

# Gestion dynamique de la QoS temps réel selon $(m,k)$ -firm

Yeqlong Song, Anis Koubâa

LORIA – UHP Nancy 1 – INPL, 2 avenue de la Forêt de Haye, 54516 Vandoeuvre  
[song@loria.fr](mailto:song@loria.fr); [Anis.Koubaa@loria.fr](mailto:Anis.Koubaa@loria.fr)

---

**RÉSUMÉ.** Un modèle de QoS (qualité de service) temps réel selon  $(m,k)$ -firm offre plus de souplesse et une meilleure utilisation de ressources par rapport à la QoS garantie. Il ouvre des nouvelles perspectives pour la mise en oeuvre des mécanismes de gestion de la QoS dans des réseaux et systèmes temps réel adaptatifs. Pour prendre en compte ce nouveau type de contrainte lors de l'ordonnancement d'accès aux ressources, les algorithmes d'ordonnancement développés pour le système temps réel dur (par exemple EDF, FP) et pour le réseau à QoS (par exemple WFQ) doivent être étendus. 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 deux algorithmes à l'issue de nos travaux récents : Matrix-DBP et  $(m,k)$ -WFQ. Nous définissons aussi la notion de la dégradation gracieuse de la QoS selon  $(m,k)$ -firm et discutons son application à la gestion dynamique de la QoS dans les réseaux.

**MOTS-CLÉS :** Qualité de service, Temps réel,  $(m,k)$ -firm, Ordonnancement non-préemptif, Network Calculus.

---

## 1. Introduction

Le temps de développer des applications temps réel en se basant sur processeur dédié et logiciel dimensionné selon le pire cas est bel et bien révolu. Depuis quelques années, la tendance est plutôt vers un développement par COTS (Commercial Off-the Shelf), c'est à dire par assemblage des composants logiciels existants en utilisant des processeurs et réseaux génériques. On peut citer des exemples du domaine de e-maintenance, e-automation, DCS (Distributed Control System) construit autour de réseau Ethernet industriel ou encore des nouvelles applications temps réel sur Internet (Jeux interactifs, Simulation interactive distribuée, ...). Mis à part les contraintes économiques, cette tendance correspond aussi au désir de disposer d'un système *adaptatif* au sens où il s'adapte au changement de l'application (évolutif), de son état (flexible) et de son environnement (robustesse face aux aléas). Avec des ressources partagées qui sont aussi limitées, une gestion fine et dynamique de la QoS sera indispensable pour satisfaire à ce besoin.

L'affectation dynamique des ressources existe depuis long temps dans les réseaux à QoS (Intserv, Diffserv, MPLS et ATM). Mais elle ne fournit généralement que deux classes de services : service garanti et best-effort. Avec des applications temps réels toujours de plus en plus nombreuses à vouloir partager cette classe garantie, il se pose le problème d'utilisation efficace des ressources partagées, et une définition plus fine des niveaux de QoS semble être une solution.

En effet, ces deux classes de QoS sont directement liées à la classification d'applications en temps réel *dures* et *souples*. Un système temps réel ou un réseau à QoS associe souvent une *échéance* à chaque *action* que le système doit exécuter. Et une telle action est *invoquée* de façon récurrente (périodique, sporadique, apériodique). Selon la conséquence du non respect de l'échéance par le système, les contraintes temps réel sont classées en dures et souples. Une action est dite sous contrainte temps réel dure si le non respect de l'échéance d'exécution d'une de ses invocations peut causer des conséquences catastrophiques sur l'environnement contrôlé (par exemple, la transmission d'une consigne de freinage par le système « X-by-Wire » dans un véhicule). Notons que pour obtenir une garantie déterministe (absolue) du respect des contraintes temps réel dures, les ressources sont souvent réservées statiquement selon l'hypothèse du pire cas. Une action est dite sous contrainte temps réel souple si le non respect de l'échéance d'exécution d'une ou plusieurs de ses invocations ne provoque pas de comportements incorrects de l'environnement sous son contrôle (par exemple : un procédé industriel, un véhicule, ...), mais seulement dégrade les performances (par exemple, la diffusion en temps réel d'informations multimédias de loisir sur l'Internet peut tolérer des retards de certain nombre de paquets). Nous pouvons remarquer que la notion de niveaux de QoS s'accommode bien à ce type d'action. Pour une action ayant des contraintes temps réel souples, on définit la notion du temps réel « *firm* » selon le traitement ou non des invocations dont le respect de leur échéance est impossible par le système. Une action ayant des contraintes temps réel souples est dite *firm* si ses

invocations dont l'échéance ne pouvant pas être respectée, sont *rejetées* par le système (non exécutées) car le traitement en retard des invocations est considéré comme inutile, permettant ainsi de réduire la charge du système. Un exemple typique est la transmission des paquets de voix téléphonique ou de vidéo. Pour préciser le nombre acceptable des invocations pouvant être écartées, une action est dit sous contrainte temps réel *(m,k)-firm* [Hamdaoui95] si au moins  $m$  parmi  $k$  invocations consécutives 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). Cette notion a été étendue dans [Bernat97] et [Bernat01] sous le nom de WHRT (Weakly-Hard Real-Time) pour spécifier plus finement la répartition des  $m$  invocations dans les  $k$  consécutives. Notons que des idées similaires peuvent être trouvées dans des travaux d'ordonnancement en cas de surcharge autour du concept de calcul imprécis [Chung90], de « skip-Over » [Koren95] et de l'ordonnancement du problème de « Pinwheel » [Chan92], [Baruah98].

Que ce soit pour le temps réel dur ou souple, un système peut donner une *garantie* du respect de l'échéance soit *déterministe*, soit *probabiliste*. Pour une application (environnement) parfaitement connue, la garantie déterministe est faisable. Pour d'autres cas plus généraux cette garantie peut être complètement utopique : soit la demande en ressources pour le pire cas est trop coûteuse, soit parce qu'il est simplement impossible de prévoir et décrire le pire cas imaginable à cause de la dynamique de l'environnement et des perturbations aléatoires imprévisibles. Prenons encore l'exemple du véhicule sous « X-by-Wire », une garantie probabiliste de  $1 - 10^{-9}$  est jugée suffisante en considérant les possibilités de défaillances et de perturbations aléatoires de l'environnement [Wilwert03], [Hammett03].

Au lieu de rechercher un système avec une garantie temps réel dur déterministe qui conduira inévitablement à un surdimensionnement du système, il est préférable d'avoir un système flexible, acceptant différents niveaux de QoS, capable de s'adapter au changement de l'environnement et faire face aux aléas (avec des mécanismes en-ligne). Le besoin d'un système temps réel flexible gérant la QoS est aussi l'un des trois objectifs du projet Européen ARTIST et d'autres arguments du pourquoi peuvent être trouvés dans [ARTIST03]. On peut d'ailleurs noter qu'un bon nombre d'applications classées temps réel dures n'ont pas forcément de contrainte temps réel dure sur toutes les actions et toutes les invocations d'une action. Prenons toujours l'exemple du véhicule, du fait que la consigne du conducteur est transmise plusieurs fois (par sur-échantillonnage) par le système « X-by-Wire », la perte ou retard d'un certain nombre de trames contenant cette même consigne peut être tolérée. En effet il suffit qu'une de ses trames parvienne à l'actionneur avant l'échéance pour que le véhicule fonctionne correctement. D'autres exemples similaires peuvent être trouvés dans [Ramanathan99], [Cervin00]. *(m,k)-firm* nous paraît convenable pour définir la QoS temps réel. Un système conçu selon *(m,k)-firm* permettra d'offrir des niveaux de QoS varié entre  $(k,k)$  et  $(m,k)$  avec autant de niveaux intermédiaires correspondant aux différentes valeurs possibles entre  $k$  et  $m$ . Il semble aussi intuitive qu'un système garantissant au minimum *(m,k)-firm* exigera moins de ressources par rapport à un système  $(k,k)$ -firm.

