintes temps réel souples sur l'Internet ont été proposées [El-Gendy03]. Mais ces solutions n'apportent qu'une garantie temps réel probabiliste. Ceci peut ne pas convenir à certains types d'applications, notamment dans le cas du contrôle-commande ou du multimédia. En effet, ces applications peuvent tolérer des pertes de paquets (ou des paquets écartés à cause du retard dépassant l'échéance requise) en transmission sur les réseaux mais à condition que ces pertes soient distribuées selon un modèle spécifié de manière déterministe et non en observant une propriété sur la moyenne de leurs occurrences (par exemple, transmission de paquet de MPEG, de voix sur IP, …). Un problème typique est comment éviter trop de pertes consécutives des paquets en cas de congestion des réseaux. Il est clair que les politiques classiques de gestion de buffers telles que TD (Tail-Drop) conduit inévitablement à des pertes consécutives tandis que RED (Random Early Detection), en écartant aléatoirement des paquets dans une région de taille de file d'attente, essaie de résoudre ce problème mais sans donner aucune garantie.

Si l'on s'attache aux systèmes temps réel distribués, on est confronté au même problème de surdimensionnement. Classiquement, chercher la garantie absolue de QdS pour des applications sous contraintes temps réel dures revient à prendre en compte du pire cas ; or le système fonctionne en temps normal avec un cas moyen très éloigné du pire cas. Ce phénomène est encore plus accentué avec le déploiement de la méthode d'ordonnancement holistique comme démontré dans [Martin04]. Contrairement à des mécanismes de QdS dans l'Internet qui s'auto-adaptent à l'état du système grâce au mécanisme de contrôle d'admission et des mesures de QdS en-ligne (une sorte de « feedback »), un système temps réel classique conçu selon l'approche temps réel dur souffre de rigidité à cause des hypothèses strictes sur le modèle de tâches/messages. Ce qui peut rendre une solution

ordonnançable vulnérable face aux aléas (de charges, de ressources, de perturbations de l'environnement) Aussi, dans le domaine du contrôle-commande, il apparaît intéressant de s'orienter vers la notion de système adaptatif afin de supporter non seulement la variation de performances du support informatique (tolérance aux fautes), mais aussi l'évolution de l'application qui induit une variation de charges par rapport aux hypothèses du départ sur le modèle d'activation de tâches. En plus, l'utilisation des composants standards, comme par exemple un réseau Ethernet partagé avec d'autres applications à la place d'un bus de terrain, exige aussi l'implémentation des mécanismes de contrôle d'admission et d'ordonnancement de trafics en fonction de la mesure en-ligne de la QdS, c'est à dire avec « feedback ». Fournir des mécanismes de gestion de la QdS appropriés dans un système temps réel adaptatif (incertitude de charges et de ressources) reste encore un problème ouvert [ARTIST03].

Par ailleurs, que ce soit en QdS dans les réseaux ou en ordonnancement dans les systèmes temps réel, les travaux antérieurs négligent le fait que la plupart des applications soient capables, dans une limite à identifier, de tolérer et/ou s'adapter à la variation de performances du système. Des exemples typiques sont la transmission de vidéo et voix qui tolère la perte occasionnelle des paquets, des systèmes de contrôle-commande sur-échantillonnés qui peuvent non seulement tolérer des pertes des échantillons, mais en plus la loi de commande peut aussi compenser des pertes et retards grâce à l'emploi de boucles de contrôle fermées ayant comme entrée supplémentaire la QdS instantanée du système support. Par exemple, des travaux regroupés sous le nom de NCS « Networked Control Systems » visent à étudier la robustesse des lois de commande en fonction de variation de performance de l'architecture informatique support (calculateurs et réseaux) ou à concevoir des lois de commande robuste en prenant en compte la variation de la QdS du système support (en particulier le réseau) [Nilsson98], [Chow01], [Jumel03]. Il est donc plus optimal de concevoir des applications temps réel en prenant compte cette capacité « naturelle » de tolérance du non-respect des échéances. Dans ce cas, le modèle (m,k)-firm [Hamdaoui95] paraît convenable pour spécifier plus précisément les contraintes temps réel. Une contrainte (m,k)-firm est définie sur une tâche récurrente. Elle exige qu'au moins m parmi k invocations consécutives de la tâche doivent être exécutées par le système en respectant leur échéance, avec $m \leq k$ (le cas où $m = k$ est équivalent du cas de temps réel dur, que nous notons aussi par (k,k)-firm). Si l'on considère qu'une application peut accepter une dégradation de service jusqu'à m exécutions avant l'échéance parmi k demandes d'exécutions consécutives quelconques, un système peut alors conçu selon l'approche (m,k)-firm pour offrir des niveaux de QdS variés entre (k,k)-firm (cas normal) et (m,k)-firm (pire cas) avec autant de niveaux intermédiaires correspondant aux différentes valeurs possibles entre k et m. Ce qui résulte en un système avec la dégradation de la QdS contrôlée (Graceful degradation).

La garantie du respect des contraintes temps réel selon le modèle (m,k)-firm dans un système temps réel dynamique et dans un réseau à QdS nécessite des efforts de recherche dans deux directions : 1) pour la prise en compte explicite de cette nouvelle contrainte (m,k)-firm, des algorithmes d'ordonnancement classiques tels que EDF (Earliest Deadline First), FP (Fixed Priority), WFQ (Weighted Fair Queueing) doivent être étendus et des algorithmes nouveaux restent à développer ; 2) bien qu'en moyenne (m,k)-firm permette de diminuer le besoin de ressources par rapport au temps réel dur qui est équivalent à (k,k)-firm, il n'est pas toujours possible de réaliser ce gain lors que l'ordonnancement est non préemptif et une garantie déterministe de (m,k)-firm (appelé aussi par certains chercheurs (m,k)-hard) est exigée. Ceci à cause de la NP-complétude du problème. Deux pistes sont possibles : soit le développement de l'analyse d'ordonnançabilité vis à vis de l'algorithme d'ordonnancement proposé dans des cas particuliers mais représentent un intérêt pratique, soit étendre le modèle (m,k)-firm initial afin de permettre de réaliser ce gain.

L'objectif de ce papier est de donner un aperçu des algorithmes d'ordonnancement pour la garantie déterministe temps réel (m,k)-firm et les appliquer à la gestion de la QdS.

La gestion de la QdS est assurée par trois fonctions fondamentales: *ordonnancement*, *gestion de files d'attente* en cas de saturation et la *régulation de trafic*. Dans ce papier, nous nous intéressons principalement à l'application du modèle (m,k)-firm dans la fonction de l'ordonnancement.

Le reste de ce papier est organisé comme ce qui suit. La section 2 présente un état de l'art sur les travaux autour de (m,k)-firm. La section 3 décrit (m,k)-WFQ qui permet à un serveur WFQ (Weighted Fair Queueing) de prendre en compte plus efficacement des contraintes temporelles des flux multimédias temps réel. La section 4 présente une analyse d'ordonnançabilité de l'algorithme DBP (Distance Based Priority) non préemptif. Enfin, la section 5 conclut le papier et indique les perspectives.

2. Etat de l'art sur (m,k)-firm

2.1. Modèle général: MIQSS

Considérons un modèle d'accès multiple à une ressource partagée que nous allons appeler MIQSS (Multiple Input Queues Single Server) dans la suite de ce document. Nous cherchons à ordonnancer des demandes d'accès au serveur commun, tout en satisfaisant leurs

contraintes temporelles et en optimisant le taux d'utilisation du serveur. Dans notre contexte de systèmes distribués temps réel, ce serveur peut modéliser un processeur pour les demandes d'exécution des invocations de tâches ou un médium de transmission (bande passante) de paquets.

Afin que nos résultats puissent aussi être applicables à la transmission de paquets, seul le cas *non-préemptif* nous intéresse. Notons que ce cas est en général plus difficile à analyser que le cas préemptif.

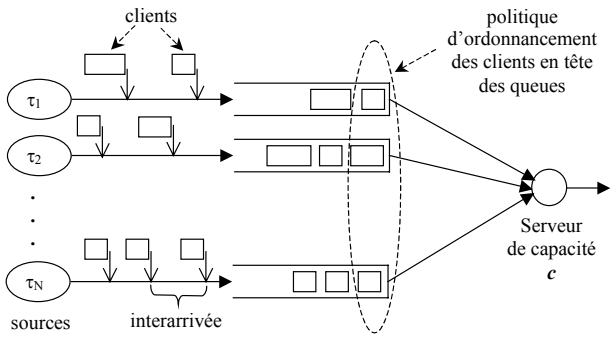

Figure 1. Modèle MIQSS

Une source τ_i est caractérisée par sa fonction de flux d'arrivée F_i. Dans cette étude, cette fonction peut être :

- **Périodique ou sporadique**: décrit par (C_i, T_i) dans le cas d'une date initiale quelconque d'arrivée du premier client et (r_i, C_i, T_i) dans le cas d'une date initiale fixe r_i, où C_i est le temps d'exécution d'un client et T_i la période d'inter-arrivée (ou d'inter-arrivée minimale dans le cas sporadique).
- **Périodique avec gigues** : (C_i, T_i, J_i) ou (r_i, C_i, T_i, J_i). Où J_i représente le déphasage maximum de l'instant d'une arrivée de client par rapport à la période.
- **(σ_i, ρ_i)-borné** : une courbe linéaire caractérisée par une taille de rafale σ_i et un débit moyen ρ_i qui majore la vraie fonction cumulative d'arrivée du travail [LeBoudec02], [Chang00]. La quantité du travail apportée par un client est définie par W_i avec notamment $C_i = W_i / c$ où c représente le débit du serveur.

