Garantir la qualité de service temps réel : ordonnancement et gestion de files d'attente

Ye-Qiong Song

Garantir la Qualité de Service Temps Réel: Ordonnancement et Gestion de Files d'Attente

Ye-Qiong Song
LORIA – Nancy Université
Campus Scientifique, BP239
54506 Vandoeuvre-lès-Nancy
song@loria.fr

Résumé

L'objectif de ce document est de présenter la problématique de la garantie de qualité de service temps réel dans les réseaux à commutation de paquets. Le problème central dans ce type de réseaux est la gestion des files d'attente des paquets dans des nœuds de commutation (commutateurs et routeurs). En considérant un modèle général de nœuds de commutation où plusieurs files d'attente partagent un même lien de sortie, nous discutons sur l'ordonnancement de paquets, présentons des techniques d'évaluation de bornes de temps de réponse et indiquons les limites de ces techniques. Pour des applications temps réel souple tolérant le rejet de paquets en cas de surcharge (par exemple la transmission de flux multimédia sur Internet), nous présentons aussi des solutions efficaces basées sur le modèle (m,k)-firm.

1. Introduction

Le déploiement des applications temps réel sur un réseau exige la garantie de temps de réponse des paquets transmis. Selon les applications, cette exigence de garantie peut être probabiliste (transmission multimédia sur Internet par exemple) ou déterministe (contrôle-commande au travers un réseau Ethernet commuté industriel par exemple). Il est clair que seul le service « best effort » n'est plus suffisant. C'est la raison pour laquelle l'Internet d'aujourd'hui intègre une architecture de la qualité de service (QdS) appelé Diffserv. Dans un routeur qui supporte Diffserv, l'ensemble de trafic est traité différemment selon les classes afin de respecter les contraintes diverses des applications.

L'ordonnancement et la gestion des files d'attente des paquets dans un réseau à commutation de paquets (par exemple Ethernet commuté, Internet) sont les deux mécanismes fondamentaux pour fournir la QdS temps réel, qualité sur laquelle repose la sûreté de fonctionnement des applications temps réel distribuées. L'ordonnancement détermine le bon ordre de transmission des paquets afin de respecter leurs contraintes temporelles (délais et gigues), tandis que les politiques de gestion dynamique de files d'attente (par exemple RED, rejet sélectif) gèrent la situation de surcharge momentanée (ou congestion) tout en maintenant le taux de perte de paquets en dessous d'une certaine limite.

L'utilisation des algorithmes d'ordonnancement existants (priorité fixe, WFQ, …) pour garantir de façon déterministe les contraintes temporelles sur la transmission de paquets pose un problème de surdimensionnement de ressources (ou on dit « pessimisme » dans l'établissement de condition d'ordonnançabilité) à cause de l'obligation de considérer le pire cas (e.g., débit crête) lors du dimensionnement. En effet, les applications ne génèrent qu'un trafic selon leur débit moyen laissant le réseau en sous utilisation dans la plupart du temps. Un autre problème est la gestion de la congestion. A cause de la connaissance incomplète de l'ensemble de trafic (par exemple dans Internet au cas où un chemin inclurait un ou plusieurs routeurs ne supportant pas Diffserv), les congestions peuvent apparaître conduisant au délai excessif voire à la perte de paquets suite au débordement de buffers. Bien que des pertes occasionnelles de paquets puissent être tolérées par certaines applications (par exemple téléphonie IP), longues pertes consécutives doivent être évitées car elles dégradent sérieusement la QdS des applications. Malheureusement les mécanismes de gestion de files d'attente existants tels que TD (Tail-Drop) ne traite pas ce problème de perte de paquets consécutifs. Floyd et Jacobson ont proposé RED (Random Early Detection) [1] pour traiter ce problème en rejetant aléatoirement des paquets lorsque la file d'attente dépasse un certain seuil avant son débordement. Mais il est clair qu'un tel mécanisme ne donne aucune garantie sur la non-consécutivité de pertes.

Pour traiter ces deux problèmes, l'idée que nous allons exploiter dans ce document consiste à profiter de la « tolérance naturelle » de perte de paquets d'une large classe d'applications temps réel souple pour réduire le

besoin en réservation de bande passante suffisante et effectuer le rejet sélectif en cas de congestion. Prenons l'exemple de la téléphonie IP, il est préférable de rejeter un paquet quand il ne peut pas être transmis à temps au lieu de continuer à le transmettre avec un délai très large.

Le modèle (m,k)-firm [2] peut être utilisé pour mieux spécifier la distribution de pertes. En effet, un flux de paquets qui respecte la contraintes (m,k)-firm garantie qu'au moins *m* parmi *k* paquets consécutifs quelconques doivent être transmis par le réseau 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 dans la suite. Basé sur le modèle (m,k)-firm nous avons développé le concept de dégradation contrôlée et des mécanismes d'ordonnancement et de gestion dynamique de files d'attente [3], [4], [5]. L'idée de base consiste à spécifier plus précisément les contraintes des applications en définissant des niveaux de dégradation de performances acceptable jusqu'à la limite du tolérable, et à développer des algorithmes d'ordonnancement permettant la garantie déterministe du niveau le plus faible de performances pendant la surcharge du système. Ainsi, le système résultant est adaptatif et robuste car capable de fonctionner correctement entre le mode nominal et les modes dégradés.

Gérer la QdS selon le modèle (m,k)-firm représente au moins deux avantages. Durant la phase de la congestion, les paquets sont rejetés selon le modèle (m,k)-firm et non plus aléatoirement comme par TD et RED. Ce qui permet d'éviter de longues séquences de perte de paquets consécutifs. Intuitivement, le fait de chercher à garantir (m,k)-firm et non plus (k,k)-firm peut exiger moins de ressources car la charge est réduit d'un facteur de m/k. Ce qui est intéressant quand on désire admettre plus de flux que ce qui est autorisé par le contrôle d'admission classique basé sur la réservation selon débit crête de chaque flux (over-provisionning).

En résumant, fournir la garantie de QdS vis à vis des applications temps réel d'exigences différentes et partageant un même réseau nécessite le développement des algorithmes d'ordonnancement adéquats et des mécanismes de gestion de files d'attente efficaces. A cela s'ajoute un souci d'utilisation de ressources. Offrir des garanties déterministes ou probabilistes de QdS tout en maintenant un fort taux d'utilisation des ressources réseaux est un défi. L'utilisation conjointe des compétences de la communauté des réseaux et de celle de l'ordonnancement temps réel devrait contribuer à relever ce défi.

Le reste de ce document est organisé comme suit. La section 2 introduit le modèle de base d'un nœud de commutation où plusieurs files d'attente partagent un même lien de sortie, discute sur les garanties possibles vis à vis des flux d'entrée de natures différentes. En supposant un ordonnancement à priorité fixe, la section 3 présente et compare deux techniques d'analyse de temps de réponse : l'analyse du pire cas de la communauté de l'ordonnancement temps réel et l'analyse de la borne supérieure de délais du « network calculus » de la communauté des réseaux. En se focalisant sur des applications temps réel sous contrainte (m,k)-firm, la section 4 s'intéresse à l'amélioration de l'ordonnancement WFQ quand ce dernier est utilisé pour fournir la garantie de temps de réponse. La section 5 analyse les causes du problème de surdimensionnement (ou de sous utilisation de ressources) et étend le modèle (m,k)-firm en R-(m,k)-firm [5] par la relaxation de la contrainte d'échéance par paquet. La section 6 donne un exemple d'un mécanisme de gestion de files d'attente basé sur le rejet sélectif de paquets pour gérer efficacement la congestion tout en garantissant la contrainte R-(m,k)-firm.

L'objectif de ce document est double. La première partie (sections 2 et 3), en introduisant et comparant les techniques de l'analyse du pire temps de réponse et l'évaluation de la borne supérieure de délais du « network calculus », vise à susciter l'intérêt des chercheurs sur l'utilisation conjointe des techniques des communautés des réseaux et du temps réel lors de la fourniture de la QdS temps réel dans les réseaux. La seconde partie, au travers d'un exemple de l'algorithme d'ordonnancement (m,k)-WFQ [3] et d'un exemple de la gestion dynamique de files d'attente « Double Leaks Bucket » [5], cherche à illustrer les apports possibles de ce croisement des techniques.

2. Modèle, flux d'entrée et types de garantie

Cette section introduit le modèle de base d'un nœud de commutation où plusieurs files d'attente partagent un même lien de sortie, discute sur les garanties possibles vis à vis des flux d'entrée de natures différentes, et décrit le modèle (m,k)-firm ainsi que son utilisation pour spécifier des contraintes temps réel.

2.1. Modèle MIQSS

Que ce soit dans un réseau à un seul saut (un réseau local au médium partagé par exemple) ou dans un grand réseau à sauts multiples (Internet par exemple), le point clé pour le calcul du temps de réponse de bout en bout est l'évaluation du temps de traversé de chaque nœud. Ce nœud peut être le médium de transmission partagé par des stations multiples dont l'accès est régulé par le protocole MAC (Medium Access Control). Ce nœud peut aussi être un multiplexeur, un commutateur ou un routeur dont un même port de sortie est partagé par toutes les lignes en entrée. Un modèle appelé MIQSS (Multiple Input Queues Single Server) peut être utilisé pour décrire un tel système (Figure 1).