Que ce soit dans des réseaux ou des systèmes temps réel, l'ordonnancement d'accès aux ressources partagées se base essentiellement sur la notion de priorité. Il est clair que pour la prise en compte de la contrainte *(m,k)-firm* des algorithmes d'ordonnancement nouveaux restent à développer et les algorithmes classiques tels que FP (Fixed Priority), EDF (Earliest Deadline First), WFQ (Weighted Fair Queueing) doivent être étendus.

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

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 détaille Matrix-DBP qui propose une amélioration à DBP-EDF. La section 4 décrit *(m,k)-WFQ* qui permet à un serveur WFQ de prendre en compte plus efficacement des contraintes temporelles des flux. La section 5 discute sur comment gérer la QoS selon le concept de *(m,k)-firm*. Section 6 conclut le papier et indique les perspectives.

## 2. Etat de l'art sur les travaux autour de *(m,k)-firm*

Nous précisons d'abord le modèle permettant de décrire les systèmes puis expliquons les spécifications de WHRT et les algorithmes principaux pour *(m,k)-firm*.

2.1. **Modèle général**

Considérons le modèle MIQSS (Multiple Input Queues Single server) dans la Figure 1. Le problème fondamental est comment ordonnancer des demandes d'accès au serveur commun, tout en satisfaisant leurs contraintes temporelles et en optimisant le taux d'utilisation du serveur. Dans le contexte de systèmes distribués, ce serveur peut être 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 (qui est plus difficile à analyser par rapport au cas préemptif).

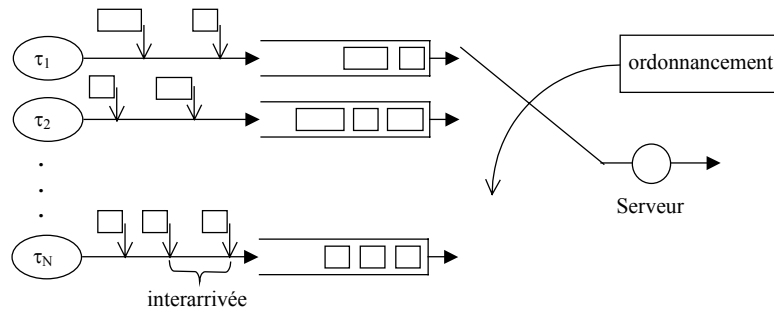


Figure 1. *Modèle général pour la description des systèmes : MIQSS*

Une source  $\tau_i$  est caractérisée par son flux de demande de travail qu'elle génère (les blocs dans la Figure 1)  $F_i$  et les contraintes temps réel  $CTR_i$ . La fonction  $F_i$  peut être :

- **Périodique ou sporadique**:  $(C_i, T_i)$  pour date initiale quelconque et  $(r_i, C_i, T_i)$  pour date initiale concrète  $r_i$ , où  $C_i$  est le temps d'exécution d'une invocation dans le serveur et  $T_i$  la période d'inter-arrivée (ou d'inter-arrivée minimale pour sporadique).
- **Périodique avec gignés** :  $(C_i, T_i, J_i)$  ou  $(r_i, C_i, T_i, J_i)$ .
- **$(\sigma_i, \rho_i)$ -borné** : une courbe linéaire caractérisée par la taille de rafale  $\sigma_i$  et le débit moyen  $\rho_i$  qui majore la vraie fonction cumulative d'arrivée du travail [LeBoudec02][Chang00].
- **Aléatoire** en suivant une loi d'arrivée probabiliste avec un temps moyen d'inter-arrivée  $T_i$  et un temps moyen d'exécution de travaux  $C_i$  (par exemple : Poisson pour une loi d'inter-arrivée exponentielle et taille de blocs constante, Poisson composée pour inter-arrivées exponentielle et taille de blocs variable).

Notons que les deux premiers cas peuvent être transposés en un modèle de flux  $(\sigma_i, \rho_i)$ -borné [Koubâa03]. Les contraintes temps réel  $CTR_i$  dans notre travail sont toujours données par  $(D_i, m_i, k_i)$  où  $D_i$  est l'échéance relative à l'instant d'arrivée d'un travail et  $(m_i, k_i)$  sont les deux paramètres de la contrainte  $(m_i, k_i)$ -firm.

Une source sous contrainte temps réel  $(m, k)$ -firm peut se trouver dans l'un des deux états : normal et échec dynamique (*dynamic failure*) [Hamdaoui95]. La connaissance de son état à l'instant  $t$  dépend de l'historique du traitement des  $k$  dernières invocations. Si on associe '1' à une invocation respectant son échéance et '0' à une invocation ratant son échéance, cet historique est entièrement décrit par une suite de longueur  $k$  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.

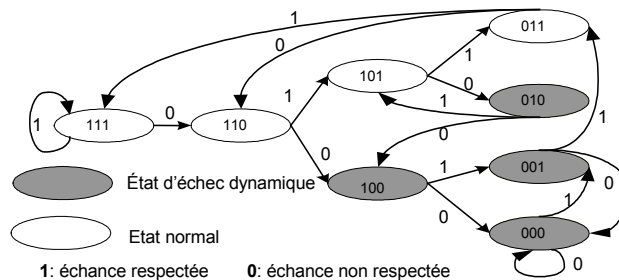


Figure 2. *Diagramme d'état-transition d'une source avec (2,3)-firm*

Dans un système qui pourrait être modélisé par MIQSS, on peut définir l'état du système d'un instant  $t$  à partir des états des sources du même instant. Un système est dit en état d'échec dynamique si au moins une de ses sources est en échec dynamique. Le terme « dynamique » veut simplement dire que l'état est transitoire puis qu'il évolue en fonction du temps.

## 2.2. $(m,k)$ -firm et WHRT

Pour étendre cette notion de  $(m,k)$ -firm, [Bernat97] et [Bernat01] ont proposé trois autres formes qui correspondent à la complémentarité et la consécuitivité qui facilite l'expression de contraintes dites WHRT :

- $(\bar{m}, \bar{k})$ -firm : pas plus que  $m$  invocations avec *échéance non respectée* dans une fenêtre quelconque de  $k$  arrivées consécutives
- $\langle m, k \rangle$ -firm : au moins  $m$  invocations *consécutives* avec échéance respectée dans une fenêtre quelconque de  $k$  arrivées consécutives
- $\langle \bar{m}, \bar{k} \rangle$ -firm : pas plus que  $m$  invocations *consécutives* avec *échéance non respectée* dans une fenêtre quelconque de  $k$  arrivées consécutives

Il convient de remarquer que certaines de ces formes peuvent être exprimées sous forme de  $(m,k)$ -firm :

- $(\bar{m}, \bar{k})$ -firm : équivalente à  $(k-m, k)$ -firm
- $\langle m, k \rangle$ -firm : pas d'équivalence dans  $(m,k)$ -firm
- $\langle \bar{m}, \bar{k} \rangle = \langle \bar{m} \rangle$  : au fait il est facile de constater qu'avec  $\langle \bar{m}, \bar{k} \rangle$ , on ne peut jamais avoir plus de  $m$  invocations *consécutives* avec *échéance non respectée* quelque soit la taille de  $k$  pourvu que  $m < k$ . Il soit aussi vrai qu'une source respectant  $(m,k)$ -firm inclut le cas particulier de  $\langle \bar{k-m} \rangle$ .