Dans la suite, les contraintes temps réel sont toujours données par (D_i, m_i, k_i) où D_i est l'échéance relative à l'instant d'arrivée d'un client et (m_i, k_i) sont les deux paramètres de la contrainte (m_i, k_i)-firm.

2.2. Expression de contraintes (m,k)-firm et WHRT

Une source sous contrainte temps réel (m, k)-firm peut se trouver dans l'un des deux états : normal et échec transitoire (*dynamic failure*) [Hamdaoui95]. La connaissance de son état à l'instant t dépend de l'historique du traitement des k derniers clients générés par la source. Si l'on associe '*1*' à un client respectant son échéance et '*0*' à un client ratant son échéance, cet historique est alors entièrement décrit par une suite de k bits appelée une *k-séquence*. La Figure 2 donne un exemple de (2,3)-firm avec par convention le déplacement vers la gauche des bits.

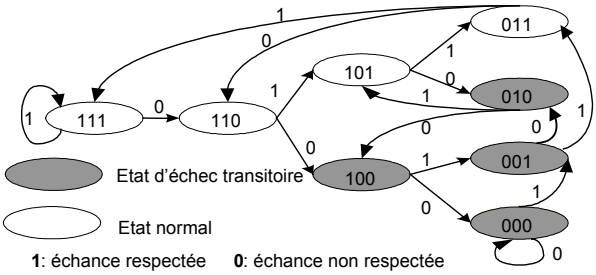

Figure 2. Diagramme d'état-transition d'une source avec (2,3)-firm

Dans un système qui peut être modélisé par MIQSS, on peut définir l'état du système à un instant t à partir des états des sources du même instant. Un système est dit en état d'échec transitoire si au moins une de ses sources est en échec transitoire (une sorte de ET logique entre les états de l'ensemble de sources).

Une source peut exprimer sa contrainte (m,k)-firm en spécifiant simplement la valeur des deux paramètres : m et k.

Afin de faciliter l'expression des contraintes du type (m,k)-firm mais avec plus de précision sur la répartition des m parmi les k clients consécutifs, [Bernat97] et [Bernat01] ont enrichi ce modèle (m,k)-firm en proposant trois autres formes qui correspondent à la complémentarité et la consécutivité:

- $\left(\overline{m}, \overline{k}\right)$-firm : au plus m clients avec *échéance non respectée* dans une fenêtre quelconque de k arrivées consécutives
- $\langle m, k \rangle$-firm : au moins m clients *consécutifs* avec échéance respectée dans une fenêtre quelconque de k arrivées consécutives
- $\langle \overline{m}, \overline{k} \rangle$-firm: au plus m clients *consécutifs* avec *échéance non respectée* dans une fenêtre quelconque de k arrivées consécutives

La notion de (m,k)-firm est alors généralisée et une source sous ces formes de contraintes est dite sous contrainte WHRT (Weakly-Hard Real-Time) [Bernat01]. Néanmoins il convient de remarquer que certaines de ces formes peuvent toujours être exprimées sous forme de (m,k)-firm :

- $\left(\overline{m}, \overline{k}\right)$-firm : équivalente à (k-m,k)-firm

- $\langle m, k \rangle$-firm : pas d'équivalence dans (m,k)-firm
- $\langle \overline{m}, \overline{k} \rangle = \langle \overline{m} \rangle$: en fait il est facile de constater qu'avec $\langle \overline{m}, \overline{k} \rangle$, on ne peut jamais avoir plus de *m* clients *consécutifs* avec *échéance non respectée* quelque soit la taille de *k* pourvu que $m < k$. De plus une source respectant (m,k)-firm inclut le cas particulier de $\langle \overline{k-m} \rangle$.

Notons que la k-séquence réalisée par un algorithme d'ordonnancement n'est pas nécessairement répétitive. On parle alors de *k-séquence dynamique*.

Un cas particulier d'expression de contrainte (m,k)-firm est la spécification d'une *k-séquence fixe* appelée un κ-pattern (ou (m,k)-pattern [Quan00]). Cette technique s'inspire du modèle de calcul imprécis [Chung90] où une tâche est composée d'une partie critique (mandatory) et d'une partie optionnelle. Le κ-pattern d'une source ayant une contrainte temporelle (m,k)-firm est défini par la succession de *k* éléments de l'alphabet {0, 1} où '0' indique une demande de traitement optionnelle et '1' une demande critique avec $\sum_{i=1}^{k} \pi_i = m$ où π_i est le $i^{ème}$ élément du κ-pattern pour $1 \le i \le k$.

En répétant continuellement le κ-pattern, on classe les demandes de traitement des clients d'un flux (ou une source) en deux catégories : optionnelle et critique. Il est facile de prouver qu'il suffit de traiter avec succès toutes les demandes critiques (les *m* « 1 ») pour satisfaire la contrainte (m,k)-firm (voir [Ramanathan99], Théorème 1). Notons que la réciproque n'est pas vraie car une garantie (m,k)-firm n'a pas objectif de produire une k-séquence fixe. Les demandes optionnelles peuvent être traitées quand le serveur n'est pas occupé ou rejetées si leur échéances ne peuvent pas être respectées par le serveur.

De ce fait, le $n^{ième}$ client (ou demande de traitement) d'un flux ayant la contrainte temporelle (m,k)-firm est considéré comme étant un client critique si et seulement s'il satisfait la relation suivante :

$$\pi_{(n\%k)} = 1 \qquad (1)$$

avec *n%k* le reste de la division de *n* modulo *k*.

L'utilisation d'un κ-pattern fixe a l'avantage de ramener le problème de l'analyse d'ordonnançabilité du système (m,k)-firm à celui de l'analyse d'ordonnançabilité classique. Par exemple quand tous les clients critiques sont ordonnancés sous la politique FP (fiexed priority) et les clients optionnels ont la priorité la moins élevée, l'analyse d'ordonnançabilité est donnée dans [Ramanathan99]. L'application de cette classification peut être utile dans le domaine du multimédia. En effet, ce concept peut être appliqué à un flux de paquets vidéos pour sélectionner les paquets critiques dans un GOP (Group of Pictures) en utilisant le standard de compression MPEG [Furht99]. Par exemple, un flux compressé utilisant la structure du GOP suivante [*IBBPBBPBB*], où les paquets I (Intra images) et P (Predicted images) sont plus importants que les paquets B (Bi-directional predicted/interpolated images), peut être considéré comme étant un flux ayant des contraintes temporelles de type (6,9)-firm et spécifié par le κ-pattern suivant $\{\pi_{i(1 \le i \le k)}\} = \{110110110\}$. Ce κ-pattern signifie qu'une partie des paquets de type B est déclarée comme optionnelle par la source de ce flux. Par exemple, le $226^{ème}$ paquet est considéré comme étant critique car $\pi_{(226\%9)} = \pi_1 = 1$ et le $228^{ème}$ paquet est considéré comme étant optionnel car $\pi_{(228\%9)} = \pi_3 = 0$.

Une fois la contrainte WHRT spécifiée, on peut alors passer à l'étape de recherche d'algorithmes d'ordonnancement pour que la contrainte soit respectée (de façon déterministe ou probabiliste).

2.3. Algorithmes d'ordonnancement pour (m,k)-firm

Il existe aujourd'hui principalement deux familles d'algorithmes qui prennent en compte (m,k)-firm: dynamique (par exemple DBP : Distance Based Priority) et statique (par exemple EFP : Enhanced Fixed Priority). Par algorithme dynamique nous voulons dire que la priorité affectée à chaque client est ajustée automatiquement en fonction de l'état du système (en particulier de la *k-séquence* des sources) à l'instant *t*. Tandis qu'une affectation statique de priorité est basée sur un paramètre fixe (taux *m/k* par exemple).

Un algorithme dynamique a l'avantage de permettre au système de s'adapter aux changements de situation (variation de flux, de capacité du serveur, …). Il convient à la gestion en-ligne de la QdS. Le problème est qu'il ne donne souvent qu'une garantie statistique (best-effort) de *m* sur *k*. C'est le cas de DBP et de la première version de DWCS (Dynamic Window Constrained Scheduling) [West99]. Une version améliorée de DWCS [West04] permet de donner une garantie déterministe de *m* sur *k* sous des conditions particulières (même C_i pour toutes les sources). *A contrario*, un algorithme statique permet une vérification hors-ligne du système et garantit de façon déterministe le respect de *m* sur *k* échéances dans le cas où le système ne violerait pas les hypothèses du pire cas.