Le modèle MIQSS est constitué d'un ensemble de N sources $\{S_i\}_{1 \leq i \leq N}$ de **clients** (qui représentent des

paquets dans un réseau) partageant le serveur unique de capacité de traitement de *c* bit/s. Une source S_i est caractérisée par le flux de clients F_i qu'elle génère et les contraintes de performances (ici on ne considère que celle du temps de réponse) CTR_i qu'elle doit respecter.

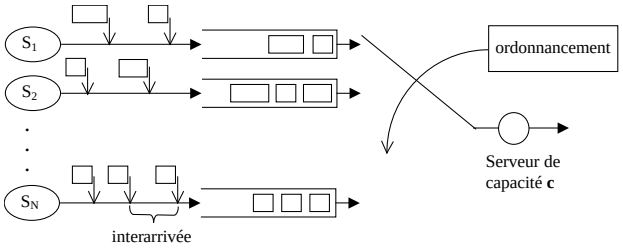

Figure 1 Modèle MIQSS

2.2. Flux d'entrée

La fonction F_i peut être :

- **Périodique ou sporadique**: elle est définie par (C_i, T_i) dans le cas d'une date de génération du premier client quelconque et (r_i, C_i, T_i) dans le cas où la date de génération du premier client notée par r_i est donnée ; C_i est le temps d'exécution d'un client par le serveur et T_i la période d'interarrivée (ou d'interarrivée minimale dans le cas sporadique). Dans la plupart du temps on considère le cas où C_i est constante. Le cas de C_i variable est souvent majoré par le temps d'exécution maximale (WCET : Worst Case Execution Time).
- **Périodique avec gigues** : elle est caractérisée par (C_i, T_i, J_i) ou (r_i, C_i, T_i, J_i) où J_i représente la gigue. La gigue est considérée comme étant le déphasage maximal d'une arrivée par rapport à la période. Dans ce document la gigue a toujours une valeur positive (i.e., un retard par rapport à la période)
- **(σ_i, ρ_i)-borné** : elle est donnée sous forme d'une courbe linéaire caractérisée par la taille de la rafale d'arrivées des clients σ_i et le débit moyen ρ_i qui majore la vraie fonction cumulative d'arrivée du travail. La quantité du travail apportée par un client est définie par W_i avec notamment $C_i = W_i / c$.
- **Aléatoire** en suivant une loi d'arrivée probabiliste avec un temps moyen d'inter-arrivée T_i et un temps moyen d'exécution de clients C_i (par exemple : une loi de Poisson pour décrire une loi d'inter-arrivée exponentielle avec la taille constante des clients, une loi de Poisson composée pour une loi d'inter-arrivées exponentielle et des tailles variables de clients).

Le flux d'arrivée périodique ou sporadique, avec ou sans gigues correspond aux modèles de tâches classiques utilisés par la communauté de l'ordonnancement temps réel [6]. Le flux (σ, ρ)-borné apparaît, quant à lui, plutôt dans la communauté de la QdS des réseaux (ATM et Internet) [7], [8], [9], [10]. Un tel flux est obtenu par un lisseur de trafic (« Leaky bucket » par exemple). Un flux aléatoire est un modèle classique du domaine de l'évaluation de performances, notamment lorsqu'il s'agit de l'évaluation de performances des réseaux par la théorie des files d'attente [11], [12]. Dans la section 3 nous allons montrer que les flux périodiques et sporadiques peuvent être approchés par un modèle de flux (σ_i, ρ_i)-borné. Ceci ouvre la possibilité de choisir entre la théorie de l'ordonnancement et celle du « network calculus » [10] lors de l'évaluation de la borne supérieure des temps de réponse.

2.3. Contraintes temps réel et (m,k)-firm

Les contraintes de temps de réponse CTR_i sont classiquement données par D_i qui est l'échéance relative à l'instant d'arrivée d'un client. Selon la conséquence du non-respect de contraintes temps réel, les applications sont classifiées en temps réel dur et souple. Certaines applications temps réel souple sont aussi qualifiées temps réel *firm* selon le traitement ou non des actions dont le respect de leur échéance est impossible par le système (en cas de surcharge momentanée du système par exemple). Une application ayant des contraintes temps réel souple est dite *firm* si ses actions, dont l'échéance ne pouvant pas être respectée, sont rejetées par le système (non exécutées) car le traitement en retard de ces actions est considéré comme inutile, permettant ainsi de réduire de façon efficace la charge du système. Notons que ce soit pour des applications temps réel dur, souple ou *firm*, la garantie peut être fournie de façon déterministe ou probabiliste.

Une large classe d'applications temps réel *firm* sont des applications multimédia temps réel sur l'internet. Un exemple typique de cette classe d'applications est la transmission des paquets audio et vidéo. Deux caractéristiques communes sont la nécessité de délivrer ces paquets dans un temps borné et la tolérance de perte de paquets, jusqu'à une certaine limite à préciser, sans dégrader notablement la qualité de l'application. En effet, il est même préférable dans certains cas (téléphonie IP par exemple) de rejeter des paquets en retard au niveau des routeurs qu'à continuer à les transmettre car les paquets en retard sont tout simplement inutilisables. Dans ce cas, le modèle (m,k)-firm nous paraît

intéressant pour spécifier plus précisément la tolérance de le dégradation de la QdS en terme du non-respect de l'échéance des paquets. Si l'on considère qu'une application peut accepter une dégradation de service jusqu'à *m* paquets transmis avant l'échéance parmi *k* paquets consécutifs quelconques, un système peut alors être 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 dégradation encore acceptable) 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* »).

2.4. Expression de contraintes temps réel selon le modèle (m,k)-firm

Une source sous contrainte temps réel (m, k)-firm exprime sa contrainte en spécifiant simplement la valeur des deux paramètres : *m* et *k*. Cette source peut se trouver dans l'un des deux états : normal et échec transitoire. 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.

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).

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. Cette technique s'inspire du modèle de calcul imprécis où une tâche est composée d'une partie critique (c'est-à-dire obligatoire) 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. 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 leurs 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 :

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

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. 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. Par exemple, un flux compressé utilisant la structure du GOP suivante [*IPBBPBBPBB*], 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 (4,10)-firm et spécifié par le κ-pattern suivant $\{\pi_{i(1 \le i \le k)}\}$ = {1100100100}. Ce κ-pattern marque les paquets des trames I et P comme critiques et les paquets des trames B comme optionnels, en se basant sur le critère d'importance relative des trames MPEG. Il convient de noter que le choix du meilleur marquage dépend de la nature des sources de flux. L'étude plus détaillée sur ce sujet sort du cadre de ce document. Par conséquent, en cas de surcharge où il est impossible d'assurer les échéances de tous les paquets d'un flux MPEG, une des possibilités pour limiter la dégradation de la QdS est de garantir la livraison à temps des trames I et P, si possible, au détriment des trames B. En spécifiant pour chaque flux sa contrainte (m,k)-firm, le respect de *m* parmi une fenêtre de *k* paquets consécutifs permettrait de maintenir une QdS acceptable en évitant de rater les échéances consécutives. En général, le rejet des paquets adéquats selon un profil de perte autorisé bien défini, permet de réduire la charge du système et résulte en une transmission accélérée des paquets en attente.

3. Techniques d'analyse de temps de réponse

Supposons que nous nous intéressons à fournir la garantie déterministe de temps de réponse dans le modèle MIQSS avec l'ordonnancement à priorité fixe. Il est clair que cela n'est possible que lorsque le flux d'entrée soit borné. Nous allons présenter dans cette section deux approches développées séparément par la communauté du temps réel et la communauté des réseaux. La première, appelée WCRTA (« Worst Case Response Time Analysis »), repose sur la trajectoire du pire cas, et la seconde, appelé NC (« Network Calculus »), sur une courbe linéaire caractérisée par la taille de la rafale (notée par σ) et le débit moyen (noté par ρ) d'une source, courbe qui majore la vraie trajectoire. Nous allons examiner dans ce paragraphe ces deux approches afin de comprendre leur différence et complémentarité.

3.1. Modèle et définitions

Nous considérons un modèle MIQSS (Figure 1) constitué d'un ensemble de N sources. A chaque client de source S_i est affecté une priorité i. Un client de S_i est plus prioritaire que celui de S_j si et seulement si $i<j$.

Une source périodique S_i, avec ou sans gigue, est caractérisée par une période T_i, une durée de traitement dans le serveur C_i et une gigue J_i sous contrainte d'échéance D_i. La gigue est considérée comme étant le déphasage maximal d'une arrivée par rapport à la période. Quand la source S_i est strictement périodique, $J_i = 0$.

Une source (σ_i, ρ_i)-bornée S_i est caractérisée par une rafale maximale σ_i et un débit moyen ρ_i sous contrainte d'échéance D_i. La quantité du travail apportée par un client est définie par W_i avec notamment $C_i = W_i / c$.

Notons que seul le cas non-préemptif nous intéresse car la transmission d'un paquet est non-préemptive.

Un flux d'arrivée périodique sans gigues ou avec gigues F(t) peut aussi être borné par le couple de paramètres (σ, ρ) comme ce que la figure 2 illustre.

Cette figure présente un exemple de deux courbes de fonction d'arrivées cumulative de travail (par clients de taille 1) : une est périodique et l'autre est périodique avec gigue, ainsi que leur bornes respectives.

Dans le modèle MIQSS à priorité fixe sans préemption, que ce soit par l'approche NC ou WCRTA, l'évaluation de borne du temps de réponse d'un client de priorité i (de source S_i) consiste toujours à trouver le temps maximal d'attente dû à : 1) l'occupation du serveur par des clients plus prioritaires (de priorités de *1* à *i-1*) ; 2) l'occupation du serveur par un client moins prioritaire car non préemptif.