## 2.3. Algorithmes d'ordonnement pour $(m,k)$ -firm

Il existe aujourd'hui principalement deux familles d'algorithmes qui prennent en compte  $(m,k)$ -firm : dynamique (exemple : DBP) et statique (exemple : EFP). Par algorithme dynamique nous voulons dire que la priorité affectée à chaque invocation 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. Il s'accommode bien à la gestion en-ligne de la QoS. Le problème est qu'il ne donne souvent qu'une garantie probabiliste de  $m$  sur  $k$  (best-effort). C'est le cas de DBP et la première version de DWCS (Dynamic Window Constrained Scheduling) [West99]. Une version améliorée de DWCS [West02] permet de donner une garantie déterministe de  $m$  sur  $k$  sous des conditions particulières (même  $C_i$  pour toutes les sources, serveur non conservatif, ce qui diminue l'efficacité du serveur). 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 et EFP. Une synthèse plus large peut être trouvée dans [Wang02].

### 2.3.1. DBP (Distance Based Priority)

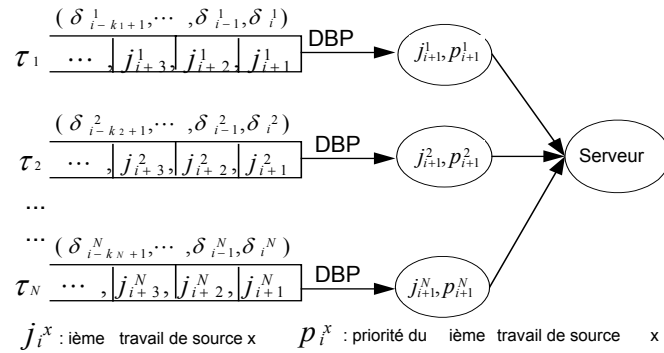
DBP [Hamdoui95] est la façon la plus directe pour la prise en compte de la contrainte  $(m,k)$ -firm. Pour une  $k$ -séquence donnée, DBP définit la distance d'aller à un état d'échec pour un instant donné comme le nombre consécutif de bits 0 qu'on doit rajouter. La priorité que DBP donne au travail en tête de la queue correspondante à la  $k$ -séquence (cf. Fig.1) est égale à cette distance. Si la source se trouve déjà en état d'échec dynamique (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 travail en tête de la queue est de priorité 2 si les 5 travaux précédents donne une  $k$ -séquence (11011), il est de priorité 3 si les 5 travaux précédents donne une  $k$ -séquence (10111).

Formellement, selon [Hamdaoui95] la priorité est évaluée comme suit. Nous notons par  $s_j = (\delta_{i-k_j+1}^j, \dots, \delta_{i-1}^j, \delta_i^j)$  la k-séquence de la source  $\tau_j$ , par  $l_j(n,s)$  la position (en comptant à partir de droite) du  $n^{\text{ième}}$  échéance respectée (ou 1) dans  $s_j$ , la priorité du  $(i+1)^{\text{ème}}$  travail de  $\tau_j$  est donnée par :

$$P_{-DBP_{i+1}^j} = k_j - l_j(m_j, s_j) + 1 \quad (1)$$

Notons lorsqu'il y a moins de  $n-1$  dans  $s$ ,  $l_j(n,s) = k_j + 1$ , 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 travaux en tête des queues

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

### 2.3.2. EFP (Enhanced Fixed Priority)

EFP est proposé dans [Hamdaoui97], [Ramanathan99]. Pour prendre en compte la contrainte (m,k)-firm, chaque source marque parmi ses  $k$  invocations consécutives  $m$  invocations *critiques* et  $(k-m)$  invocations *optionnelles* tout comme dans le modèle de calcul imprécis [Chung90]. En faisant ainsi le serveur pourra rejeter une invocation optionnelle en cas de surcharge (au cas où son échéance ne pourrait plus être respectée par le serveur). Toutes les invocations critiques peuvent être ordonnancées par un algorithme à priorité fixe tel que RM (Rate Monotonic). Les travaux optionnels sont servis avec la priorité la plus basse selon la politique FIFO. Pour commencer le marquage, la première invocation de chaque source est marquée critique.

Pour une source  $\tau_i$ , la technique pour le marquage des invocations critiques et optionnelles selon sa contrainte  $(m_i, k_i)$ -firm est donnée par l'équation suivante.

La  $a^{\text{ième}}$  invocation ( $a = 0, 1, \dots$ ) est marquée critique si  $a$  vérifie :

$$a = \left\lfloor \frac{a \cdot m_i}{k_i} \right\rfloor \cdot \frac{k_i}{m_i} \quad (2)$$

#### Exemple:

$$\begin{aligned} \tau_1 : C_1 = 1, T_1 = 2, m_1 = 2, k_1 = 3, \\ \tau_2 : C_2 = 3, T_2 = 4, m_2 = 2, k_2 = 3 \end{aligned}$$

Pour  $\tau_1$ , selon l'équation 2, à partir de sa 8ème invocation, une parmi trois est marquée optionnelle. Pour  $\tau_2$ , les invocations n°24, 48, 84, 108, ..., sont marquées optionnelles.

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 trois problèmes:

- La première invocation de chaque source est marquée critique par défaut. Ce qui force artificiellement le système de se retrouver dans un « pire cas ».
- L'équation 2 distribue régulièrement les  $m$  invocations 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 se verront leur invocations critiques distribuées 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 invocations critiques de toutes les sources devrait donner une meilleure ordonnancement, [Quan00] a amélioré l'algorithme dans [Hamdaoui97, Ramanathan99]. Il a été d'abord prouvé que trouver une partition optimale est NP-dur. Puis, un algorithme heuristique est proposé pour optimiser la distribution de  $m_i$  invocations critiques parmi  $k_i$  invocations consécutives en prenant en compte la relation entre les sources.

### 3. Matrix-DBP

DBP+EDF a été initialement conçu pour être appliqué à la transmission des paquets vidéos où on suppose que les sources ont des flux semblables (même moyenne et loi de distribution de  $C_i$ ,  $T_i$  et  $D_i$ ) mais avec des  $(m_i, k_i)$  différents. Appliquer DBP+EDF dans une application temps réel plus générale selon le modèle MIQSS (Figure 1) ne donne pas toujours des résultats optimaux [Poggi03]. L'un des problèmes est que DBP affecte la priorité que selon la  $k$ -séquence de la source. La notion de priorité globale n'existe pas bien que les priorités locales soient comparées par le serveur. Un deuxième problème est lié au fait que  $(m, k)$ -firm est considéré comme plus important que les autres paramètres temporels comme par exemple l'échéance car EDF n'est utilisée qu'en cas d'égalité de priorité.

Dans [Poggi03] nous avons proposé une solution appelée « Matrix-DBP » pour remédier à ces deux problèmes pour le cas de flux périodiques. L'idée consiste à utiliser une matrice pour prendre en compte les paramètres temporels tels que *l'échéance*, le *temps d'inter-arrivée*, le *temps d'exécution* ainsi que *l'importance relative* entre les travaux des sources différentes. La construction de cette matrice s'appuie sur les deux conditions d'ordonnancement, nécessaires mais non suffisantes, suivantes.