Dans ce qui suit nous expliquons le principe de DBP, DWCS et EFP.

2.3.1. DBP (Distance Based Priority)

DBP [Hamdoui95] est la façon la plus directe pour la prise en compte de la contrainte (m,k)-firm. Pour une k-séquence donnée, DBP définit à chaque début du service d'un client la distance d'aller à un état d'échec transitoire comme le nombre consécutif de bits 0 à ajouter pour atteindre cet état. La priorité que DBP donne au client en tête de la queue correspondante à la k-séquence est égale à cette distance. Si la source se trouve déjà en état d'échec transitoire (i.e., moins de *m* 1 dans la k-séquence), la plus haute priorité 0 est affectée. Par exemple, pour une source sous contrainte (3,5)-firm, le client en tête de la queue est de priorité 2 si les 5 clients précédents forment une k-séquence (11011), il est de priorité 3 si les 5 clients précédents forment une k-séquence (10111).

Formellement, selon [Hamdaoui95] la priorité est évaluée comme suit. Nous notons par $s_j = \left(\delta^j_{i-k_j+1}, \cdots, \delta^j_{i-1}, \delta^j_i\right)$ la k-séquence de la source τ_j, par $l_j(n,s)$ la position (en comptant à partir de droite) de la n^{ième} échéance respectée (ou 1) dans s_j, la priorité du (i+1)^{ème} client de τ_j est donnée par :

$$P\_DBP^j_{i+1} = k_j - l_j\left(m_j, s_j\right) + 1 \qquad (2)$$

Notons que lorsqu'il y a moins de *n* 1 dans *s*, alors $l_j(n,s) = k_j + 1$, afin que la plus haute priorité (= 0) soit affectée.

La Figure 3 schématise comment DBP est utilisé pour l'affectation de priorité. Cette politique d'affectation dynamique de priorité peut être facilement et efficacement implémentée en matériel car l'historique de chaque source peut être stocké dans un registre de k_j bits.

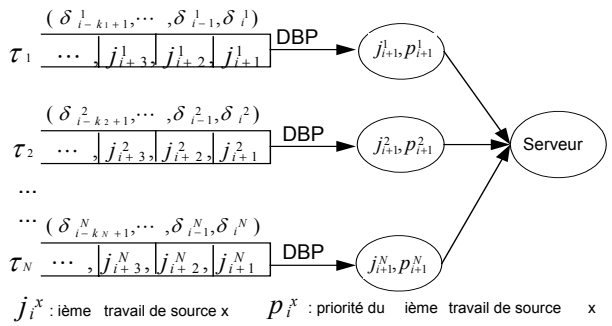

Figure 3. DBP pour l'affectation de priorité des clients en tête des queues

Le serveur choisit les clients présents en tête des queues selon leur priorité. Dans le cas d'égalité de priorité parmi les clients à choisir, EDF (Earliest Deadline First) est utilisée par défaut. Nous notons par DBP-EDF ce système.

2.3.2. DWCS (Dynamic Window Constrained Scheduling)

L'algorithme DWCS a été conçu dans [West99] pour maximiser l'utilisation de la bande passante du réseau en cas de surcharge pour des flux temps-réel tolérant aux pertes. Il se charge de garantir la contrainte de type $(\overline{2x}, \overline{x+y}) - firm$, c'est-à-dire, pas plus que $2x$ dépassements d'échéances dans n'importe quelle fenêtre de $x+y$ paquets consécutifs tout en ayant la capacité de partager la bande passante entre les paquets des flux en compétition en proportion de leurs échéances et tolérances aux pertes, avec x représente le nombre de paquets qui pourraient être perdus ou transmis en retard pour chaque fenêtre fixe de taille y paquets consécutifs. DWCS est développé pour être employé comme une alternative à EDF dans des conditions de surcharge, étant donné que les performances de EDF se dégradent sérieusement pour une charge supérieure à un.

Cet algorithme nécessite deux attributes par flux pour assurer l'ordonnancement des paquets :

- *L'échéance* D_i : elle est définie comme étant le temps maximum entre le service de deux paquets consécutifs au sein d'un même flux. Dans le cas d'un flux périodique, l'échéance D_i d'un flux τ_i est égale à sa période T_i.
- *La contrainte de fenêtre fixe:* elle est aussi appelée *facteur de tolérance aux pertes*. Elle est spécifiée par la valeur $W_i = x_i / y_i$ où x_i représente le nombre maximum de paquets perdus (ou transmis en retard) pour chaque fenêtre fixe de taille y_i paquets consécutifs.

Bien que DWCS s'intéresse à une fenêtre fixe, cette contrainte peut inclure le cas de fenêtre glissante du modèle (m,k)-firm. Dans [West04], il a été montré que cette contrainte (*x,y*) permet, au pire (quand les *x* paquets perdus se trouvent à la fin d'une fenêtre de taille *y* et les *x* autres paquets se perdent au début de la fenêtre suivante), de garantir le respect de $(\overline{2x}, \overline{x+y}) - firm$. Comme DBP, DWCS maintient l'information d'état par flux mais l'utilisation de cette information diffère significativement de DBP. En effet, DBP affecte la priorité relative à un flux en se basant sur l'historique des *k* derniers clients, alors que DWCS utilise la notion de la fenêtre fixe dans laquelle *x* et *y* changent de valeurs au cours du temps selon un algorithme que nous expliquons par la suite.

DWCS choisit les paquets à servir en fonction de leurs échéances ainsi que leurs facteurs de tolérance aux pertes. Dans [West99], l'affectation de priorité selon la

première version de DWCS (DWCS[1]) se résume en six règles et est présentée dans le tableau suivant.

1	Choisir le paquet avec la plus petite contrainte de fenêtre (*plus petit facteur de tolérance aux pertes*) $= \min_{i=1..N}(W_i = x_i / y_i)$ avec $y_i \neq 0$
2	S'il existe $1 \leq i, j \leq N \ / W_i = W_j \neq 0$, alors servir avec EDF $= \min_{n=1..N}(D_n)$
3	S'il existe $1 \leq i, j \leq N \ / W_i = W_j \neq 0$ et $D_i = D_j$, alors servir le paquet ayant le plus petit numérateur de la contrainte de fenêtre $= \min_{i=1..N}(x_i)$
4	Si $W_i = W_j = 0$ et $y_i = y_j = 0$, alors servir avec EDF $= \min_{n=1..N}(D_n)$
5	Si $W_i = 0$, alors servir le paquet ayant le plus grand dénominateur de la contrainte de fenêtre $= \max_{n=1..N}(y_n)$
6	Tous les autres cas sont traités par FIFO

Nous observons que si deux paquets ont les mêmes valeurs de facteurs de tolérance aux pertes et les mêmes valeurs d'échéances, alors les paquets sont servis selon l'ordre croissant des x_i où x_i / y_i représente la valeur *courante* du facteur de tolérance aux pertes pour tous les paquets du i[ème] flux. Ainsi, la priorité est affectée au paquet du flux ayant la contrainte de perte la plus étroite, afin d'éviter des pertes consécutives de paquets. Si les facteurs de tolérance ainsi que les dénominateurs y_i des deux paquets sont nuls, alors les paquets sont servis dans l'ordre croissant de leurs échéances ; Sinon, si les dénominateurs y_i sont non nuls, alors le paquet ayant la plus grande valeur du dénominateur de la contrainte de fenêtre sera affecté la plus haute priorité.

Chaque fois qu'un paquet du flux *i* est transmis, la contrainte de fenêtre du i[ème] flux est ajustée. De même, les contraintes de fenêtre des autres flux sont ajustées dans le cas où il existe des paquets de ces flux qui ont dépassé leurs échéances.

Pour les flux tolérant les pertes de paquets, les paquets ayant raté leurs échéances sont tout simplement rejetés. Pour les flux ne tolérant pas de pertes de paquets, les échéances servent à réduire le délai d'attente dans les files avant leur transmission. La valeur du facteur de tolérance sert dans ce cas à éviter un retard excessif des paquets de tel flux.