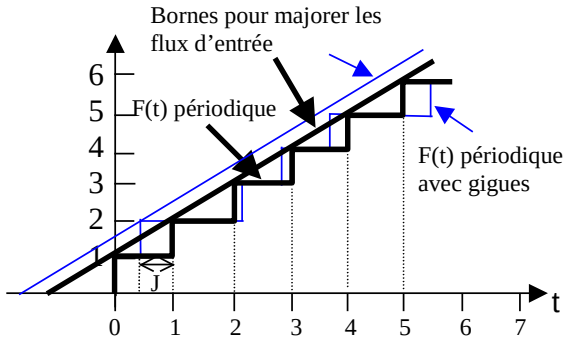

Figure 2 Flux d'arrivée périodique sans gigues et avec gigues et bornes (σ, ρ)

L'approche NC considère que l'occupation du serveur par des clients plus prioritaires correspond au temps de service d'un flux $(\sum_{j=1}^{i-1}\sigma_j , \sum_{j=1}^{i-1}\rho_j)$-borné. L'occupation du serveur par un client moins prioritaire à cause de la non préemption est représenté par le temps de service nécessaire du client le plus grand parmi les clients moins prioritaires ($max_{i+1\le j\le N}(W_j)$).

L'approche WCRTA est plus fine. Au lieu de considérer une simple superposition des fonctions de flux d'arrivée linéaire (σ, ρ), WCRTA considère la pire superposition des fonctions de flux en escalier (peut être obtenue en décalant ces fonctions sur l'axe du temps). L'occupation du serveur par des clients plus prioritaires correspond au temps de service de cette pire superposition de flux. L'occupation du serveur par un client moins prioritaire à cause de la non préemption est représentée par le temps de service nécessaire du client le plus grand parmi les clients moins prioritaires ($max_{i+1\le j\le N}(C_j)$).

3.2. Pire temps de réponse d'un système non préemptif à priorité fixe

3.2.1. Définition du pire temps de réponse

En premier, nous définissons le pire temps de réponse R_{max} comme étant la plus longue durée entre **l'instant d'arrivée** (où le client est généré par la source) d'un client et l'instant le plus tard possible de

fin de service (cf. 3). Cette définition du temps de réponse est aussi celle utilisée dans le modèle (σ,ρ)-borné. Ce qui permet d'une comparaison ultérieure.

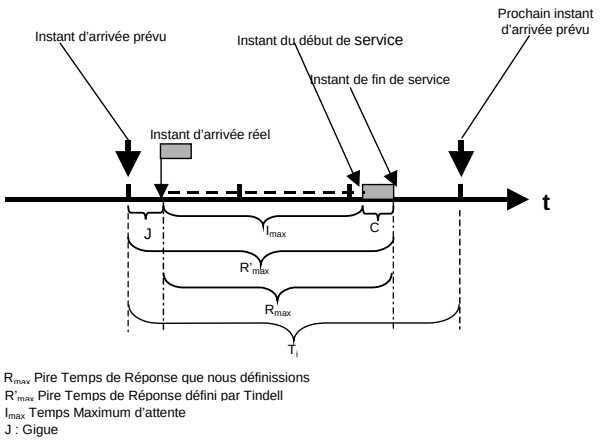

Figure 3 Définition du pire temps de réponse

Le pire temps de réponse R'_{max} défini dans [13] est la plus longue durée entre l'instant d'arrivée prévu d'un client et l'instant le plus tard possible de fin de service. L'instant d'arrivée prévu est le temps où l'arrivée devrait avoir lieu quand les inter-arrivées sont strictement périodiques sans perturbations de gigues.

Nous remarquons que le pire temps de réponse que nous considérons est légèrement différent de celui défini dans [13], qui ne répond pas directement à notre besoin dans cette étude car il n'est pas compatible avec la définition de la borne sur le délai du modèle (σ,ρ)-borné. Nous donnons dans le paragraphe suivant l'équation permettant d'obtenir le pire temps de réponse correspondant à cette définition. Notons que le terme « temps de réponse » et le terme « délai » désignent la même chose dans la suite.

3.2.2. Analyse du pire temps de réponse

Quelques rappels

L'approche de l'analyse du pire temps de réponse WCRTA pour un ensemble de tâches périodiques et préemptibles à priorité fixe a été introduite par Liu et Layland [6] et étendue par [14, 15, 16, 17, 13]. Joseph et Pandya [14] dérivent une expression récursive pour le calcul du pire temps de réponse pour un ensemble de tâches périodiques (ou sporadiques). [15] a étendu les résultats pour un ensemble de tâches périodiques ayant des échéances arbitraires (supérieures à la période) en prenant le maximum des temps de réponse calculés sur une période occupée. [16], [17] et [13] ont généralisé le calcul du pire temps de réponse. Leur méthode permet de prendre en compte la notion de gigue qui affecte la périodicité et le facteur de blocage (une tâche peut être bloquée par une autre de priorité inférieure). Quant aux résultats spécifiques à la non-préemption, on peut remarquer qu'il n'existe que très peu de travaux [18], [19]. Néanmoins les résultats dans [16], [17] et [13] conçus pour l'ordonnancement préemptif peuvent être aisément adaptés à l'ordonnancement non préemptif. Il suffit de considérer le facteur de blocage comme dû à la non-préemption et examiner toutes les périodes d'occupation durant $r + 2P$ (où r est le dernier instant de génération de la première tâche d'une source et P le PPCM des périodes des sources en présence). Le fait d'être obligé de considérer la macro période de $r + 2P$ [19] est dû à la non-préemption car le pire cas défini dans [15] pour des sources périodiques péemptives n'est plus vrai. En effet, lors du calcul du pire temps de réponse, l'unique différence provient du décalage pouvant être engendré par un client moins prioritaire, occupant le serveur à l'instant critique. Les clients ne pouvant être préemptés, l'instant critique est décalé d'au maximum la durée de service du client le plus long ($max_{i+1\leq j\leq N}(C_j)$), parmi les clients moins prioritaires. Le pire cas survient alors lorsque ce dernier débute son service une unité de temps avant l'instant critique, c'est à dire l'instant -1 en imposant que l'instant critique survienne à la date zéro (changement de l'origine des temps) [20]. Pour un état de l'art plus complet sur le calcul des pires temps de réponse dans un modèle MIQSS à priorité fixe et à priorité dynamique (e.g. EDF), avec et sans préemption, nous conseillons la lecture de [20].

Formellement, le pire temps de réponse d'un client de priorité i (ou de source i) dans un système à priorité fixe et sans préemption, est donné par :

$$I_i^{n+1} = \max_{i+1\leq j\leq N}(C_j) + \sum_{j=1}^{i-1}\left(\left\lfloor \frac{I_i^n + J_j}{T_j} \right\rfloor + 1\right)C_j \qquad (1)$$

$$R'_i = C_i + I_i + J_i \qquad \text{si } R'_i \leq T_i \qquad (2)$$

Où $\max_{i+1\leq j\leq N}(C_j)$ est le facteur de blocage dû à la non-préemption. Compte tenu de la définition de [13], l'équation 2 prend en compte la valeur de la gigue. Par contre R'_i ne correspond pas à notre définition du

temps de réponse R_i (cf. 3). Il faut alors enlever la gigue dans l'équation 2 pour l'adapter à notre cas.

$$R_i = C_i + I_i \qquad \text{si } J_i \in [0, T_i[\tag{3}$$

Cette équation calcule le temps de réponse pour le premier client en tête de la file d'attente de la i^{ème} source.

Si la valeur retournée de R_i par l'équation 3 est supérieure à la période du client, le pire temps de réponse peut se produire pour les prochains clients car au moins le prochain sera gêné par ce présent qui est de la même priorité [21]. En incluant la gigue dans [21] ou en prenant les résultats dans [13], l'équation générale du pire temps de réponse du k^{ième} client pour $J_i \in [0, T_i[$ s'exprime par :

$$I_{i,k}^{n+1} = \max_{i+1 \le j \le N}(C_j) + (k-1) * C_i + \sum_{j=1}^{i-1}\left(\left\lfloor \frac{I_i^n + J_j}{T_j} \right\rfloor + 1 \right) C_j \tag{4}$$

$$E_{i,k} = \left\{ C_i + I_{i,k} \right\} \tag{5}$$

$$R_i = \underset{k}{Max}\left(E_{i,k} - (k-1) \cdot T_i \right) \tag{6}$$

Où $E_{i,k}$ est l'instant de fin de service du k^{ième} client de la période occupée. L'itération s'arrête pour la première valeur de $E_{i,k} < T_i$ car cela marque la fin de la période occupée. Compte tenu du résultat dans [19], si une période occupée est inférieure à *r+2P*, nous prenons le soin d'examiner toutes les périodes occupées durant r + 2P afin de déterminer le pire temps de réponse. L'équation 4 donnant $I_{i,k}$ converge si et seulement si la charge générée par les sources des *i* premières priorités est inférieure ou égale à *1*.

La prise en compte du cas de plusieurs sources par priorité

Bien que dans [13, page26] il soit remarqué que ces équations restent encore valables même si la gigue est supérieure à la période, dans notre cas, le domaine de validité des équations 4 à 6 se limite bien au cas des gigues inférieures aux périodes. En effet, avec des gigues supérieures à la période, les clients ne suivent plus nécessairement l'ordre d'arrivées prévu (équivalent à l'ordre de leur génération à la source sans gigue) comme illustré par le cas 3 dans la Figure 4.