Pour  $N$  sources périodiques  $\tau = (\tau_1, \tau_2, \dots, \tau_N)$  avec  $\tau_i = \{T_i, D_i, C_i, m_i, k_i\}$ , l'ensemble  $\tau$  n'est pas ordonnancement si l'une des deux conditions suivantes n'est pas remplie :

$$C1: \text{Charge globale} \quad \sum_{i=1}^N \left[ \frac{C_i m_i}{T_i k_i} \right] \leq 1$$

$$C2: \text{Ordonnancement mutuelle} \quad n_{i,j} \leq k_i - m_i$$

$$n_{j,i} \leq k_j - m_j$$

Où  $n_{i,j}$  est le nombre minimal d'échéances ratées de la source  $i$  pendant le temps de service d'un travail de la source  $j$ .

La condition C1 impose que la charge globale ne doit pas dépasser la capacité du serveur. Notons que le facteur de blocage due à la non-préemption n'est pas considéré comme nous citons ici que la condition nécessaire. La condition C2 veut simplement dire que pour n'importe quelles deux sources  $i$  et  $j$ , pendant le service d'un travail de  $i$  (resp.  $j$ ) la source  $j$  (resp.  $i$ ) ne doit pas subir plus de  $k_i - m_i$  (resp.  $k_j - m_j$ ) échéances ratées. Le calcul de  $n_{i,j}$  est donné par [Poggi03] :

$$n_{i,j} = \left\lceil \frac{C_j + 2C_i - D_i}{T_i} \right\rceil - 1 \quad (3)$$

Afin de prendre en compte l'ordonnabilité mutuelle entre toutes les sources, une matrice de dimension  $N \times N$  est utilisée avec comme élément générique  $m_{i,j} = \max(0, n_{i,j})$ .

La priorité DBP sera corrigée en utilisant cette matrice selon l'équation suivante :

$$P\_MatrixDBP_{i+1}^j = P\_DBP_{i+1}^j - \max_{k=1, \dots, N} (m_{j,k} Q_k(t)) \quad (4)$$

Où la variable Booléenne  $Q_j(t)$  indique si la queue de source  $\tau_j$  est vide ou pas à l'instant  $t$ .  $Q_j(t) = 1$  si la queue n'est pas vide. Le vecteur  $(Q_1(t), Q_2(t), \dots, Q_N(t))$  peut donc être utilisé pour connaître quels sont les travaux en tête des queues qui sont en compétition d'accès au serveur à l'instant  $t$ .

Pour donner une idée concrète de l'amélioration de DBP par utilisation de Matrix-DBP nous montrons deux exemples.

### Exemple 1 : Cas de deux sources Sa et Sb

	(m,k)-contrainte	$C_i$	$T_i = D_i$	k-séquence initiale	P_DBP( $t_0$ )	P_MatrixDBP( $t_0$ )
Sa	(4,5)	15	30	{01111}	2	2
Sb	(2,5)	2	5	{00101}	3	1

Tableau 2. Configuration simple de deux sources

Supposons qu'à l'instant  $t_0$ , les deux sources ont généré deux travaux avec les k-séquences initiales présentées dans le Tableau 2. On voit que le nombre minimal d'échéances ratées par la source Sb durant le temps de service d'un travail de la source Sa est 2 (et au maximum 3). La matrice selon l'équation 3 est donc :  $M = \begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix}$  ce qui a permis la correction de la priorité du travail de la source Sb de 3 à 1.

La Figure 4 montre que selon DBP, la source Sb va être en état d'échec dynamique. Cette situation est évitée par Matrix-DBP car la priorité est donnée à la source Sb à l'instant  $t_0$ .

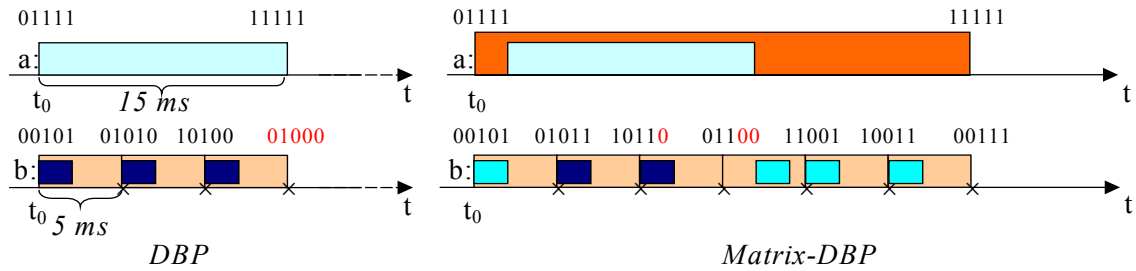


Figure 4. DBP vs. Matrix-DBP

### Exemple 2 : Simulation d'un cas complexe

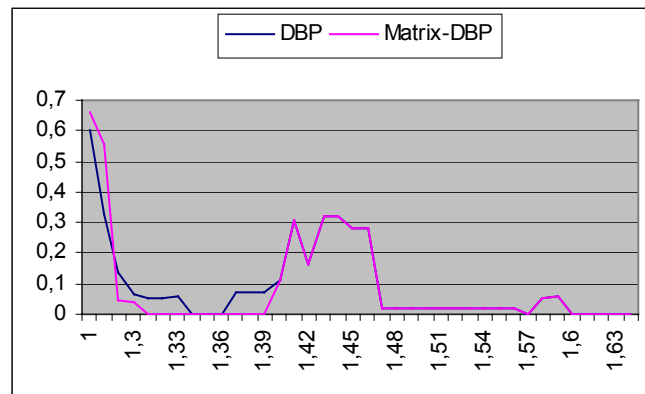
	(m,k)-contrainte	$C_i$	$T_i = D_i$
Stream 0	(2,5)	8/c	12
Stream 1	(4,5)	10/c	20
Stream 2	(3,6)	2/c	5
Stream 3	(1,5)	4/c	6

Tableau 3. Configuration d'un cas complexe

Nous prenons comme paramètre de performance la proportion du temps où le système se trouve en état d'échec dynamique. Afin de comparer DBP et Matrix-DBP avec différentes capacités du serveur, nous faisons varier la capacité du serveur  $c$  de 1,00 à 1,50. Le cas de  $c = 1$  correspond à une charge globale de 1 selon la condition nécessaire C1. La matrice avec  $c = 1$  selon l'équation 3 est :

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

On voit que la condition C2 est vérifiée comme dans le Tableau 3 toutes les sources vérifient  $k - m \geq 1$ . Tous les éléments de cette matrice sont devenus 0 pour la première fois quand  $c = 1,5$  ( $m_{3,1}$  est le dernier élément qui devient 0 selon l'équation 3). Donc logiquement la simulation pour  $c > 1,5$  doit donner les mêmes performances pour matrix-DBP et DBP. Notons qu'en cas d'égalité de priorité, EDF est utilisé par défaut pour DBP et Matrix-DBP.



**Figure 5.** Proportion du temps où le système est en état d'échec dynamique en fonction de  $c$

On peut constater selon la Figure 5 que Matrix-DBP donne une meilleure performance pour  $c > 1,2$ . Pour la valeur de  $c$  entre 1 et 1,2 Matrix-DBP se comporte pire que DBP. Ce phénomène est due au choix de ne prendre que le nombre minimal d'échéances ratées lors du calcul des éléments de la matrice (équation 3). Au fait, la vraie valeur de  $m_{i,j}$  se trouve entre  $n_{i,j}$  et  $n_{i,j} + 1$ . Si on pouvait imaginer une version « fluide » de notre modèle non-préemptif en autorisant à  $m_{i,j}$  de prendre des valeurs réelles au lieu des valeurs entières, ce phénomène pourrait disparaître. Nous constatons également la non-monotonie des courbes car l'augmentation de la capacité du serveur peut parfois dégrader les performances. Ce phénomène est à la fois lié à la non-préemption et aux anomalies de Richards [Graham76]. La Figure 5 montre aussi que toutes les contraintes  $(m_i, k_i)$ -firm sont garanties (i.e. probabilité d'échec = 0) pour  $c > 1,6$ . Du point de vue théorique, la condition suffisante sur  $c$  pour une garantie déterministe de toutes les contraintes  $(m_i, k_i)$ -firm est bornée par celle du cas de  $(k_i, k_i)$ -firm (c.à.d. temps réel dur) donnée dans [Jeffay91]. Nous avons donné une borne moins pessimiste dans [Li03].