Les contraintes x et y sont ajustées au cours du temps en fonction des échéances si elles sont ratées ou non. Considérons un flux *i* ayant la contrainte de fenêtre originale $W_i = x_i / y_i$ à l'instant initial. Notons par $W_i' = x_i' / y_i'$ la contrainte de fenêtre courante. Si le paquet du flux *i* est transmis avant le dépassement de son échéance, les contraintes x_i' et y_i' sont ajustées de la façon suivante :

$$\begin{cases} \text{si } \left(y_i' > x_i'\right) \text{ alors } y_i' = y_i' - 1 \\ \text{si } \left(x_i' = y_i' = 0\right) \text{ alors } x_i' = x_i \ ; y_i' = y_i \end{cases}$$

Cependant, pour tous les paquets des autres flux en attente, si un paquet du flux $j / j \neq i$ rate son échéance, alors les contraintes sont ajustées selon la règle suivante:

$$\begin{cases} \text{Si } \left(x_j' > 0\right) \text{Alors} \\ \quad \left| \begin{array}{l} x_j' = x_j' - 1; \ y_j' = y_j' - 1; \\ \text{Si } \left(x_j' = y_j' = 0\right) \text{ Alors } x_j' = x_j; \ y_j' = y_j \end{array} \right. \\ \text{Sinon Si } \left(x_j' = 0\right) \text{ Alors} \\ \quad \left| \begin{array}{l} \text{Si} \left(x_j > 0\right) \text{ Alors } y_j' = y_j' + \left\lceil \dfrac{y_j + x_j}{x_j} \right\rceil \\ \text{Si} \left(x_j = 0\right) \text{ Alors } y_j' = y_j' + y_j \end{array} \right. \end{cases}$$

Donc à chaque fois qu'une échéance du flux *j* est ratée, le facteur de tolérance aux pertes de ce flux est ajusté de façon à lui donner plus d'importance dans le prochain tour de sélection de paquet. Cette approche évite le problème de famine en affectant des priorités plus élevées aux flux qui sont susceptibles de violer leurs contraintes de fenêtre initiales. Inversement, un paquet du flux *i* est servi avec respect de son échéance, conduit à la diminution du facteur de tolérance des autres flux réduisant ainsi sa priorité aux prochains tours.

Récemment, West et al. proposent dans [West04] la deuxième version de DWCS (DWCS[2]). La différence principale avec la première version est que les deux premières lignes du tableau sont inversées. Dans la deuxième version de DWCS, la première règle d'affectation de priorité est identique à EDF, i.e. le paquet ayant la plus petite échéance est le plus prioritaire. La deuxième règle dans DWCS[2] fait recours à une comparaison des contraintes de fenêtre lorsque les échéances sont égales. West et al. expliquent que le changement de l'ordre des règles est dû à l'optimalité de EDF dans des conditions de charge normale pour respecter les échéances et par conséquent les contraintes de fenêtre. Cependant, l'algorithme DWCS[1] reste toujours plus performant que EDF dans des conditions de surcharge où il est impossible de respecter toutes les échéances.

Dans [West04], les auteurs étudient les caractéristiques temporelles de DWCS[2] et montre analytiquement que, dans le cas où il existe un ordonnancement faisable pour un ensemble de flux périodiques, les délais des flux en service sont toujours

bornés même en situation de surcharge. En effet, il a été montré que le délai garanti à chaque flux est indépendant des autres flux en service même en situation de surcharge. De plus, les résultats de simulation montrent que DWCS et DBP on des performances similaires en termes de nombre d'échéances ratées et de violation de la contrainte de fenêtre. Enfin, une implémentation sur Linux de DWCS est téléchargeable à partir du site de l'auteur.

2.3.3. EFP (Enhanced Fixed Priority)

EFP est proposé dans [Hamdaoui97], [Ramanathan99]. Pour prendre en compte la contrainte (m,k)-firm, il suffit que chaque source définisse un κ-pattern et marque parmi ses k clients consécutifs m clients *critiques* et k-m clients *optionnels*. En faisant ainsi le serveur pourra rejeter un client optionnel en cas de surcharge (c'est à dire au cas où son échéance ne peut plus être respectée par le serveur). Tous les clients critiques peuvent être ordonnancés par un algorithme à priorité fixe tel que RM (Rate Monotonic). Les clients optionnels sont servis avec la priorité la plus basse selon la politique FIFO. Le problème revient donc à définir un κ-pattern. Pour commencer le marquage, le premier client de chaque source est marqué critique par défaut. Pour une source τ_i, le marquage des clients critiques et optionnels selon sa contrainte (m_i,k_i)-firm est alors entièrement donné par l'équation suivante.

Le $n^{\text{ième}}$ client ($n = 0, 1, \ldots$) est marqué critique si n vérifie : $n = \left\lfloor \left\lceil \frac{n \times m}{k} \right\rceil \times \frac{k}{m} \right\rfloor$

Ce qui donne comme κ-pattern suivant (pour $i=1, 2, \ldots, k$) :

$$\pi_i = \begin{cases} 1 & \text{si } \; i = \left\lfloor \left\lceil \frac{i \times m}{k} \right\rceil \times \frac{k}{m} \right\rfloor \\ 0 & \text{sinon} \end{cases} \qquad (3)$$

Le marquage ne dépend que du rapport m_i/k_i. Une condition suffisante est donnée dans [Ramanathan99] pour la garantie déterministe de contrainte (m_i,k_i)-firm.

Cet algorithme souffre néanmoins trois problèmes:

- Le premier client de chaque source est marqué critique par défaut. Ce qui force artificiellement le système de se retrouver dans un « pire cas ».
- L'équation 3 distribue régulièrement les m clients critiques parmi les k arrivées consécutives. Ce qui peut ne pas être optimal dans certaines situations.
- La technique de marquage ne dépend que du rapport m_i/k_i, mais pas de C_i et T_i. Deux sources ayant des C_i et T_i très différents mais avec la même contrainte (m,k)-firm relèveront du même κ-pattern et donc se verront leur clients critiques distribués de la même façon. Le fait de ne pouvoir les distinguer peut conduire à une situation non optimale.

Partant de l'idée qu'une partition judicieuse et globale des clients critiques de toutes les sources devrait donner une meilleure ordonnançabilité, [Quan00] a amélioré l'algorithme présenté dans [Hamdaoui97, Ramanathan99]. Il a d'abord prouvé que trouver une partition optimale est NP-difficile. Puis, il donne une heuristique pour optimiser la répartition de m_i clients critiques parmi k_i clients consécutifs en prenant en compte les relations entre les sources.

3. (m,k)-WFQ pour une meilleure gestion de la QdS temps réel

L'ordonnanceur WFQ (Weighted Fair Queueing) [Parekh93] est déployé dans les commutateurs et routeurs du réseau Internet pour fournir de la QdS grâce à ses propriétés de garantie de bande passante et de délai borné pour des flux (σ,ρ)-bornés. L'algorithme (m,k)-WFQ consiste à intégrer les contraintes temporelles (m,k)-firm au processus d'ordonnancement de WFQ. Nous faisons d'abord un rappel du principe de WFQ afin d'expliquer ensuite l'apport de (m,k)-WFQ.

WFQ garantit à chaque flux servi la proportion de la bande passante réservée selon son coefficient de partage Φ_i. Chaque paquet de messages est estampillé par un tag appelé temps virtuel de départ. Le serveur sélectionne toujours le paquet dont le temps virtuel de départ est le premier à partir de l'instant de sélection. Dans WFQ le temps virtuel de départ est défini par :

$$F_i^k = \max\left\{F_i^{k-1}, V(t)\right\} + \frac{L_i^k}{\Phi_i} \qquad (4)$$

avec

- F_i^k : temps virtuel de départ du $k^{ième}$ paquet du $i^{ème}$ flux,
- V(t) : le temps virtuel quand le $k^{ième}$ paquet arrive,
- Φ_i : le coefficient de partage du $i^{ème}$ flux,
- L_i^k : la taille du $k^{ième}$ paquet du $i^{ème}$ flux,
- $\max\left\{F_i^{k-1}, V(t)\right\}$: le temps virtuel du début de service du $k^{ième}$ paquet.

Avec WFQ, il est montré dans [LeBoudec02] que pour un flux τ_i de type (σ_i,ρ_i)-borné et ayant un débit moyen réservé $g_i \geq \rho_i$, le délai garanti par WFQ à ce flux est borné par :

$$D_{i,\max} = \frac{\sigma_i}{g_i} + \frac{L_{\max}}{c} \qquad (5)$$

où L_{max} est la taille maximale du paquet parmi tous les paquets dans tous les flux et c la capacité de traitement du serveur.

Nous rappelons qu'un flux est dit (σ,ρ)-borné si sa fonction cumulative d'arrivée $R(t)$ vérifie la relation $\forall 0 \le s \le t,\ R(t) - R(s) \le \sigma + \rho(t-s)$ avec σ la taille maximale de rafale et ρ le débit moyen à long terme.

La borne fournie par WFQ sur le temps de réponse d'une source de flux est étroitement liée au coefficient de partage de la bande passante ρ_i et à la taille de la rafale σ_i. Pour avoir un délai d'attente court, un flux doit réserver une large bande passante. Pour un flux de faible débit moyen et ayant une grande rafale ceci peut conduire à une mauvaise utilisation de la bande passante. Ce problème peut être résolu avec la politique WFQ priorisé proposé dans [Wang02] mais la notion de (m,k)-firm n'est pas prise en compte.