Cette figure montre qu'une source génère des clients dans l'ordre (1, 2, 3). Mais ces clients ont subi des délais divers (gigues) avant arriver dans la file d'attente pour être servis. On voit bien que dans le cas 3, le 3ème client est arrivé avant le 2ème. Considérer cet *ordre d'arrivée* ou simplement *l'ordre de génération* périodique de clients (cas 1) change la façon de calculer le pire temps de réponse

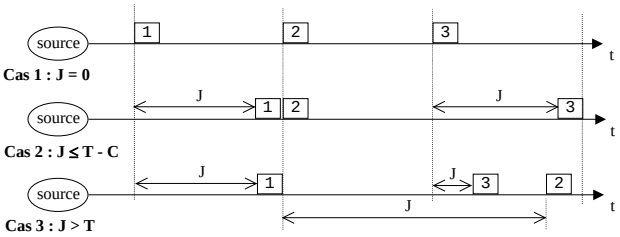

Figure 4 Ordre d'arrivée des clients avec des gigues différentes

Dans le cas considéré dans [13, page19], l'ordre de traitement de clients par le serveur est l'ordre de leur génération à la source. Pour ce faire, on peut imaginer de donner des priorités décroissantes aux clients générés, garantissant ainsi l'ordre de génération pendant la période où plusieurs clients de la même source sont en attente de service dans la file d'attente. Un client venant d'une source de priorité *i* ne sera donc jamais bloqué par ceux qui vont arriver après (dans le cas sans gigue) de la même source et qui ont la priorité *i*.

Dans notre cas, comme on s'intéresse à *l'ordonnancement de paquets dans un réseau*, un nœud intermédiaire d'un réseau de commutation de paquets n'aura pas la connaissance de cet ordre logique à la source et ne peut donc considérer que l'ordre d'arrivée des clients (qu'il soit identique ou différent de l'ordre du départ à la source). Il faut alors prendre en compte en plus, dans le pire cas, le blocage du service d'un client de priorité *i* par tous les autres clients de la même priorité qui risquent d'arriver avant le client considéré à cause de la gigue.

Nous proposons donc une extension pour estimer le pire temps de réponse pour des valeurs de gigues importantes dans le cas où l'ordre d'arrivé est considéré à la place de l'ordre de génération [13, page19]. Pour une gigue J_i telle que $mT_i \le J_i < (m+1)T_i$, alors dans le pire cas, on peut avoir jusqu'à *m* clients de la même source qui bloquent le client courant. Nous ajoutons dans l'expression du pire temps de réponse un délai supplémentaire comme étant un *facteur de blocage* additionnel. Pour le premier client d'une période occupée, le pire temps de réponse est exprimé par :

$$I_{i,k}^{n+1} = \left\{ \max_{i+1 \le j \le N} (C_j) + mC_i \right\} + \sum_{j=1}^{i-1} \left(\left\lfloor \frac{I_i^n + J_j}{T_j} \right\rfloor + 1 \right) C_j \quad (7)$$

La même procédure que précédemment pour les équations 4, 5 et 6 reste valable dans ce cas pour calculer les pires temps de réponse des prochaines itérations si nécessaire. L'état stationnaire est atteint pour une valeur de $E_{i,k}$ inférieure à la période. L'équation générale du temps de réponse du $k^{ième}$ client de la $i^{ème}$ source pour une gigue $J_i \in [mT_i, (m+1)T_i[$ s'exprime par :

$$I_{i,k}^{n+1} = \left\{ \max_{i+1 \le j \le N} (C_j) + mC_i \right\} + (k-1)C_i + \sum_{j=1}^{i-1} \left(\left\lfloor \frac{I_i^n + J_j}{T_j} \right\rfloor + 1 \right) C_j \quad (8)$$

$$E_{i,k} = \left\{ C_i + I_{i,k} \right\} \quad (9)$$

$$R_i = \underset{k}{Max} \left(E_{i,k} - (k-1) \cdot T_i \right) \quad (10)$$

Ce résultat est très important pour analyser les pires temps de réponse pour des topologies complexes et des flux à gigues importantes dans un réseau.

3.3. Bornes supérieures de délais du « Network Calculus »

Partant du fait que les réseaux implémentent des limiteurs du trafic en entrée, par exemple suivant la stratégie de « leaky bucket » pour fournir une garantie de temps de réponse, Cruz [7] a proposé le modèle de trafic (σ,ρ)-borné et a donné une borne supérieure sur le délai introduit par un élément du réseau. Enfin il a déduit dans [8] le délai de bout en bout dans un réseau. Une autre extension de ce modèle est réalisée dans [9] et [10] qui fournit une représentation mathématique de ce modèle par transposition dans l'algèbre (min,+) et (max,+). Les résultats sont pratiquement les mêmes mais cette représentation algébrique est plus convenable pour des calculs numériques. Ce modèle est très utilisé dans la communauté Internet pour le contrôle de la QdS car il est simple à implémenter dans des mécanismes de contrôle d'admission pour estimer le délai.

Cette approche considère des flux ayant des fonctions cumulatives d'arrivées $F_i(t)$ qui sont (σ,ρ)-bornées. Ceci se traduit par la relation suivante :

$$F_i(t) - F_i(s) \le \sigma_i + \rho_i (t-s) \quad \forall 0 \le s \le t \quad (11)$$

σ_i représente la taille de la rafale maximale et ρ_i représente le débit moyen à long terme de la $i^{ème}$ source.

Considérons un ensemble de N sources de clients, tel que chaque source i est (σ_i , ρ_i)-borné, non préemptive et servie selon la politique à priorité fixe. Rappelons que le client de la source S_i est plus prioritaire que le client de S_j si et seulement si $i<j$.

La borne du temps de réponse d'un client de la i^{ème} source est [7] :

$$D_{i\max} = \frac{\sum_{j=1}^{i} \sigma_j + \max_{i+1 \le j \le N} (W_j)}{c - \sum_{j=1}^{i-1} \rho_j} \quad (12)$$

Pour la condition de stabilité $c \ge \sum_{i=1}^{N} \rho_i$.

Où c est la vitesse de traitement du serveur (bit/s) et $W_i = c.C_i$ la taille d'un client (en bits).

3.4. Relation entre la gigue et la taille de la rafale

Dans le cadre du trafic périodique avec gigues la valeur du débit moyen à long terme est facile à déterminer. Pour une source ayant une période T_i et une quantité de travail par client W_i, le débit moyen est :

$$\rho_i = \frac{W_i}{T_i} \quad (13)$$

Le problème principal reste donc à déterminer la taille optimale de la rafale. Elle est définie comme étant la plus petite valeur de σ_i qui satisfait la contrainte (11). En général la rafale existe quand le trafic est perturbé par une variation de gigue. On appelle pente optimale, la courbe (σ+ρt) avec σ la taille optimale de la rafale. Naturellement, il existe une relation entre la taille de la rafale et la gigue.

Le problème est expliqué dans Figures 5 et 6.

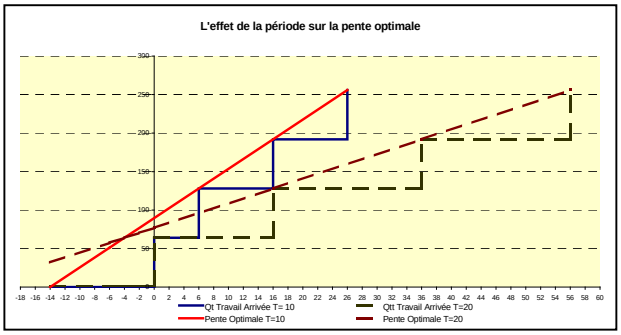

Figure 5 Effet de la période sur la pente optimale

La Figure 5 trace l'arrivée du travail cumulatif pour *T = 10* ou *20*, *J = 4*, *W = 64* octets.

Dans ce premier scénario, on fait varier la période pour voir son impact sur le modèle (σ,ρ)-borné. Le deuxième scénario, montré par la Figure 6, est tracé pour une valeur fixe de la période mais pour deux valeurs différentes de la gigue.

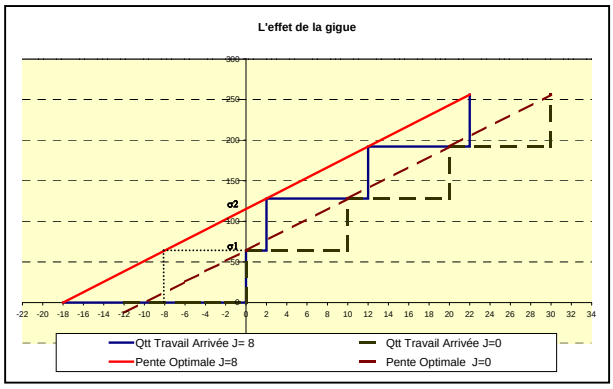

Figure 6 Effet de la période sur la taille optimale de la rafale

La Figure 5 montre que la variation de la période affecte beaucoup plus la pente que la taille de la rafale. Pour la même valeur de gigue, la taille de la rafale est pratiquement la même pour les deux configurations. Ce comportement est normal car la valeur de ρ dépend essentiellement de la période.

Maintenant à partir de la Figure 6, il est facile de remarquer l'effet de la gigue sur la taille optimale de la rafale. Ce graphe est tracé pour *T = 10*, *J = 0 et 8*, *W = 64* octets.