#### 4. $(m, k)$ -WFQ

L'ordonnanceur WFQ (Weighted Fair Queueing) est déployé dans les commutateurs et routeurs du réseau Internet pour fournir de la QoS grâce à ses propriétés de garantie de bande passante et de délai borné pour des flux  $(\sigma, \rho)$ -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 garantie à chaque flux servi la proportion de la bande passante réservée selon son coefficient de partage  $\Phi_i$ . Chaque paquet de message est estampillé par un tag appelé temps virtuel de départ. Le serveur sélectionne toujours le paquet dont son temps virtuel de départ est le plus proche futur de l'instant de sélection. Dans WFQ le temps virtuel de départ est défini par :

$$F_i^k = \max\{F_i^{k-1}, V(t)\} + \frac{L_i^k}{\Phi_i} \quad (5)$$

avec

- $F_i^k$  : temps virtuel de départ du  $k^{ième}$  paquet de l' $i^{ième}$  flux,
- $V(t)$  : le temps virtuel quand le  $k^{ième}$  paquet arrive,
- $\Phi_i$  : le coefficient de partage de l' $i^{ième}$  flux,
- $L_i^k$  : la taille du  $k^{ième}$  paquet de l' $i^{ième}$  flux,
- $\max\{F_i^{k-1}, V(t)\}$  : le temps virtuel du début de service du  $k^{ième}$  paquet.

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

$$D_{i, \max} = \frac{\sigma_i}{g_i} + \frac{L_{\max}}{c} \quad (6)$$

Où,  $L_{\max}$ , est la taille maximale du paquet parmi tous les paquets dans tous les flux et  $c$  étant la capacité de traitement du serveur. Nous rappelons qu'un flux est dit  $(\sigma, \rho)$ -borné si sa fonction cumulative d'arrivée  $R(t)$  vérifie la relation  $\forall 0 \leq s \leq t, R(t) - R(s) \leq \sigma + \rho(t-s)$  avec  $\sigma$  la taille maximale de rafale et  $\rho$  le débit moyen à long terme.

La borne sur le temps de réponse fournie par WFQ à une source de flux est étroitement liée au coefficient de partage de la bande passante  $\rho_i$  et à la taille de la rafale  $\sigma_i$ . 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 WFQ priorisé proposé dans [WangS02] mais la notion de (m,k)-firm n'est pas prise en compte.

Nous avons proposé dans [Koubâa03a] une approche appelée (m,k)-WFQ. Pour que l'ordonnanceur WFQ puisse prendre en compte les contraintes temporelle (m,k)-firm, la source marque  $m$  paquets critiques parmi tous les  $k$  paquets consécutifs et les restes étant optionnels. L'ordonnanceur (m,k)-WFQ estampille ensuite le paquet par son temps virtuel de départ décrit par l'équation 5. L'algorithme est décrit dans la Figure 6. Le processus de service est activé quant 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 leurs queues. Si aucun paquet critique n'existe, le choix sera fait parmi les paquets optionnels. Après, si le paquet sélectionné est critique, il est exécuté (ou transmis) directement par le serveur, sinon et si le paquet est optionnel, l'ordonnanceur vérifie avant son exécution si ce paquet pouvait éventuellement satisfaire son échéance. Si l'échéance souhaitée est ratée ou sera ratée 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 résulte 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 réels par rapport à WFQ standard. Dans ce qui suit nous montrons quantitativement cette amélioration par simulation d'un exemple.

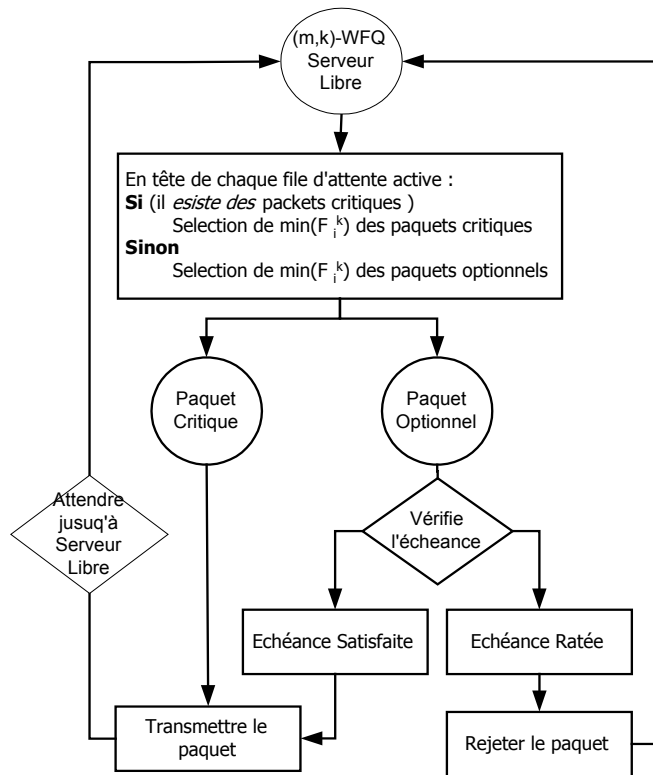


Figure 6. Algorithme (m,k)-WFQ

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. Tableau 4 récapitule les paramètres de simulation pour chacun des flux.

	(m,k)	Débit	Trafic	$\kappa$ -pattern	Echéance
<b>Voix</b>	(4,5)	64 kb/s	ON/OFF (500/755/50)ms	11011	10 ms
<b>Vidéo</b>	(3,5)	2Mb/s	Pseudo Périodique ~2Mb/s	10110	4 ms
<b>FTP</b>	(0,1)	7,936 Mb/s	Pseudo Périodique ~7.936 Mb/s	0	Infinie

Tableau 4. Configuration simulée

Pour le marquage de paquets en critiques et optionnels, afin de simplifier nous ne considérons ici que le cas de  $\kappa$ -pattern fixe.

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  $\kappa$ -pattern fixe le profil de la séquence comme : 11011 11011 11011 ...

La deuxième source est une source CBR (Constant Bit Rate) qui génère un flux vidéo pseudo-périodique de 2 Mb/s. L'échéance des paquets est fixée à 4 ms avec une garantie de type (3,5). Le  $\kappa$ -pattern fixe le profil de la séquence comme : 10110 10110 10110 ...

La troisième source est un agrégat de flux FTP qui consomme le reste de la bande passante ayant donc un débit de 7,936 Mb/s. 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. Pour cela, tous les paquets issus de cette application

sont considérés comme optionnels. Cependant, nous configurons l'ordonnanceur (m,k)-WFQ de ne rejeter que les paquets optionnels des sources temps-réel, voix et vidéo. En effet, le flux FTP est vulnérable à la perte de paquets. Ainsi, comme tous les paquets de ce flux sont optionnels, le serveur est averti de faire passer tous ses paquets. Ceci est réalisé en mettant une échéance infinie pour les flux best-effort.