Nous avons proposé dans [Koubâa04a], [Koubâa04b] une approche appelée (m,k)-WFQ. Pour que l'ordonnanceur WFQ puisse prendre en compte les contraintes temporelle (m,k)-firm, nous exprimons la contrainte par un κ-pattern, donc la source marque m paquets critiques parmi tous les k paquets consécutifs et les autres étant optionnels. L'ordonnanceur (m,k)-WFQ estampille ensuite le paquet par son temps virtuel de départ décrit par l'équation 4. L'algorithme est décrit dans la Figure 4. Le processus de service est activé quand au moins un paquet existe dans la file d'attente du système. Le serveur sélectionne le paquet ayant le plus petit temps virtuel de départ parmi tous les paquets critiques présents en tête de files. Si aucun paquet critique existe, le choix sera fait parmi les paquets optionnels. Puis, si le paquet sélectionné est critique, il est exécuté (ou transmis) directement par le serveur, tandis que si le paquet est optionnel, l'ordonnanceur vérifie avant son exécution si ce paquet pouvait éventuellement satisfaire son échéance. Si l'échéance souhaitée ne peut être garantie après l'exécution, le serveur rejette le paquet et refait une nouvelle sélection, sinon, il l'envoie.

L'avantage de l'algorithme proposé est qu'il permet de garantir une bande passante à un flux tout en intégrant les propriétés temporelles dans le processus d'ordonnancement ce qui revient à gérer les flux plus efficacement. En effet, le rejet des paquets optionnels qui ne satisfont pas leurs échéances permet au serveur de donner la main plus rapidement aux paquets critiques en attente. Cette perte ne dégrade pas les performances des flux servis tant que leurs contraintes (m_i,k_i)-firm sont satisfaites. Ainsi, (m,k)-WFQ diminue forcément les bornes sur les temps de réponse des flux temps réel par rapport à WFQ standard. Dans ce qui suit nous montrons quantitativement cette amélioration par simulation d'un exemple.

```
Entrées
Flux τi = {(Période ou Débit), Echéance Désirée, (mi,ki), (Gigue ou Rafale), Taille de Paquet)

Affectation de priorité
A l'arrivée du a^ième du flux [i] {
  si (π(a%ki)=1) alors {
      Marquer le paquet comme critique;
      }
  sinon {
      Marquer le paquet comme optionnel;
      }
  Calculer le temps virtuel de départ F_i^k ;
  Estampiller le paquet avec F_i^k ;
  Mettre le paquet dans sa file d'attent;

Discipline de Service
Serveur = libre;

   si (serveur!= occupé) {
     Choisir le paquet dont F_i^k plus petit
      si (paquet est critique) {
         Envoyer le paquet;
         Serveur = occupé;
         }

            si (l'échéance serait ratée){
                 Rejet du paquet;
                 Serveur = libre;
                 }

                 Envoyer le paquet;
                 Serveur = occupé;
                 }
         }
      }
   si (serveur== occupé) {
    attendre jusqu'à tx totale du paquet;
    Serveur= libre;
  }
```

Figure 4. Algorithme (m,k)-WFQ

Considérons un réseau constitué de trois sources de trafic. Ces trois sources partagent un lien de 10 Mbit/s selon leurs coefficients de réservation. Dans cette simulation, on considère une taille fixe à tous les paquets des trois flux de 8 *Kbits*. Le Tableau 1 récapitule les paramètres de simulation pour chacun des flux.

Le marquage de paquets en critiques et optionnels est spécifié par un κ-pattern fixe pour chaque source.

La première source génère un flux de voix selon le modèle de trafic ON/OFF. Les périodes d'activité ON et de silence OFF sont exponentiellement distribuées avec les moyennes $1/\mu_{ON} = 500ms$ et $1/\mu_{OFF} = 755ms$ avec une période de génération de paquets dans la période d'activité de *50 ms*. Donc, le débit moyen du flux est de 64 Kb/s. Les contraintes temporelles sont de type (4,5) et l'échéance souhaitée d'un paquet est fixée à 10 ms. Le κ-pattern fixe le profil de la séquence comme : `11011 11011 11011 …`

	(m,k)	Débit	Trafic	κ-pattern	Echéance
Voix	(4,5)	64 kb/s	ON/OFF (500/755/50)ms	*11011*	10 ms
Vidéo	(3,5)	2Mb/s	Périodique avec gigue ~2Mb/s	*10110*	4 ms
FTP	(0,1)	7,936 Mb/s	Périodique avec gigue ~7.936 Mb/s	*0*	Infinie

Tableau 1. Configuration simulée

La deuxième source est une source CBR (Constant Bit Rate) périodique avec gigue (95% de T_i-C_i) qui génère un flux vidéo de 2 Mbit/s. L'échéance des paquets est fixée à 4 ms avec une garantie de type (3,5). Le κ-pattern fixe le profil de la séquence comme : `10110 10110 10110 …`

La troisième source est un agrégat de flux FTP, que nous supposons périodique avec gigue (95% de T_i-C_i) et qui consomme le reste de la bande passante ayant donc un débit de 7,936 Mb/s. Un flux FTP est vulnérable à la perte de paquets et ce trafic fonctionne en mode Best-Effort. Donc, il ne possède pas de propriétés temporelles strictes comme dans le cas des deux sources temps-réel : Voix et Vidéo. Par conséquent, nous fixons une garantie de type (0,1) pour le flux FTP et une échéance infinie afin d'éviter tout rejet de paquets FTP optionnels.

Le tableau 2 montre les bornes mesurées sur le temps de réponse des paquets pour chacun des flux et ce pour le serveur (m,k)-WFQ, le serveur WFQ, le serveur (m,k)-FIFO et le serveur FIFO.

	(m,k)-WFQ	WFQ	(m,k)-FIFO	FIFO
Voix	9,769 (*taux de rejet = 6,8%*)	2428,031	20,529	48,031
Vidéo	3,999 (*taux de rejet = 5,5%*)	55,391	21,086	49,031
FTP	9,696	36,562	21,442	49,083

Tableau 2. Bornes sur les temps de réponse (ms)

Les cas du serveur FIFO et (m,k)-FIFO sont simulés pour que l'on puisse les comparer avec le cas du serveur (m,k)-WFQ. Un serveur (m,k)-FIFO est simplement un serveur FIFO avec le rejet des paquets optionnels ayant leur échéances ratées.

Comme prévu, (m,k)-WFQ fournit une garantie plus étroite sur le délai pour les flux temps-réel. Dans ce scénario, on peut remarquer que le délai maximal garanti par WFQ au trafic de la voix est assez grand. Ce résultat découle de deux facteurs majeurs (*cf. équation 5*) : le faible taux de bande passante réservée (64 Kbit/s) et la taille importante de la rafale. L'algorithme (m,k)-WFQ permet de réduire considérablement les bornes sur les temps de réponse en sacrifiant quelques paquets optionnels selon les contraintes temporelles (m,k)-firm de chaque flux. Le rejet des paquets optionnels ne satisfaisant pas leurs échéances améliore nettement le délai des paquets critiques. En comparant (m,k)-WFQ avec la politique (m,k)-FIFO, on peut aussi constater que (m,k)-WFQ conserve la bonne propriété de WFQ en terme de distinction des flux (garantie par flux).

Pour fournir la garantie déterministe de (m,k)-firm dans (m,k)-WFQ, nous donnons la borne sur le temps de réponse de (m,k)-WFQ. L'évaluation de cette borne n'est pas triviale essentiellement à cause de la difficulté de déterminer la part de paquets optionnels que le serveur a effectivement servi. La Figure 5 montre le modèle en « network calculus » qui a permis le calcul de cette borne.

Le calcul de la borne sur le temps de réponse utilise le formalisme du Network Calculus [LeBoudec02]. Dans [Koubâa04a] nous avons intégré les contraintes (m,k)-firm dans le formalisme du Network Calculus en introduisant la notion du (m,k)-

filtre qui permet de filtrer tous les paquets optionnels et fournir en sortie seulement les paquets critiques. La Figure 5 montre la technique pour modéliser le *flux effectif* qui devra être servi par un serveur, garantissant un débit fixé tel que celui de WFQ. Le flux effectif contient tous les paquets critiques et le nombre maximum de paquets optionnels qui pourront être servis par l'ordonnanceur. Les paquets optionnels servis sont ceux qui ne ratent pas leurs échéances. Ce flux effectif est utilisé pour le calcul de la borne sur le délai garanti par (m,k)-WFQ.