Nous remarquons que l'inclinaison de la pente est la même dans les deux cas. Cependant, la taille de la rafale dépend fortement des valeurs de la gigue. En considérant le pire cas qui correspond à une activation qui se déclenche *J* unités de temps avant l'instant d'activation normal, nous avons montré dans [3] que la taille optimale de la rafale (σ minimale qui majore la courbe d'arrivée) est donnée par :

$$\sigma = \frac{W}{T}(T + J) \qquad (14)$$

Cette méthode d'approximation optimale nous permet alors de comparer les deux approches analytiques. Dans [3] nous avons appliqué cette relation sur plusieurs scénarios qui montrent que l'approche WCRTA fournit des bornes plus petites ou égales à celles fournies par l'approche NC. Ce fait semble logique car en effet, l'estimation de la borne du temps de réponse par le modèle (σ,ρ)-borné est statique. Ceci signifie que nous supposons la connaissance de toutes les tailles maximales des rafales pour déduire statiquement par l'équation (12) le délai maximum, qui est une valeur surestimée du temps de réponse. Par contre, les bornes de temps de réponse calculées par l'approche WCRTA, qui calcule itérativement à travers les équations (1), (4), et (8) les interactions entre les clients de plus hautes priorités, sont plus réalistes que celles fournies par NC.

La méthode NC permet d'évaluer la borne du temps de réponse avec une complexité réduite par rapport à la méthode WCRTA car la fonction majorante de flux d'arrivée est une courbe linéaire à deux paramètres (σ, ρ). Mais la borne de NC est plus large que celle de WCRTA. Cela peut devenir gênant lors du calcul de la borne du temps de réponse de bout en bout dans un réseau multi-sauts. Dans [3] nous avons proposé de remplacer la valeur σ de NC à la sortie de chaque nœud (saut) par celle de WCRTA afin de réduire la taille de la rafale entrant dans le nœud suivant, réduisant ainsi la borne du temps de réponse global de bout en bout.

3.5. Application à un commutateur Ethernet

Ce paragraphe présente l'application des résultats précédents à l'évaluation des temps de réponse dans le réseau Ethernet commuté.

Pour appliquer l'approche WCRTA, une agrégation de priorités utilisateurs en seulement 8 niveaux de priorité IEEE802.1p est nécessaire.

Ethernet commuté (ou Ethernet industriel) est considéré comme une infrastructure fédératrice capable de supporter la communication temps réel grâce à la commutation [22] et apparaît comme une alternative intéressante à des solutions basées sur les bus de terrain. Par rapport à un réseau Ethernet partagé, un réseau Ethernet commuté transforme le problème de collisions en un problème d'attente de la disponibilité des lignes de sortie dans les files d'attente des commutateurs. Les paquets (trames Ethernet) doivent être transmis dans l'ordre de leur priorité si l'on veut gérer des paquets sous contraintes temps réel.

Plaçons-nous dans un cadre général en considérant le modèle MIQSS suivant (Figure 7)

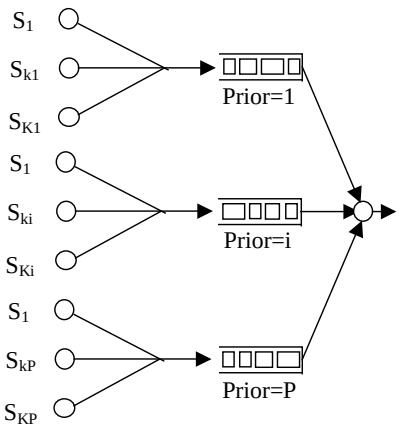

Figure 7 Modèle MIQSS avec K_i sources par priorité

Ce modèle peut correspondre à un commutateur Ethernet qui ne peut gérer qu'un nombre limité de priorités *P (≤8 selon IEEE802.1p)*, alors que le nombre de priorités utilisatrices (ou le nombre de sources) *N* est supérieur à *P*. On regroupe alors ces *N* sources en *P* priorités avec $N = K_1 + K_2 + \ldots + K_P$ où K_i représente le nombre de sources par priorité *i*. Si l'ordonnancement pour accéder au serveur est avec priorité fixe sans préemption, nous pouvons appliquer la technique de WCRTA pour l'obtention du pire temps de réponse d'un paquet d'une source de priorité *i*.

Dans le cas où $K_i = 1$ (une source par priorité) et l'échéance ≤ période, le pire cas pour une source de priorité *i* correspond à la situation où toutes les autres priorités supérieures sont présentes et doivent être servie avant, plus le service en cours d'une priorité inférieure, les équations 1, 2 ou 3 s'appliquent directement.

Dans le cas où $K_i = 1$ (une source par priorité) et l'échéance quelconque, les équations 4, 5 et 6 s'appliquent directement.

Le cas $K_i \geq 1$ avec échéance quelconque correspond au cas traité par les équations 8, 9 et 10. L'idée clé est de considérer l'interférence due aux clients des autres sources de même priorité comme des clients qui ont subi des gigues plus grandes que les périodes. Les expressions 8, 9 et 10 sont alors directement applicables.

Pour plus de détail, voir [23].

L'application de l'approche du NC est directe avec l'équation 3-12.

Pour un réseau composé de commutateurs en cascade, des résultats existent dans [24, 3, 25].

4. (m,k)-WFQ

L'ordonnanceur WFQ (*Weighted Fair Queueing*) [26] 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 [3] 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.

4.1. Algorithme WFQ et ses limites

WFQ garantit à chaque flux servi la proportion de la bande passante réservée selon son coefficient de partage Φ_i. Chaque paquet est estampillé par un *tag* appelé date virtuelle de fin de service. Le serveur sélectionne toujours le paquet dont la date virtuelle de fin de service est le premier à partir de l'instant de sélection. Dans WFQ la date virtuelle de fin de service est définie par :

$$F_i^k = \max\left\{F_i^{k-1}, V\left(t_i^k\right)\right\} + \frac{L_i^k}{\phi_i} \qquad (15)$$

avec :

- t_i^k : la date réelle d'arrivée du $k^{ième}$ paquet du $i^{ème}$ flux,
- F_i^k : la date virtuelle de fin de service du $k^{ième}$ paquet du $i^{ème}$ flux, avec $F_i^0 = 0$,
- $V(t_i^k)$: la date virtuelle quand le $k^{ième}$ paquet arrive, avec *V(0) = 0*,

- Φ_i : le coefficient de partage du $i^{ème}$ flux,

- L_i^k : la taille du $k^{ième}$ paquet du $i^{ème}$ flux,

- $\max\left\{F_i^{k-1}, V\left(t_i^k\right)\right\}$: la date virtuelle du début de service du $k^{ième}$ paquet.

Avec WFQ, il est montré (voir [26] ou [10]) que pour un flux S_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 :

$$D_{i,\max} = \frac{\sigma_i}{g_i} + \frac{L_{\max}}{c}. \qquad (16)$$

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 (bit/s).

Nous rappelons qu'un flux est dit (σ,ρ)-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 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 » proposée dans [27] mais la notion de (m,k)-firm n'est pas prise en compte.

4.2. Algorithme (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 fin de service décrit par l'équation 15. L'algorithme est décrit dans la figure 8. Le processus de service est activé quand il y a au moins un paquet dans une des files d'attente du système. Le serveur sélectionne selon les F_i^k le paquet ayant le plus petit temps virtuel de fin de service parmi tous les paquets critiques présents en tête des files actives (c'est-à-dire qui contient au moins un paquet). Si aucun paquet critique n'existe, le choix sera fait parmi les paquets optionnels. Puis, si le paquet sélectionné est critique, il est transmis directement par le serveur, tandis que si le paquet est optionnel, l'ordonnanceur vérifie avant sa transmission si ce paquet peut satisfaire son échéance. Si l'échéance souhaitée ne peut être garantie, le serveur rejette le paquet et refait une nouvelle sélection, sinon, il l'envoie.

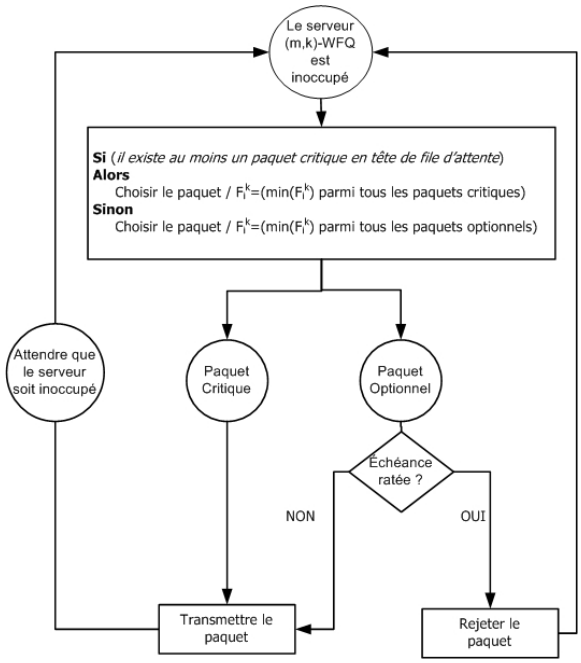

Figure 8. Algorithme (m,k)-WFQ

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 dégrade les performances des flux servis mais reste acceptable 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. Dans ce qui suit nous montrons quantitativement cette amélioration par la simulation d'un exemple.