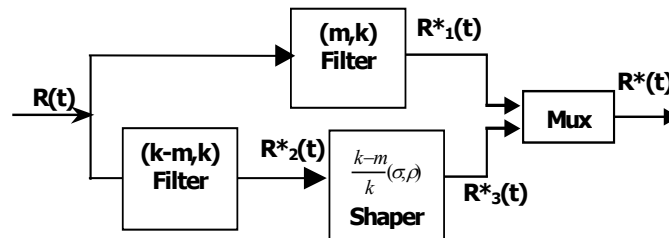
Le Tableau 5 montre les bornes mesurées sur le temps de réponse des paquets pour chacun des flux fournis par les deux serveurs.

	(m,k)-WFQ	WFQ
<b>Voix</b>	9,769	4776,83
<b>Vidéo</b>	3,999	41,084
<b>FTP</b>	3,837	18,048

**Tableau 5.** Bornes sur les délais (ms)

Comme prévu, (m,k)-WFQ fournit une garantie plus étroite sur le délai pour les flux temps-réel. Dans ce scénario, il est remarquable que le délai maximal garanti par WFQ au trafic de la voix soit assez grand. Ce résultat découle de deux facteurs majeurs (cf. équation 6) : 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.

Pour fournir la garantie (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 7 montre le modèle qui nous a permis la déduction de cette borne.



**Figure 7.** Modèle de Network calculus

Le calcul de délai utilise le formalisme du Network Calculus [LeBoudec02]. Dans [Koubâa03b] 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 7, 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 servi 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.

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

$$D_{\max}^* = \lambda_{m,k} \cdot \frac{\sigma}{g} + \lambda_{k-m,k} \cdot \frac{b}{g} + \frac{L_{\max}}{c} \quad (7)$$

Avec  $b \leq \sigma$  la taille maximale de rafale des paquets optionnels qui pourrait être transmise 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  $b = gD_{op}$ .

## 5. Gestion de la QoS selon (m,k)-firm

Les deux modèles de QoS IntServ (RFC1633) et DiffServ (RFC2475) connus dans la communauté de l'Internet visent à fournir des garanties déterministe et probabiliste sur un réseau du type « best effort ». Pour être capable de fournir une garantie de QoS, le trafic doit être borné et des ressources suffisantes doivent être réservées (en utilisant le protocole RSVP par exemple). Pour cela quatre mécanismes de base ont été définis : *régulation de trafic* dont le plus connu est « leaky bucket » qui forme le flux d'entrée en un flux  $(\sigma, \rho)$ -borné ; *ordonnement de paquets* pour le partage efficace des accès aux ressources dont le plus répandu est WFQ (Weighted Fair Queueing) ; *gestion de buffers* pour la détection et résolution de débordement de mémoires tampon dont le plus connu est RED (Random Early Discard) ; *contrôle d'admission* des nouvelles demandes de connexion qui a pour objectif de décider l'acceptation/rejet d'une demande en calculant la possibilité de la garantie de QoS en fonction des ressources encore disponibles. Du point de vue de la recherche, toute la complexité se situe dans les méthodes de calcul de garantie dans le contrôle d'admission. Pour la garantie déterministe, la plupart des méthodes sont basées sur l'approche de « Network calculus » due à sa simplicité [Cruz91], [LeBoudec02], tandis que pour la garantie probabiliste les méthodes se basent plutôt sur l'approche stochastique [Knightly99] ou sur l'approche dite « statistical network calculus » [Chang00], [Burchard00], [Burchard01], [Vojnovic01], [Varsou01].

Comme nous avons déjà expliqué dans l'introduction, le problème de la garantie déterministe est qu'elle nous oblige de considérer le pire cas conduisant souvent à un surdimensionnement exagéré du système support, au point qu'aucun nœud/réseau existant ne pourra la satisfaire (ou la satisfait avec un coût non raisonnable). Une garantie probabiliste permet effectivement de résoudre ce problème de surdimensionnement. Mais bon nombre d'applications ne peuvent pas accepter une garantie de QoS probabiliste car elle ne fournit aucun moyen de préciser comment l'application tolère le non-respect des contraintes. Par exemple « jamais plus de 3 pertes sur 10 paquets consécutifs » ne sera pas satisfait par la garantie de probabilité de 0,7. Ce qui explique pourquoi nous sommes intéressés par le concept de la garantie réglable de QoS.

Nous empruntons la notion de **dégradation gracieuse** (Graceful Degradation [Herlihy91]) dans les systèmes tolérant aux fautes pour le contrôle de la QoS pour des applications sous contrainte (m,k)-firm. L'idée est de définir le niveau maximum de QoS lorsque le système peut supporter (k,k)-firm ; Le niveau minimum de QoS lorsque le système ne peut garantir que (m,k)-firm. Toutes les autres valeurs entre m et k peuvent être considérées comme niveaux de QoS intermédiaires. Ce concept permet d'une part de donner la priorité en cas de surcharge momentanée du réseau aux paquets plus importants en écartant les moins importants, résultant ainsi en un réseau auto-adaptatif, et d'autre part d'améliorer le taux d'utilisation des ressources réseaux par rapport au dimensionnement classique selon l'hypothèse du pire cas (garantie déterministe du temps réel dur).

Par rapport au *taux de perte* (loss-rate) qui est l'un des paramètres de QoS classiques, le modèle de QoS selon (m,k)-firm permet de ne mesurer la QoS que pendant une durée limitée (une fenêtre glissante de k paquets) et de préciser comment ces pertes sont distribuées. Par exemple, des pertes consécutives des paquets audio (écartées car leurs échéances ne sont pas respectées) peuvent dégrader considérablement la qualité auditive. Alors des pertes espacées de façon adéquate peuvent être reconstituées par des techniques d'interpolation [Cho94]. Un autre avantage de ce modèle de (m,k)-firm est lié au fait que son implémentation en-ligne est naturelle (avec DBP par exemple) facilitant ainsi la gestion dynamique (en-ligne) de la QoS. Nous discutons par la suite sur l'implémentation de ce modèle dans le cadre de IntServ et DiffServ de l'IETF pour la garantie de la QoS des flux temps réel.

### 5.1. Spécification de flux temps réel selon (m,k)-firm

Le concept du  $\kappa$ -pattern fixe comme le profil de perte autorisé suivant la contrainte définie par le couple  $(m,k)$  peut être directement appliquée au contrôle de la dégradation gracieuse de la QoS des sources vidéo et audio.

Le  $\kappa$ -pattern d'un flux ayant une contrainte temporelle (m,k)-firm est défini par la succession de  $k$  éléments de l'alphabet  $\{O, C\}$  tel que :

$\left\{ \begin{array}{l} 'O' \text{ indique un paquet optionnel} \\ 'C' \text{ indique un paquet critique} \end{array} \right.$

On représente par  $\kappa(i)$  le  $i^{\text{ème}}$  élément du  $\kappa$ -pattern pour  $1 \leq i \leq k$ .

En utilisant le concept du  $\kappa$ -pattern, on divise les paquets d'un flux en deux classes : optionnelle et critique. Avec cette classification, il suffit de transmettre avec succès les paquets critiques pour satisfaire cette contrainte. Cependant, il reste toujours la possibilité de transmettre des paquets optionnels sans violer cette contrainte. En général, les paquets optionnels peuvent être transmis ou rejetés suivant les besoins de l'algorithme d'ordonnancement.

Par exemple, un flux ayant des contraintes de type (3,5) peut satisfaire plusieurs  $\kappa$ -patterns (10110, 01011, 11010, ...) de longueur 5 et ayant exactement 3 paquets critiques.