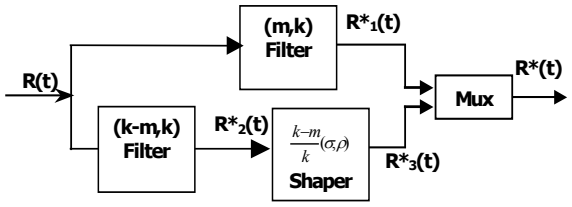

Figure 5. Modèle de Network calculus

Le délai maximal garanti pour une source (σ,ρ)-borné respectant une contrainte temporelle (m,k)-firm avec un taux de partage de bande passante $g \geq \rho$ et servi par un ordonnanceur (m,k)-WFQ est :

$$D_{\max}^{*} = \lambda_{m,k} \cdot \frac{\sigma}{g} + \lambda_{k-m,k} \cdot \frac{e}{g} + \frac{L_{\max}}{c} \qquad (6)$$

Avec $e \leq \sigma$ la taille maximale de rafale des paquets optionnels qui pourraient être transmis par l'ordonnanceur. $\lambda_{m,k}$ désigne le taux de bits critiques du flux et $\lambda_{k-m,k}$ le taux de bits optionnels du flux. Dans le cas où la taille du paquet est constante $\lambda_{m,k} = \frac{m}{k}$. Si aucun paquet optionnel n'est servi, $D_{\min}^{*} = \lambda_{m,k} \cdot \frac{\sigma}{g} + \frac{L_{\max}}{c}$ est la plus petite borne sur le délai. Pour garantir un délai entre D_{min}* et D_{max}*, on peut ajuster l'échéance maximale D_{op} qui détermine $e = gD_{op}$.

L'algorithme (m,k)-WFQ peut être étendu et intégré dans Intserv et le réseau ATM. L'idée de base est que chaque source voulant profiter d'une garantie avec dégradation adaptée doit marquer ses paquets en tant que optionnel ou critique selon sa contrainte (m,k)-firm et son κ-pattern associé. L'ordonnanceur WFQ qui garantit le débit dans le cadre du service garanti, doit tenir compte de cette classification. Les paquets optionnels dont l'échéance ne peut être respectée sont rejetés. (m,k)-WFQ permet alors de garantir des bornes sur le délai plus précises et d'une manière plus flexible. Pour une source ayant un trafic défini par le TSPEC (M,p,b,r) de Intserv et d'ATM avec *M* la taille maximale d'un paquet, *p* le débit crête, *b* la taille maximale de la rafale autorisée et *r* le débit moyen à long terme associé à la contrainte (m,k)-firm et autorisant un délai maximal pour les paquets optionnels égal à D_{op}, le délai maximal D_{max} a été obtenu dans [Koubâa04b] de façon similaire à l'obtention de l'équation 6.

4. Garantie déterministe et condition suffisante de DBP

On vient de voir que beaucoup d'algorithmes d'ordonnancement ont été proposés pour fournir une garantie en moyenne (best-effort) et déterministe du temps réel (m,k)-firm. S'il est vrai que par rapport à la garantie déterministe du temps réel dur, le fait de ne plus viser que garantir en moyenne *m* échéances parmi les *k* instances consécutives d'une tâche résulte en moins de demande de ressources en moyenne, il n'est pas évident que cet avantage est toujours conservé lorsqu'on cherche une garantie déterministe de (m,k)-firm. Cette question est fondamentale pour savoir si un algorithme d'ordonnancement pour (m,k)-firm peut apporter des avantages par rapport à un algorithme connu (EDF, FP, …) pour le temps réel dur avec garantie déterministe. Le point clé pour répondre à cette question est la recherche de conditions suffisantes d'ordonnançabilité. Un ensemble de sources $\tau = (\tau_1, \tau_2, \ldots, \tau_N)$ (dans le modèle MIQSS) ordonnançable respecte alors la contrainte (m,k)-firm de façon déterministe car l'analyse d'ordonnançabilité est réalisée dans le pire cas.

Le cas de (m,k)-WFQ donne relativement simplement cette garantie déterministe grâce à WFQ qui transforme en fait un serveur partagé en *N* serveurs dédiés à *N* sources, avec comme facteur d'interférence la longueur maximale d'un paquet L_{max}. L'obtention d'une condition suffisante dans un modèle MIQSS avec non préemption est en général un problème difficile.

Dans ce paragraphe, nous donnons d'abord un état de l'art sur ce problème de recherche de conditions suffisantes pour l'ordonnancement non préemptif, puis une condition suffisante pour la garantie déterministe du temps réel (m,k)-firm avec l'ordonnancement NP-DBP-EDF (Non Preemptive - Distance Based Priority - Earliest Deadline First) [Li03], [Li04].

4.1. Etat de l'art sur les conditions suffisantes

Nous commençons par nous intéresser à la condition suffisante pour la garantie déterministe (k,k)-firm (i.e. temps réel dur). Pour un ensemble de sources périodiques ou sporadiques $\tau = (\tau_1, \tau_2, \ldots, \tau_N)$

avec $\tau_i = \{T_i, C_i, D_i\}$ et des dates initiales quelconques, [Jeffay91] a donné un ensemble de conditions suffisantes et nécessaires d'ordonnançabilité sous EDF non préemptif (noté par NP-EDF : Non-Preemptive EDF). Dans la suite de ce paragraphe nous supposons que le temps est discrétisé et indexé par les entiers. Nous supposons également que l'échéance est égale à la période (ou à l'intervalle d'interarrivée minimal s'il s'agit du cas sporadique).

Théorème de [Jeffay91] :

Considérons un ensemble de N sources périodiques ou sporadiques $\tau = (\tau_1, \tau_2, \ldots, \tau_N)$ avec $\tau_i = \{T_i, C_i, D_i\}$ classées dans l'ordre non-décroissant des périodes (i.e. pour deux sources τ_i, τ_j, si $i < j$, alors $T_i \le T_j$) et $D_i = T_i$. Si τ est ordonnançable, on a :

C1: $$\sum_{i=1}^{N} \frac{C_i}{T_i} \le 1$$

C2: $\forall i,\ 1 < i \le N;\ \forall L,\ T_1 < L < T_i$:

$$L \ge C_i + \sum_{j=1}^{i-1} \left\lfloor \frac{L-1}{T_j} \right\rfloor C_j$$

Si τ satisfait les conditions C1 et C2 ci-dessus, alors NP-EDF peut ordonnancer n'importe quel ensemble concret généré à partir de τ. C'est à dire τ_i avec une date initiale r_i.

Le sens de C1 est clair. C'est la charge globale normalisée qui ne doit jamais dépasser 1. Une autre interprétation de C1 peut être que pour un intervalle de temps quelconque, la demande de traitement est toujours inférieure à la longueur de l'intervalle. C2 décrit une répartition extrême des flux d'arrivée : le client C_i occupe le serveur et tous les autres arrivent juste après une unité de temps (temps discret). Le serveur doit alors être capable de terminer le traitement de C_i, ainsi que le traitement des autres arrivées (représentées par le deuxième terme dans C2) sans dépasser une échéance.

Pour un ensemble τ dans le modèle MIQSS on peut utiliser ce théorème pour dimensionner la capacité de traitement du serveur c ($C_i = W_i/c$). Dans [Li03] un algorithme est développé pour trouver le c minimal.

En ce qui concerne la garantie déterministe (m,k)-firm dans le modèle MIQSS, si l'on considère (m,k)-WFQ comme un cas particulier de MIQSS et DWCS [West04] comme étant trop restreint, un seul autre résultat proposé par [Ramanathan99] existe pour le cas de κ-pattern fixe selon l'équation 3 que nous instancions ici dans le modèle MIQSS pour prendre en compte les sources multiples.

Pour une source $\tau_i = \{T_i, C_i, D_i, m_i, k_i\}$ le κ-pattern correspondant est une suite binaire de k_i bits $\Pi_i = \{\pi_{i1}, \pi_{i2} \ldots \pi_{ik_i}\}$, qui satisfait : *(1)* le $n^{ème}$ client est critique si $\pi_{1(n\%ki)} = 1$ et optionnel si $\pi_{1(n\%ki)} = 0$; *(2)* $\sum_{j=1}^{k_i} \pi_{ij} = m_i$.

Le κ-pattern proposé dans [Ramanathan99] est donné par :

$$\pi_{ij} = \begin{cases} 1 & \text{Si } j = \left\lfloor \left\lceil \frac{j \times m_i}{k_i} \right\rceil \times \frac{k_i}{m_i} \right\rfloor \\ 0 & \text{Sinon} \end{cases} \qquad j = 1,2,\cdots k_i \tag{7}$$

Les demandes de traitement critiques sont ordonnancées selon RM (Rate Monotonic). La condition suffisante est donnée par le théorème suivant.