4.3. Performances de (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.

	(m,k)	Débit	Trafic	κ-pattern	Echéance
Voix	(4,5)	64 kb/s	ON/OFF (500/750/50) 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

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` etc.

La deuxième source est une source 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` etc.

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 Mbit/s. Un flux FTP est vulnérable à la perte de paquets mais peut accepter un délai long. Bien que ce flux ne possède pas de propriétés temporelles strictes et que l'on aurait pu de ne pas lui réserver de bande passante (c'est-à-dire le laisser en best-effort), mais afin d'illustrer le comportement des différents algorithmes au cas où le système serait très chargé, nous avons choisi de lui réserver une bande passante de 7,936 Mbit/s. Par conséquent, nous fixons une garantie de type (0,1)-firm pour le flux FTP et une échéance infinie afin d'éviter tout rejet de paquets FTP optionnels.

Il est à noter que certains paramètres ont été choisis afin de simplifier l'interprétation des résultats de simulation. Ils ne correspondent pas nécessairement à des flux réels. C'est notamment le cas de taille constante de paquets de 8 Kbits, ainsi que les différentes contraintes (m,k)-firm données dans le tableau 8.2. Pour un cas plus proche des flux réels, le lecteur peut consulter [3].

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. 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 leurs échéances ratées.

	(m,k)-WFQ	WFQ	(m,k)-FIFO	FIFO
Voix	9,769 (*taux de rejet = 6,8 %*)	2428,031	5,111	48,031
Vidéo	3,999 (*taux de rejet = 5,5 %*)	55,391	4,952 (*taux de rejet = 3 %*)	49,031
FTP	9,696	36,562	5,339	49,083

Tableau 2. Bornes sur le**s temps de réponse (ms)**

Nous alertons le lecteur que le système sans rejet avec WFQ est en état de surcharge et n'atteint pas un régime stable, comme on peut le remarquer par son délai extrêmement grand. Ainsi, juste à titre de comparaison, ce délai est obtenu pour une durée de simulation large mais limitée puisque le système ne converge pas vers un état stationnaire pour le flux de voix avec WFQ. D'après la théorie des files d'attentes, un système ayant un processus d'arrivée aléatoire (ce qui est notre cas du flux ON/OFF exponentiel) et avec une charge moyenne égale à *1* fournit des délais qui tendent vers l'infini.

Comme prévu, (m,k)-WFQ fournit des bornes temporelles plus petites que celles garanties par WFQ pour chacun des flux. En effet, l'application FTP est « gourmande » en termes de bande passante et possède le taux de service le plus élevé bien qu'elle ne présente aucune exigence temporelle critique. WFQ ne peut différencier les différents flux que selon leurs étiquettes du temps virtuel de fin de service. Pour cette raison, le flux de voix ayant le plus petit taux de service et la plus grande taille de rafale, subit un délai très important. Cependant, (m,k)-WFQ réduit considérablement le délai mesuré pour les flux temps-réel en tenant compte leurs exigences temporelles (échéances et contraintes (m,k)-firm), en plus du taux de service. Dans cet exemple, étant donné que le trafic FTP n'a pas d'exigences temporelles strictes, tous ses paquets sont considérés comme optionnels par (m,k)-WFQ. Par conséquent, (m,k)-WFQ assure le service des paquets critiques des flux temps-réel avant le trafic *best-effot*. Ceci n'étant pas

possible avec WFQ parce qu'il ne peut pas différencier les différents types des flux.

D'autre part, (m,k)-FIFO fournit une borne sur le délai d'environ 5 ms pour tous les flux comme il n'y a aucune différentiation de service. Etant donné que le délai requis par le flux de voix est supérieur à celui garanti par (m,k)-FIFO, tous les paquets de voix seront transmis avec succès et respecteront tous leurs échéances désirées, tandis que 3 % des paquets vidéo seront perdus dus au rejet des paquets optionnels ratant leurs échéances de 4 ms.

4.4. Borne analytique du délai dans le cas d'une courbe d'arrivée (σ,ρ)-borné

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 8.5 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* [10]. Dans [3] 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 9 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.

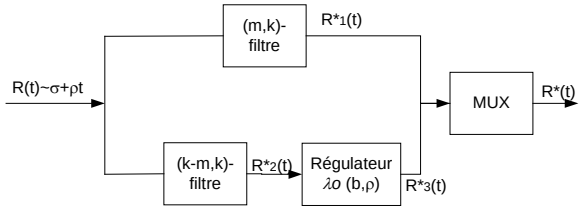

Figure 9. Modèle du network calculus

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 :

$$D_{\max}{}^{*} = \lambda_C \cdot \frac{\sigma}{g} + \lambda_O \cdot \frac{e}{g} + \frac{L_{\max}}{c} \qquad (17)$$

Avec $e \leq \sigma$ la taille maximale de rafale des paquets optionnels qui pourraient être transmis par l'ordonnanceur. λ_C désigne le taux de bits critiques du flux et λ_O le taux de bits optionnels du flux. Dans le cas où la taille des paquets est constante, on a $\lambda_C = m/k$ et $\lambda_O = (k-m)/k$. Si aucun paquet optionnel n'est servi, on a $D_{\min}{}^{*} = \lambda_C \cdot \frac{\sigma}{g} + \frac{L_{\max}}{c}$ qui est la plus petite borne sur le délai. Enfin si $e = \sigma$ qui signifie que l'on veut garantir la transmission de tous les paquets optionnels, on a le délai de WFQ donné par l'équation 16. Par conséquent, pour garantir un délai entre D_{min}* et D_{max}*, on peut alors ajuster l'échéance maximale D_{op} qui détermine la taille maximale de rafale des paquets optionnels qu'on désire admettre avec la relation $e = gD_{op}$.

4.5. Borne analytique dans le cas d'une courbe d'arrivée à deux segments

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 que WFQ. 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 [3] de façon similaire à l'obtention de l'équation 17. Dans ce cas le délai s'exprime par:

$$D_{\max} = \max\left[\begin{array}{l} \frac{M}{R} + \left(\frac{e-M}{R}\right)\left(\frac{p-R}{p-r}\right)^{+}, \\ \left(\frac{(\lambda_c M + \lambda_o \sigma)}{R} + \left(\frac{b-M}{R}\right)\left(\frac{\omega - R}{p-r}\right)\right) \end{array}\right] + T \qquad (18)$$

Avec $e \leq b$ la taille maximale de rafale des paquets optionnels qui pourraient être transmis par l'ordonnanceur et $\omega = \lambda_C p + \lambda_O e$. La figue 10 illustre le flux effectif réellement servi et sa borne sur le délai exprimé par l'équation 18.

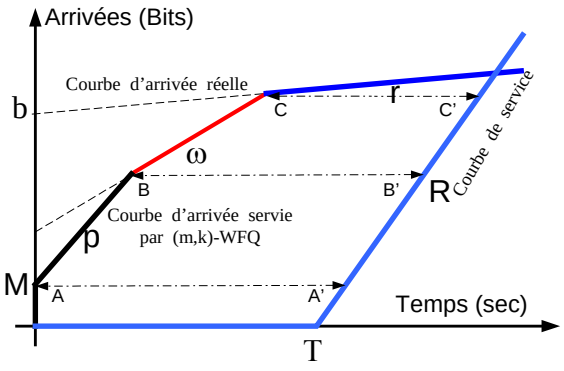

Figure 10. Borne maximale sur le délai par (m,k)-WFQ dans le cas du trafic (M,p,b,r)-borné

5. Problème de taux d'utilisation de ressources

Comme nous avons dit à l'introduction, intuitivement réserver des ressources pour satisfaire (m,k)-firm au lieu de (k,k)-firm devrait demander moins de ressources. C'est effectivement le cas quand l'ensemble de flux est servi par un serveur WFQ ou quand on peut se contenter d'une garantie statistique de (m,k)-firm. Malheureusement dans le cas général sans préemption, puisque le pire cas doit être considéré, il a été prouvé [28, 29, 30] que garantir de façon déterministe (m,k)-firm conduit souvent à exiger une réservation de ressource comme si on devait garantir (k,k)-firm. Au fait, le problème de l'ordonnancement non préemptif des paquets de N flux sous contrainte (m_i,k_i)-firm avec un serveur partagé a été prouvé NP-dur au sens fort. En plus les contraintes (m_i,k_i)-firm peuvent être violées même sous une charge arbitrairement basse. Ce qui diminue sérieusement l'intérêt pratique d'utilisation du modèle (m,k)-firm lors de la réservation de bande passante dans un réseau pour supporter efficacement la QdS.

Face à ce problème NP-dur, trois approches sont possibles pour réduire le pessimisme lors de la réservation de ressources. La première consiste à se contenter de la garantie probabiliste et par conséquent s'intéresser à la distribution de temps de réponse dans l'intervalle $[C_i, D_i]$. La deuxième consiste à spécialiser l'ensemble de flux. Par exemple, West et al. [31] ont étudié le cas où tous les flux ont des paquets de la même taille et avec la même période et ont obtenu un taux d'utilisation à 100%. Néanmoins il est évident que ce modèle tellement particulier ne peut pas être appliqué directement à la transmission de flux multimédia car chaque flux peut avoir sa propre taille de paquets et sa période. La troisième approche consiste à relâcher la contrainte (m,k)-firm (dans la mesure du possible). Par exemple, l'échéance de chaque paquet est relâchée dans [32] permettant de réduire le besoin en réservation de ressources. Par rapport à la première approche, la troisième permet de fournir une garantie (m,k)-firm relâchée mais déterministe.