De ce fait, le  $n^{\text{ème}}$  paquet d'un flux ayant des contraintes temporelles (m,k)-firm, est considéré comme étant un paquet critique si et seulement s'il satisfait la relation suivante :

$$\kappa(n \% k) = 'C' \quad (8)$$

avec  $n \% k$  le reste de la division de  $n$  modulo  $k$ .

L'application de cette classification peut être utile dans le domaine multimédia. En effet, ce concept peut être appliqué à un flux vidéo pour sélectionner les trames critiques dans un GOP (Group of Picture) en utilisant le standard de compression MPEG [Furht99]. Par exemple, un flux compressé utilisant la structure du GOP suivante [IBBPBBPBB] peut être considéré comme étant un flux ayant des contraintes temporelles de type (5,8) affecté au  $\kappa$ -pattern suivant 'COOCOOCOO'. Ceci dit que toutes les trames de type B sont considérées comme optionnelles. Par exemple, la 226<sup>ème</sup> trame est considérée comme étant critique car  $\mu(226 \% 9) = \mu(1) = 'C'$  et la 227<sup>ème</sup> trame est considérée comme étant optionnelle car  $\mu(227 \% 9) = \mu(2) = 'O'$

Pour une application audio par exemple permettant de tolérer 2 échéances consécutives, le  $\kappa$ -pattern peut prendre la forme suivante 'COOC' pour exprimer ce besoin.

Nous précisons que cette technique de classification est statique et initiée par l'application ou l'utilisateur. Cependant, nous pouvons imaginer une classification réalisée dynamiquement par l'ordonnancement suivant un algorithme donné tel que DBP. Ceci peut être avantageux pour optimiser les rejets des paquets, mais il nécessite un traitement additionnel par le serveur.

### 5.2. (m,k)-firm dans IntServ

Le modèle IntServ a été conçu par l'IETF pour répondre au besoin d'adapter les services fournis par le réseau Internet aux exigences des applications temps-réel notamment les flux multimédias, les communications interactive (VoiceIP) et diffusion télévisée en temps-réel. Il propose deux types de services : *Contrôle de charge* et *Garanti*.

Cependant, il s'avère que les flux ayant besoin d'un support spécifique pour leur transmission sur le réseau n'ont pas des exigences aussi dures que ce qui sont prévues par le service garanti ni aussi souples que celles du service à contrôle de charge. En effet, les flux multimédias comme un exemple typique requièrent une qualité de service temporelle spécifique du type (m,k)-firm (cf. section 5.1). Il serait

intéressant d'intégrer dans le modèle IntServ le *service à QoS gracieusement dégradable* comme étant un niveau intermédiaire entre le service garanti et le service à contrôle de charge.

[Koubâa03b] a développé une technique pour intégrer les garanties (m,k)-firm dans l'architecture de QoS de IntServ. L'idée de base est que chaque source voulant profiter d'une garantie gracieusement dégradable doit marquer ses paquets en tant que optionnel ou critique selon sa contrainte (m,k)-firm et son  $\kappa$ -pattern associé. Les ordonnanceurs qui garantissent le débit dans le cadre du service garanti, doivent tenir compte de cette classification. Les paquets optionnels dont le deadline a expiré sont rejetés.

Ce service permet alors de garantir des bornes sur le délai plus petit et d'une manière plus flexible. Pour une source ayant un trafic défini par le TSPEC (M,p,b,r) de Intserv avec  $M$  la taille maximal 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 paquet optionnel égal à  $D_{op}$ , selon nos travaux dans [Koubâa03b], le délai maximal  $D_{max}$  est donné par :

$$D_{max} = \max \left[ \frac{M}{R} + \left( \frac{\sigma - M}{R} \right) \left( \frac{p - R}{p - r} \right)^+, \left( \frac{(\lambda_{m,k} M + \lambda_{k-m,k} \sigma)}{R} + \left( \frac{b - M}{R} \right) \left( \frac{\rho - R}{p - r} \right) \right) \right] + T \quad (9)$$

Avec  $\sigma$  la taille maximale de rafale des paquets optionnels susceptible d'être transmise par l'ordonnanceur et défini par  $\sigma = rD_{op}$ .  $\lambda_{m,k}$  désigne le taux du nombre de bits critiques du flux et  $\lambda_{k-m,k}$  est la taux du nombre de bits optionnel du flux. Dans le cas où la taille du paquet est constante  $\lambda_{m,k} = \frac{m}{k}$ . L'applicabilité de cette formule s'étend donc à une taille de paquet variable mais qui doit satisfaire une organisation cyclique correspondant à un  $\kappa$ -pattern dont la valeur de  $\lambda_{m,k}$  est constante dans chaque période du  $\kappa$ -pattern.

Cette borne sur le délai est inférieure au délai maximal garanti par le modèle IntServ [LeBoudec02] et tient compte des contraintes temporelles (m,k)-firm et du profil de perte selon  $\kappa$ -pattern du flux.

### 5.3. (m,k)-firm dans DiffServ

DiffServ est une architecture de QoS qui était développé pour résoudre le problème de scalabilité du modèle IntServ. Contrairement à la garantie par flux de IntServ, DiffServ présente l'avantage de définir des classes de service et de traiter les flux par agrégat. Par conséquent, la garantie dans DiffServ est par classe. Une source ayant des contraintes temps réel particulières se contentera de choisir une classe appropriée immédiatement au-dessus de sa demande. Ce qui résulte en une garantie gros-grain et une sous utilisation des ressources. Le modèle selon (m,k)-firm peut alors permettre à une source de choisir plus judicieusement sa classe parmi celles satisfaisant une QoS entre (m,k)-firm et (k,k)-firm.

Pour la gestion de congestion, des mécanismes de coloriage de paquets ainsi que RED sont utilisés actuellement pour le rejet de paquets en cas de surcharge du réseau. Le principe de marquage en se basant sur les contraintes (m,k)-firm ainsi que le  $\kappa$ -pattern (cf. section 4) peut être appliqué pour faire le rejet sélectif des paquets. En effet, en cas de congestion, il est judicieux de rejeter des paquets optionnels plutôt qu'un rejet aléatoire pour minimiser les risques de violation de la qualité de service pour les flux temps réels. Cette technique de marquage peut aussi être utilisée dans les régulateurs de trafic à l'entrée du réseau qui assureront la conformité du trafic à sa spécification.

[Striegel00] propose aussi d'utiliser les contraintes (m,k)-firm dans les réseaux DiffServ. La proposition consiste à créer des classes de service dynamiquement. Chaque classe contient l'agrégat de flux ayant des contraintes temporelles (m,k)-firm similaires. Les algorithmes d'ordonnancement tel que DBP et DWCS sont alors appliqués à ces classes pour assurer un service en fonction des contraintes temporelles (m,k)-

firm. L'avantage de cette technique est d'assurer le compromis entre la scalabilité et granularité de la qualité de service en limitant le nombre de files d'attente à gérer par serveur (routeur ou commutateur). Cependant, l'inconvénient de cette approche est que le respect de la contrainte temporelle par classe de service n'implique pas la satisfaction de cette contrainte pour chacun des flux de cette classe. Ce fait dépend essentiellement du degré de granularité choisi par le système.

Comment trouver un bon compromis entre la QoS par classe et la QoS par flux (source) en utilisant le modèle (m,k)-firm reste encore un sujet ouvert.