Théorème de [Ramanathan99] :

Considérons un ensemble de N sources périodiques ou sporadiques $\tau = (\tau_1, \tau_2, \ldots, \tau_N)$ avec $\tau_i = \{T_i, C_i, D_i, m_i, k_i\}$ classées dans l'ordre non-décroissant des périodes (i.e. pour deux sources τ_i, τ_j, si $i < j$, alors $T_i \le T_j$) et $D_i = T_i$. Définissons les termes ci-dessous :

$$R_{ij} = \left\{ \left\lfloor l \cdot \frac{k_i}{m_i} \right\rfloor T_j : \left\lfloor l \cdot \frac{k_i}{m_i} \right\rfloor T_j < T_i, l \in Z_+ \right\}$$

$$R_i = \bigcup_{j=1}^{i-1} R_{ij}$$

$$n_j(t) = \left\lceil \frac{m_j}{k_j} \left\lceil \frac{t}{T_j} \right\rceil \right\rceil$$

$$W_i(t) = C_i + \sum_{j=1}^{i-1} n_j(t) \cdot C_j$$

Si $min_{t \in R_i} W_i(t)/t \le 1$ pour tout $1 \le i \le N$, alors la politique RM respecte de façon déterministe toutes les contraintes (m_i,k_i)-firm.

Dans la pratique pour un ensemble de source $\tau = (\tau_1, \tau_2, \ldots, \tau_N)$ avec dates initiales quelconques, trouver la capacité du serveur c minimale requise pour la garantie déterministe selon ce théorème est NP-difficile [Quan00].

Dans [Quan00] des algorithmes heuristiques sont proposés. Afin de minimiser la charge instantanée dans le pire cas (qui permet de diminuer la demande en c), [Quan00] propose de répartir plus uniformément les m_i parmi les k_i en faisant la rotation des m_i selon l'équation suivante.

$$\pi_{ij} = \begin{cases} 1 & \text{si } j - s_i = \left\lfloor \left\lceil \frac{((j-1)-s_i)\times m_i}{k_i} \right\rceil \times \frac{k_i}{m_i} \right\rfloor + 1 \\ 0 & \text{sinon} \end{cases} \qquad j = 1,2,\cdots k_i \tag{8}$$

où s_i est le nombre de périodes obtenues par le décalage circulaire vers la droite.

Un algorithme heuristique choisit une valeur de s_i provoquant ainsi moins d'interférence de demandes d'exécution par rapport à l'algorithme de [Ramanathan99]. Ce principe de rotation ne change pas de κ-pattern vis à vis d'une source mais change simplement la répartition dans l'axe du temps des demandes d'exécution critiques de N sources. En réalité, cette rotation veut introduire une sorte de κ-pattern dynamique. De ce point de vue DBP le fait plus facilement par l'affectation de priorité en-ligne.

4.2. Condition suffisante pour NP-DBP-EDF

Par rapport à NP-EDF dans [Jeffay91], NP-DBP-EDF introduit une variable supplémentaire qui est la valeur de DBP à l'instant t. Pour un client de la source τ_i sa priorité DBP est calculée selon l'équation 2 et on la note par $DBP_j(t)$. A un instant t, l'ensemble des clients peut être classé en trois classes suivantes :

1) Le client en cours de service dans le serveur
2) Les clients en attente avec $DBP_j(t) = 1$, i.e., ces clients doivent être exécutés par le serveur et terminer leur service avant leurs échéances respectives, sinon la garantie (m,k)-firm sera violée

2+*i*) Les clients en attente avec $DBP_j(t) = i$ (i>1), i.e., un tel client sera exécuté si le serveur est disponible et si l'exécution peut terminer avant son échéance, sinon il sera écarté par le serveur et le prochain client de la source aura sa priorité augmentée : $DBP_j(t+T_j) = DBP_j(t)-1$

Nous rappelons qu'en cas d'égalité de priorité DBP, EDF est utilisé.

La condition suffisante est donnée par le théorème suivant (cf. [Li03] pour la preuve).

Théorème de [Li03] :

Considérons un ensemble de N sources périodiques ou sporadiques $\tau = (\tau_1, \tau_2, \ldots, \tau_N)$ avec $\tau_i = \{T_i, C_i, D_i, m_i, k_i\}$ classées dans l'ordre non-décroissant des périodes (i.e. pour deux sources τ_i, τ_j, si $i < j$, alors $T_i \le T_j$) et $D_i = T_i$. Si τ satisfait les conditions C1 et C2 suivantes durant un intervalle de temps L quelconque, NP-DBP-EDF peut alors ordonnancer n'importe quel ensemble concret τ' généré par τ. C'est à dire qu'il n'y aura aucune violation de contrainte (m_i,k_i)-firm pour $i = 1, 2, \ldots, N$.

C1:
$$\sum_{i \in U} \left(\left\lfloor \frac{L}{k_i T_i} \right\rfloor m_i + Min\left(\left\lfloor \frac{L - k_i T_i \left\lfloor \frac{L}{k_i T_i} \right\rfloor}{T_i} \right\rfloor, m_i \right) \right) C_i +$$

$$\sum_{j \in \tau - U} \left(\left\lfloor \frac{L-1-(DBP_j(t)-2)T_j}{k_j T_j} \right\rfloor m_j + Min\left(\left\lfloor \frac{\left(L-1-(DBP_j(t)-2)T_j\right) - k_j T_j \left\lfloor \frac{L-1-(DBP_j(t)-2)T_j}{k_j T_j} \right\rfloor}{T_j} \right\rfloor, m_j \right) \right) C_j \le L$$

C2:
$$\forall i, \forall L, L > \min(T_i)$$

$$\left(\left(\left\lfloor \frac{L-C_i}{k_i T_i} \right\rfloor m_i + 1 \right) + Min\left(\left\lfloor \frac{L - C_i - \left\lfloor \frac{L-C_i}{k_i T_i} \right\rfloor k_i T_i}{T_i} \right\rfloor - 1, m_i - 1 \right)_+ \right) C_i +$$

$$\sum_{j\in\tau-\tau_i}\left(\left\lfloor\frac{L-1-(DBP_j(t)-2)T_j}{k_jT_j}\right\rfloor m_j+Min\left(\left\lfloor\frac{\left(L-1-(DBP_j(t)-2)T_j\right)-k_jT_j\left\lfloor\frac{L-1-(DBP_j(t)-2)T_j}{k_jT_j}\right\rfloor}{T_j}\right\rfloor,m_j\right)\right)C_j\le L$$

Où U est l'ensemble de clients de DBP = 1 qui peuvent arriver au même instant t et τ - U l'ensemble des autres clients. Dans le pire cas cet ensemble U peut inclure un client de chaque source et τ - U = $\varnothing$ (ensemble vide). Dans la pratique pour un ensemble concret τ', ce pire cas peut ne jamais être atteint.

En fait, cette expression de condition suffisante est celle de NP-EDF avec une variable qui est $DBP_j(t)$.

Nous avons démontré [Li03] par ailleurs que pour un système avec des valeurs de m_i et k_i quelconques (avec $m_i < k_i$, pour $i = 1, 2, \ldots N$ qui représente le numéro de source), cette condition suffisante peut être équivalente à la condition définie dans le cas du temps réel dur : (k,k)-firm.

Pour un ensemble concret de sources, un programme développé dans [Li03] peut être utilisé pour évaluer la différence en terme de demande de capacité de traitement du serveur entre (m,k)-firm et (k,k)-firm.

Figure 6 et Figure 7 montrent ce qu'on peut obtenir par ce programme pour le cas concret du Tableau 3. L'abscisse représente un intervalle de temps L et l'ordonnée la demande de serveur devant être exécutée avant la fin de l'intervalle de temps L (i.e. arrivée cumulative du travail). Dans chaque figure la courbe supérieure correspond à la demande de (k,k)-firm et celle inférieure correspond à la demande de (m,k)-firm. On a supposé que toutes les $DBP_i(t) = 1$ (le pire cas) pour (m,k)-firm.

	contrainte (m,k)	C_i	$T_i = D_i$
Source 1	(2,5)	8	12
Source 2	(4,5)	10	20
Source 3	(3,6)	2	5
Source 4	(1,5)	4	6

Tableau 3. Un cas concret du MIQSS

On peut constater que la demande de serveur de (m,k)-firm ne dépasse jamais celle de (k,k)-firm mais les deux courbes se superposent pour des petites valeurs de L.

Comment éviter cette situation indésirable constitue alors un objectif de nos travaux futurs car le dimensionnement du serveur du modèle MIQSS en dépend directement. Dans [Li03] une analyse des causes de la superposition est développée et nous concluons que la meilleure approche d'ordonnancement et les meilleurs κ-patterns doivent être donnés par le serveur (ordonnanceur). Ce qui peut être réalisé par l'établissement d'un protocole de négociation de la QdS entre les sources et l'ordonnanceur.