Dans ce document nous suivons la troisième approche en modifiant légèrement le modèle (m,k)-firm. Cette modification consiste à relâcher la contrainte traditionnelle de l'échéance sur chaque paquet. Nous définissons une échéance globale pour un ensemble de k paquets consécutifs. Ce nouveau modèle est noté par R-(m,k)-firm (Relaxed (m,k)-firm). Nous allons montrer que ce relâchement peut satisfaire certain nombre d'applications et permet de résoudre efficacement le problème du taux d'utilisation.

5.1. Définition de R-(m,k)-firm

Un exemple de R-(3,5)-firm est illustré (Figure 11).

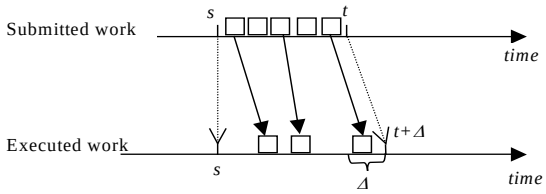

Figure 11. Contrainte R-(3,5)-firm

Définition de R-(m,k)-firm:
Pour un intervalle $[s, t]$ quelconque (avec $t\text{-}s \geq l \geq 0$), une source génère k unités de charge au serveur. Le serveur doit finir l'exécution d'au moins m parmi k unités avant $t+\Delta$, où Δ est le délai maximal tolérable du groupe de k unités de charge.

Les points suivants peuvent aider à comprendre cette définition.

1) Un intervalle $[s, t]$ quelconque sous entend qu'il s'agit une fenêtre glissante, qui peut commencer à n'importe que l'instant note par s.
2) Une source S_i périodique ou sporadique (C_i, T_i, m_i, k_i) ou (σ_i,ρ_i)-borné sous contrainte (m_i,k_i)-firm avec un délai Δ comme l'échéance d'un groupe quelconque de k paquets consécutifs commencé à l'instant s.
3) La contrainte est donnée à chaque source. C'est à dire que chaque source peut exiger sa propre contrainte sans considérer les autres sources. Le système doit garantir individuellement la contrainte R-(m_i,k_i)-firm pour chaque source.
4) Notre modèle inclut alors deux facteurs: **facteur (m,k) et facteur délai Δ**.
5) Le facteur (m,k): au moins m parmi k paquets consécutifs quelconques doivent être transmis avant $t+\Delta$. En général, le facteur (m,k) s'applique à la fenêtre glissante. Il est aussi applicable au cas de

fenêtres fixes sans superposition comme ce qui est défini dans [31].

6) Facteur délai Δ: ce facteur assure que le facteur (m,k) doit être réalise avant un délai maximal après la fin de la génération du $k^{ième}$ paquet en comptant à partir de l'instant *s*.

Avec cette nouvelle définition, il est clair que la garantie n'est plus donnée pour respecter l'échéance de chaque paquet individuellement mais elle est donnée à chaque flux pour respecter globalement une échéance sur un groupe quelconque de *k* paquets consécutifs.

5.2. Application dans un réseau

Figure 12 montre un exemple d'utilisation de R-(m,k)-firm dans un réseau. Supposons que la source a *k* paquets à transmettre à la destination durant l'intervalle [*s, t*], la QdS temps réel fournie par le réseau doit assurer que la destination reçoit au moins *m* paquets et ce avant $t+\Delta$.

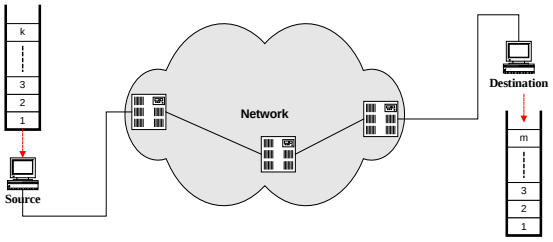

Figure 12: QdS de bout en bout par flux

La contrainte R-(m,k)-firm peut intéresser la transmission de paquets audio et video sur l'Internet. En effet, les paquets arrivés sont souvent mis en mémoire tampon jusqu'à une certaine limite Δ avant le déroulement du contenu (musique ou film) dans le buffer (*k* paquets par exemple). En cas de surcharge du réseau, il peut être préférable de rejeter un certain nombre de paquets (*k-m* paquets par exemple) afin de réduire immédiatement la congestion tout en assurant une certaine fluidité au détriment de la qualité (auditive ou visuelle).

5.3. Exemple de R-(m,k)-firm

Figure 13 montre l'avantage de R-(m,k)-firm par rapport au (m,k)-firm. Chaque bloc représente un paquet. La première ligne montre ce qui se passe au niveau de la source. Un flux périodique envoie un paquet au début de chaque période. La deuxième ligne montre une trace de l'ordonnancement sous contrainte (3,5)-firm. Dans cette deuxième ligne, le serveur est obligé de servir tous les blocs gris, sinon le système se trouverait dans l'état d'échec (comme dans le cas du $15^{ème}$ paquet). Une fois tomber dans un état d'échec (i.e., violation de contrainte (m,k)-firm), le serveur doit servir les trois paquets suivants ($16^{ième}$, $17^{ième}$ et $18^{ième}$) afin de recouvrir le système. Cette obligation de traiter des paquets consécutifs, s'il y a d'autres sources qui partagent le même serveur, va créer un besoin important de ressource (appelé point d'interférence dans [28], [33]). Il est clair qu'une bonne distribution de ces points d'interférence permet de réduire le besoin de ressources. Néanmoins trouver une distribution optimale est NP-difficile, rendant impossible d'éviter le sur-dimensionnement. La troisième ligne montre une séquence d'ordonnancement sous contrainte R-(3,5)-firm dont la taille de la fenêtre est $5T+\Delta$. Le lecteur peut vérifier, en glissant la fenêtre ou en décalant celle-ci de façon non superposée (i.e., fenêtre fixe), qu'il y a toujours au moins 3 paquets dedans.

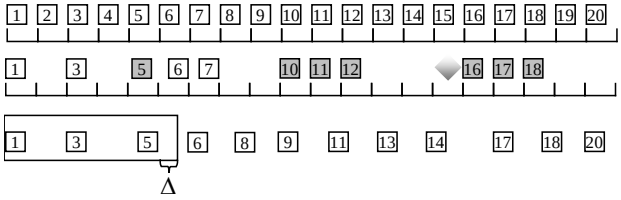

Figure 13: Exemple de l'ordonnancement sous contraintes (3,5)-firm et R-(3,5)-firm

Cet exemple montre que R-(m,k)-firm est plus souple car nous avons une liberté plus grande pour servir des paquets dans une fenêtre de *k* paquets, permettant ainsi d'éviter plus facilement les points d'interférence (pics de charge).

Maintenant le modèle R-(m,k)-firm est décrit, nous allons proposer dans la section suivante un mécanisme de gestion de file d'attente appelé DLB (Double Leaks Bucket) pour garantir de façon déterministe la contrainte R-(m,k)-firm tout en fournissant un taux élevé d'utilisation de ressources. En effet, satisfaire la contrainte R-(m,k)-firm n'est pas un trivial car le facteur (m,k) et le facteur délai sont antagonistes. D'une part, servir plus de paquets favorise le facteur (m,k) mais conduit à plus de délai. D'autre part, rejeter plus de paquets peut réduire le délai mais risqué de violer le facteur (m,k).

6. DLB pour la gestion des files d'attente

Pour garantir la contrainte *R-(m,k)-firm*, nous avons développé DLB pour rejeter une proportion de paquets d'un flux en cas de congestion [5]. Dans cette section nous décrivons ce mécanisme, donnons la condition suffisante pour la garantie *R-(m,k)-firm* des flux (σ,ρ)-borné et comparons ses performances avec RED et TD.

6.1. Fonctionnement de DLB

Le mécanisme DLB est directement dérivé du « leaky bucket » classique. DLB possède un seau (bucket) et deux sorties : sortie de flux servi notée par SL (Serving Leak) et sortie de flux rejeté ou une sorte d'évacuation de trop plein notée par DL (Discarding Leak) comme montré par la Figure 14.

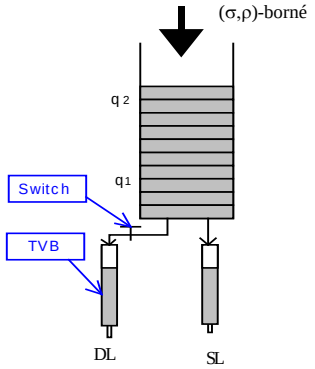

Figure 14: Model de DLB (version paquet)

Le réseau doit garantir les paquets qui traversent SL, tandis que DL, contrôlé par un « switch », donne la capacité de rejet de DLB. Avec une capacité garantie de SL, la contrainte *R-(m,k)-firm* peut être satisfaite. Les paquets sortis par DL peuvent d'ailleurs soit être rejetés définitivement, soit être conservés pour une transmission ultérieure avec une autre classe de trafic moins prioritaire par exemple.

Comme montré dans Figure 14, il y a deux buffers avec la capacité d'un paquet pour DL et SL notés par TVB (*temporary vessel buckets*). Ces buffers sont bloquants au sens où on ne peut prendre un paquet du DLB si le paquet dans le TVB est sorti. En plus, le TVB de DL ne peut prendre un paquet de DLB que seulement quand le « switch » est ouvert (en état 1). La Figure 15 montre le fonctionnement du « switch ». Soient C_1 et C_2 les capacités respectives de SL et DL. Le « switch » fonctionne selon le niveau du seau noté par q et est contrôlé par une fonction à double seuils (Figure 15) où 1 correspond à l'état ouvert et 0 à l'état fermé du « switch ».