## 6. Conclusion et perspectives

Le développement d'applications temps réel utilise de plus en plus l'approche COTS. Garantir le respect des contraintes temps réel avec des ressources limitées (CPU, Middleware, Réseau) et supporter des applications évolutives exigent un système adaptatif et capable de fournir des niveaux de QoS appropriés [ARTIST03]. Les modèles de QoS d'Internet fournissant une garantie déterministe et une garantie probabiliste constituent souvent une source d'inspiration. Mais cette garantie déterministe ne permet pas d'une utilisation efficace de ressources. Et bon nombre d'applications temps réel ne peuvent pas se contenter d'une garantie probabiliste. Dans cette situation le modèle (m,k)-firm ou plus généralement WHRT nous semble être une voie prometteuse. Ce modèle nous permet d'introduire une QoS gracieusement dégradable et les algorithmes d'ordonnancement associés permettent au système de s'adapter dynamiquement aux changements de son état.

La prise en compte de (m,k) comme paramètres supplémentaires exige par contre un effort de recherche et développement des algorithmes d'ordonnancement nouveaux, et par conséquent de prouver les conditions suffisantes d'ordonnancabilité si une garantie déterministe de (m,k)-firm est demandée. Une intégration de ce modèle de QoS gracieusement dégradable avec ses algorithmes d'ordonnancement dans des OS [West02], Middlewares et Réseaux [Wang02] constitue en un autre challenge.

## 7. Bibliographie

- [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.
- [Baruah98] Baruah, S. and Lin, S.S., "Pfair scheduling of generalized Pinwheel task systems", *IEEE Trans. on Computers*, 47(7), pp812-816, July 1998.
- [Bernat01] Bernat, G., A. Burns and A. Llamosi, "Weakly-hard real-time systems", *IEEE Transactions on Computers*, 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.
- [Burchard00] Burchard, A., R. Boorstyn, J. Liebeherr, and C. Oottamakorn, "Statistical service assurances for traffic scheduling algorithms", *IEEE Journal for Selected Areas in Communications* 18 (2000), 2651-2664.
- [Burchard01] Burchard, A., J. Liebeherr, S. Patek, "A Calculus for End-to-end Statistical Service Guarantees", *Technical report: University of Virginia*, CS-2001-19
- [Cervin00] Cervin, A., "Towards the integration of control and real-time scheduling design" *PhD thesis, Lund University, Sweden*, [www.control.lth.se/~anton](http://www.control.lth.se/~anton), May 2000.
- [Chan92] Chan, M.Y. and Chin, F.Y.L., "General schedulers for the Pinwheel problem based on double-integer reduction", *IEEE Trans. on Computers*, 41(6):755-768, June 1992.
- [Chang00] Chang, C. S., *Performance Guarantees in Communication Networks*. New York: Springer-Verlag, 2000.
- [Cho94] Cho, Y.-J. and Un C.-K., « Performance analysis of reconstruction algorithms for packet voice communications », *Computer Networks and ISDN Systems*, Vol.26, pp.1385-1408, 1994.
- [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.
- [Cruz91] Cruz, R. L., "A calculus for network delay, Part I". *IEEE Trans. on IT*, 37(1):114-131, Jan. 1991.
- [Furht99] B. Furht (Editor), *Handbook of multimedia computing*, CRC Press LLC, 1999.

- [Graham76] Graham, R., "Bounds on the performance of scheduling algorithms", Chapter in *Computer and Job Shop Scheduling Theory*, John Wiley and Sons, pp.165-227, 1976.
- [Hamdaoui95] Hamdaoui M. and P. Ramanathan, "A dynamic priority assignment technique for streams with (m, k)-firm deadlines", *IEEE Transactions on Computers*, 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 Computers*, 46(12), pp.1325–1337, Dec.1997.
- [Hammett03] Hammett, R. and P Babcock, "Achieving 10U-U9 Dependability With Drive-By-Wire Systems", *SAE 2003 World Congress & Exhibition*, Detroit (USA), March 2003.
- [Herlihy91] Herlihy, M.P. and Wing, J.M., "Specifying graceful degradation", *IEEE Trans. on Parallel and Distributed Systems*, Vol.2, no.1, pp93-104, 1991.
- [Jeffay91] Jeffay, K., Stanat, D.F. and Martel, C.U., "On Non-Preemptive Scheduling of Periodic and Sporadic Task", *IEEE real-time systems symposium*, pp129-139, San Antonio (USA), Dec. 4-6, 1991.
- [Koren95] Koren G. and D. Shasha, "Skip-over: Algorithms and complexity for overloaded systems that allows skips", *Proceedings of Real-Time Systems Symposium*, pages 110–117, Dec. 1995.
- [Koubâa03] Koubâa, A. and Song, Y.Q., "Evaluation et Amélioration des Bornes du Temps de Réponse pour des Applications Temps Réel avec Ordonnancement à Priorité Fixe et Non-Préemptif", *MSR03*, Metz (France), Oct. 2003.
- [Koubâa03a] Koubâa, A. and Song, Y.Q., " Amélioration des Délais dans les Réseaux à Débits Garantis pour des Flux Temps-Réel Sous Contrainte « (m,k)-Firm »", *SETITE'2003*, Sousse, Tunisie, 17-21 Mars 2003.
- [Koubâa03b] Koubâa, A., Song, Y.Q. and Thomesse J.P., " Integrating (m,k)-Firm Real-Time Guarantees in the IntServ QoS Model", *Rapport intern, LORIA-TRIO*, Avril 2003.
- [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.
- [Lindsay99] Lindsay W. and P. Ramanathan, "DBP-M, A Technique for Meeting end-to-end (m, k)-firm Guarantee requirements in point-to-point networks", *IEEE Conference on Local networks*, pp.294-303, 1999.
- [Poggi03] Poggi, E.-M., Y.-Q. Song, A. Koubaa, Z. Wang, "Matrix-DBP For (m, k)-firm Real-Time Guarantee", *RTS2003*, pp457-482, Paris (France), 1-3 April 2003.
- [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.
- [Ramamritham96] Ramamritham, K., "Where do time constraints come from and where do they go?", *International Journal of Database Management*, 7:2, 1996.
- [Ramanathan99] Ramanathan P., "Overload management in Real-Time control applications using (m, k)-firm guarantee". *IEEE Transactions on Parallel and Distributed Systems*, 10(6):549–559, Jun 1999.
- [Striegel00] Striegel A., G. Manimaran, "Best-effort Scheduling of (m,k)-firm Real-time Streams in Multihop Networks", *International Workshop of Parallel and Distributed Real-Time Systems (WPDRTS2000)*, Cancun (Mexico), May 1-2, 2000.
- [Varsou01] Varsou A., "Rate Scheduling Techniques and Delay Quality of Service Guarantees for Heterogeneous Networks", PhD thesis, Princeton University, 2001
- [Vojnovic01] Vojnovic M. and J.-Y. Le Boudec, "Bounds for independent regulated inputs multiplexed in a service curve network element," in *Proc. GLOBECOM 2001*, San Antonio, TX, Nov. 2001.
- [WangS02] 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.
- [Wang02] Wang Z., Song Y.Q., Poggi E. and Sun Y.X., "Survey of Weakly-Hard Real Time Scheduling Theory and Its Application". *International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES2002)*, 2002.11.
- [West02] West R., and Jason Gloudon, " 'QoS Safe' Kernel Extensions for Real-Time Resource Management", in the *14th EuroMicro International Conference on Real-Time Systems*, June 2002
- [West99] West R. and K. Schawn, "Dynamic window-constrained scheduling for multimedia applications", *6<sup>th</sup> International Conf. On Multimedia Computing and Systems, ICMCS'99*, IEEE, June 2000.
- [Wilwert03] Wilwert, C., Song, Y.Q., Simonot-Lion, F., Clément, T., "Evaluating Quality of Service and Behavioral Reliability of Steer-by-Wire Systems ", *IEEE ETFA '03*, Lisbon (Portugal), 2003.