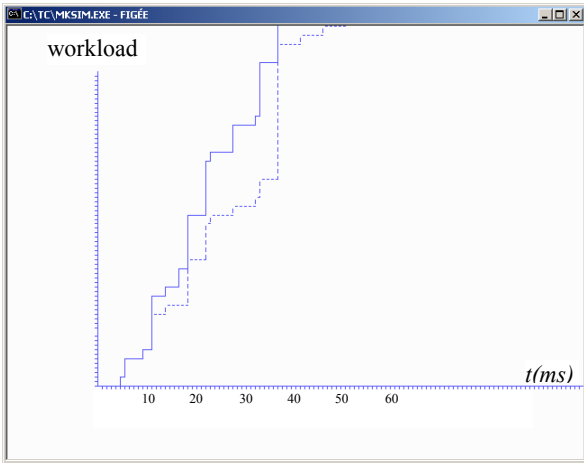

Figure 6. Différence en demande de serveur entre conditions 1 de [Li03] et de [Jeffay91]

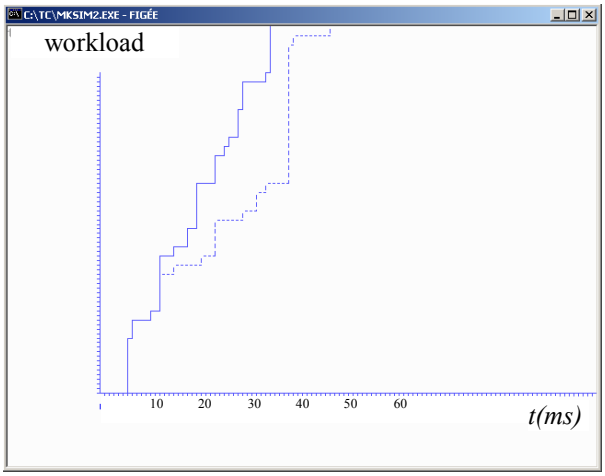

Figure 7. Différence en demande de serveur entre conditions 2 de [Li03] et de [Jeffay91]

5. Conclusion et perspectives

Offrir la QdS temps réel avec dégradation contrôlée selon le modèle (m,k)-firm consiste en une piste intéressante pour la conception des systèmes temps réel adaptatifs. En effet par rapport à la conception des lois de commande adaptatives en fonction de la variation de la QdS dans un système de contrôle-commande distribué, qui est basée sur une métrologie explicite en temps réel de la QdS du réseau [Michaut03], une approche utilisant par exemple DBP a l'avantage d'être simple car la « métrologie » de la QdS du système (certes se réduit au seul paramètre qui est équivalent à la charge) est réalisée implicitement par la k-séquence qui peut être considérée comme un historique de la QdS du réseau. Parmi les algorithmes d'ordonnancement sous contrainte (m,k)-firm, nous préférons les algorithmes dynamiques tels que DBP et DWCS aux algorithmes utilisant un κ-pattern fixe. Ceci pour essentiellement deux raisons : la capacité d'adaptation en-ligne à la variation d'état du système (variation en flux d'entrée, en capacité de traitement du serveur, …) et le potentiel de mieux utiliser le serveur dans le modèle MIQSS. Cette dernière est simple à comprendre. Considérons une source ayant déjà les *m* premiers clients servis. Le serveur, en cas de surcharge, peut ne pas servir les *k-m* clients suivants tout en satisfaisant la contrainte de (m,k)-firm. Tandis qu'avec un κ-pattern fixe, le serveur ayant déjà servi les *m* premiers clients (critiques et optionnels) risque de continuer à servir encore des clients s'il y des clients critiques dans les *k-m* clients suivants selon le κ-pattern.

Nos travaux futurs visent principalement deux directions : 1) Implémentation de la gestion dynamique de QdS selon le modèle (m,k)-firm dans les réseaux; 2) Recherche de conditions suffisantes d'ordonnançabilité avec d'autres algorithmes d'ordonnancement pour la garantie déterministe de (m,k)-firm ainsi que leur exploitation pour le dimensionnement du serveur dans le modèle MIQSS.

References

[ARTIST03] Project IST-2002-34820, Roadmap, "Adaptive Real-Time Systems for Quality of Service Management", http://www.systemes-critiques.org/ARTIST/Roadmaps/ , May 6th 2003.

[Bernat01] Bernat, G., A. Burns and A. Llamosi, "Weakly-hard real-time systems", *IEEE Transactions on Computer*s, 50(4), pp.308-321, April 2001.

[Bernat97] Bernat G. and A. Burns, "Combining (n, m)-hard deadlines and dual priority scheduling", *Proceedings of Real-Time Systems Symposium*, pages 46–57, Dec 1997.

[Chang00] Chang, C. S., *Performance Guarantees in Communication Networks*. New York: Springer-Verlag, 2000.

[Chow01] Chow, M.Y., Y. Tipsuwan, "Network-based control adaptation for network QoS variation", *IEEE Military Communications Conference (MILCOM2001)*, Vol. 1, pp257 -261, 2001.

[Chung90] Chung, J.Y., Liu, J.W. and Lin, K.J., "Scheduling periodic jobs that allows imprecise results", *IEEE Trans. on Computers*, 39(9):1156-1175, sep. 1990.

[El-Gendy03] El-Gendy, M.A., A. Bose, K.G. Shin, "Evolution of the Internet QoS and support for soft real-time applications", *proceedings of the IEEE*, Vol.91, No.7, pp1086-1104, July 2003.

[Furht99] Furht, B. (Editor), *Handbook of multimedia computing*, CRC Press LLC, 1999.

[Hamdaoui95] Hamdaoui M. and P. Ramanathan, "A dynamic priority assignment technique for streams with (m, k)-firm deadlines", *IEEE Transactions on Computer*s, 44(4), 1443–1451, Dec.1995.

[Hamdaoui97] Hamdaoui M. and P. Ramanathan, "Evaluating Dynamic Failure Probability for Streams with (m, k)-firm Deadline", *IEEE Transactions on Computer*s, 46(12), pp.1325–1337, Dec.1997.

[Jeffay91] Jeffay, K., Stanat, D.F. and Martel, C.U., "On Non Pre-emptive Scheduling of Periodic and Sporadic Task", *IEEE real-time systems symposium*, pp129-139, San Antonio (USA), Dec. 4-6, 1991.

[Jumel03] Jumel, F., N. Navet, F. Simonot-Lion, « Influence des performances d'une architecture informatique sur la fiabilité des systèmes échantillonnés ». *Edition Teknéa RTS'2003.* (Paris). 2003.

[Koubâa04a] Koubâa, A., Song, Y.Q., "Loss-Tolerant QoS using Firm Constraints in Guaranteed Rate Networks", *10th IEEE Real-Time and Embedded Technology and Applications (RTAS'2004)*, Toronto (Canada) 25-28 May 2004.

[Koubâa04b] Koubâa, A., *Gestion de la qualité de service temporelle selon la contrainte (m,k)-firm dans les réseaux à commutation de paquets*, Thèse de l'INPL, Nancy, 27 octobre 2004.

[LeBoudec02] Le Boudec, J.Y. and P. Thiran, *Network Calculus: A Theory of Deterministic Queueing Systems For The Internet*, Online Version of the Book of Springer Verlag – LNCS 2050, July 2002.

[Li03] Li, J., « Sufficient condition for guaranteeing (m,k)-firm real-time requirement under NP-DBP-EDF scheduling », *Rapport de DEA d'informatique de Lorraine*, LORIA, Juin 2003.

[Li04] Li, J., Y.Q. Song, F. Simonot-Lion, « Schedulability analysis for systems under (m,k)-firm constraints », *WFCS2004*, Vienna (Austria), Sept. 22-24, 2004.

[Martin04] Martin S., « *Maîtrise de la dimension temporelle de la Qualité de Service dans les réseaux* », Thèse Université Paris 12 (projet Hipercom INRIA), 6 Juillet 2004.

[Michaut03] Michaut F., « *Adaptation des applications distribuées à la Qualité de Service fournie par le réseau de communication* », Thèse UHP Nancy 1, 26 Nov. 2003.

[Nilsson98] Nilsson, J., et al, "Stochastic analysis and control of real-time systems with random time delays", *Automatica*, vol.34, pp.57-64, 1998.

[Parekh93] Parekh, A.K., R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case", *IEEE/ACM Transactions on Networking*, Volume 1, Issue 3, pp344-357, June 1993.

[Quan00] Quan G. and X. Hu, "Enhanced Fixed-priority Scheduling with (m, k)-firm Guarantee", *Proc. Of 21st IEEE Real-Time Systems Symposium*, pp.79-88, Orlando, Florida, (USA), November 27-30, 2000.

[Ramanathan99] Ramanathan P., "Overload management in Real-Time control applications using (m, k)-firm guarantee". *IEEE Transactions on Parallel and Distributed System*s, 10(6):549–559, June 1999.

[Wang02] Wang, S., Y. Wang, K. Lin, "Integrating Priority with Share in the Priority-Based Weighted Fair Queueing Scheduler for Real-Time Networks" *Journal of RTS,* p119-149, 2002.

[West04] West, R., Y. Zhang, K. Schwan and C. Poellabauer, "Dynamic Window-Constrained Scheduling of Real-Time Streams in Media Servers", *IEEE Transactions on Computers*, Volume 53, Number 6, pp. 744-759, June 2004.

[West99] West, R. and K. Schawn, "Dynamic window-constrained scheduling for multimedia applications", *6th International Conf. On Multimedia Computing and Systems, ICMCS'99*, IEEE, June 2000.