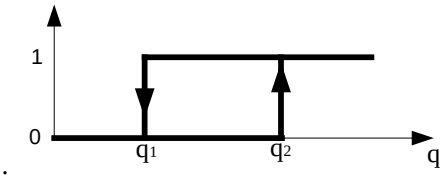

Figure 15: Fonction du contrôle du « switch »

Durant l'incrémentation du niveau, le « switch » reste fermé jusqu'au seuil q_2. Une fois ouvert à partir de q_2, le « switch » reste ouvert jusqu'au moment où le niveau est baissé à q_1. Quand le « switch » est en état ouvert, DL peut prendre des paquets un par un pour les rejeter avec un débit ne dépassant pas C_2.

6.2. Condition suffisante de garantie

Pour que la garantie soit possible, nous limitons le flux d'entrée à (σ,ρ)-borné. La Figure 16 montre l'évolution de la courbe de service en dessous de la courbe d'arrivée (σ,ρ). Notons que contraiement à DL qui est sous le contrôle du « switch », SL est tout le temps en service sauf quand le seau est vide.

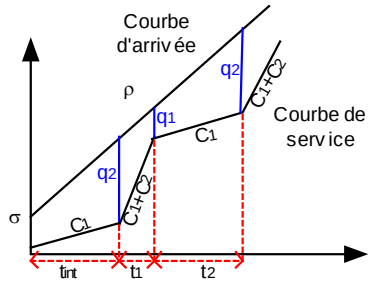

Figure 16 : Evolution de la courbe de service

Soit *S* la taille d'un paquet, la garantie déterministe de la contrainte *R-(m,k)-firm* d'un flux (σ,ρ)-borné est donné par les deux conditions suivantes.

Condition (1) $q_1 \geq \frac{C_1}{C_2} \geq \frac{m}{k-m}$

Condition (2) Si $\sigma < q_2$ alors

$$\max\left(\frac{q_2-1}{C_1}S, \left(\frac{q_2-q_1}{C_1+C_2}+\frac{q_1}{C_1}\right)S\right) \leq \Delta$$

sinon

$$\max\left(\frac{q_2-1}{C_1}S, \left(\frac{\sigma-q_1}{C_1+C_2}+\frac{q_1}{C_1}\right)S\right) \leq \Delta$$

Condition (1) assure le facteur (m,k) tandis que Condition (2) assure le facteur délai Δ de R-(m,k)-firm.

Nous renvoyons le lecteur à [5] pour la preuve.

6.3. Application numérique

Considérons les paramètres d'un flux audio-CBR générant un débit de 1.4Mbit/s. Le flux est borné par (σ,ρ) = (2kbits, 1,4Mbit/s). La taille des paquets S=144 octets (taille normale de transmission audio). Comme un

exemple, nous supposons qu'un tel flux est sous contrainte R-(3,5)-firm[1] et avec △=20ms.

Par exemple, à un moment donné, seulement 1,2 Mbit/s est disponible quand le flux audio arrive. Visiblement cette bande passante n'est pas suffisante pour garantir l'échéance de tous les paquets.

Nous configurons alors les paramètres de DLB comme suivant. C_1=1.008Mbit/s, C_2=0.672Mbit/s, q_1=2, q_2=5.

Avec ces paramètres et en appliquant Condition (2) nous garantissons la contrainte *R-(m,k)-firm* car $t_{delay} \le \frac{q_2 - 1}{C_1} S$ =4,6ms ≤△=20.

Lorsqu'il s'agit de garantir l'échéance de 20ms pour tous les paquets du flux, il faut $\rho + \sigma / t_{delay}$ = 1,5Mbit/s

Cet exemple illustre clairement l'intérêt de DLB lorsqu'il s'agit de gérer la congestion ou maximiser le nombre de flux temps réel admis pour une bande passante donnée.

6.4. Comparaison de performances

Nous venons de montrer que DLB configuré selon notre condition suffisante permet effectivement de fournir la garantie déterministe vis à vis des flux (σ,ρ)-borné. Afin d'illustrer les performances de DLB pour des flux généraux, nous simulons le comportement de DLB ainsi que de TD et RED avec un flux d'arrivée de Poisson du taux moyen d'arrivé de $\lambda=1$ paquet par unité de temps (ms).

Comme son nom l'indique, Le mécanisme TD (Tail-Drop) rejette tous les paquets quand la file d'attente est pleine. RED a besoin de 5 paramètres : la taille maximale de file d'attente q_{max}, les seuils minimal (min_{th}) et maximal (max_{th}) qui définissent la région RED, la probabilité maximale de rejet (max_p), et le facteur poids utilisé pour calculer la taille moyenne de la file d'attente.

Nous supposons que la source générant ce flux peut tolérer jusqu'au 33% de perte de paquets. DLB est paramétré comme suivant. q_1=3 paquets, q_2 = 6 paquets, q_{max} = 9 paquets, C_1 = 0.8 paquet/ms, et C_2 = 0.4 paquet /ms.

Pour une comparaison équitable, la meme taille maximale de file d'attente et la même capacité de serveur sont donné à DLB, TD et RED. RED est donc configure comme suivant. min_{th} = 3, max_{th} = 6, q_{max} = 9, w_q = 0.2, max_p = 0.34, C = 0.8, de sorte que les paquets sont rejetés de façon probabiliste quand la taille moyenne de la file d'attente se trouve entre min_{th} = 3 et max_{th} = 6.

Figure 17 montre une trace de simulation avec 1 qui représente un paquet servi et 0 un paquet rejeté.

DLB (q_1=3, q_2 = 6, q_{max} = 9, C_1 = 0.8, C_2 = 0.4)
01101101101101101101101101101101101101111111111111
11110110110110110110110110111111111011011011011011
01101101101101111111111111111111111111111101101101
10110110110110110110110110

TD (q_{max} = 9, C = 0.8)
10110111110000110001111111111111111111111111011101
01001011111111101111110100111110101001010000111110
01111111111111111111111111111111111111101110110011
00011111101111110111111110

RED (min_{th} = 3, max_{th} = 6, q_{max} = 9, w_q = 0.2, max_p = 0.34, C = 0.8)
11110100100001111111111111111111111111111101011111
11111001100111110111111111111111111111111111111011
00101110111110111111111111111111101000011110001100
01010111111111111111111101010

Figure 17: Séquences de pertes de DLB, TD, RED

Les performances de DLB, RED et TD sont données dans le Tableau 3 en termes de taille moyenne de la file d'attente, délai, taux de pertes, ainsi que la maximale et la moyenne de pertes consécutives.

	DLB	RED	TD
Taille moyenne de file	4.7	4.3	9.0
Délai moyen (ms)	4.8	4.5	8.7
Taux de perte	22%	22%	20%
Pertes consécutives max.	1	4	4
Pertes consécutives moy.	1	1.64	1.54

Tableau 3: Comparaison de DLB, RED et TD

Du Tableau 3, on voit que le taux de pertes est presque le même pour les trois mécanismes, ce qui est normal car la même capacité est donnée à DLB, RED et TD. En termes de délai et de taille de file d'attente, DLB et RED sont clairement mieux. Pour le même taux de pertes, la distribution des pertes est très différente. DLB se comporte remarquablement bien en termes de pertes consécutives maximales et en moyenne. Ce dernier point est le plus important car il a un impact sur la QdS.

[1] La détermination de valeur judicieuse de m et k dépend de l'application. (3,5) n'est donné que pour prendre un exemple numérque.

Références

[1] Floyd, S., and Jacobson, V., "Random Early Detection Gateways for Congestion Avoidance". In *ACM/IEEE Transactions on Networking*, 3(1), August, 1993.

[2] Hamdaoui, M. and Ramanathan, P., "A dynamic priority assignment technique for streams with (m,k)-firm deadlines", *IEEE Transactions on Computers*, 44(4), pp1443–1451, Dec. 1995.

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

[4] Song Y.Q., Koubâa A., Li J., « Chapitre 8 : Qualité de service temps réel selon le modèle (m,k)-firm », *Systèmes temps réel*, Vol.2, pp213-244, Hermes-Lavoisier, 2006.

[5] Li J., *Garantir la qualité de service temps réel selon l'approche (m,k)-firm*, Thèse INPL, 14 février 2007.

[6] Liu, C.L. and J.W. Layland, "Scheduling Algorithms for Multiprogramming in Hard Real-Time Environment", *Journal of ACM*, Vol. 20 (1), pp46-61, 1973.

[7] Cruz, R. L., "A calculus for network delay, Part I", *IEEE Trans. on Information Theory*, 37(1):114-131, Jan. 1991.

[8] Cruz, R. L., "A Calculus for Network Delay, Part II: Network Analysis, *IEEE Transactions on Information Theory*, 37(1): 132-141, Jan. 1991.

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

[10] 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.

[11] Kleinrock L., Queueing systems, Vol. 1: Theory, John Wiley, 1975.

[12] Kleinrock L., Queueing systems, Vol. 2: Computer application, John Wiley, 1976.

[13] Tindell, K., *Fixed Priority Scheduling of Hard Real-Time Systems*, PhD Thesis, Department of Computer Science, University of York,YO1 5DD England, 1994.

[14] Joseph, M., and P. Pandya, "Finding Response Time in a Real-Time System", *BCS Computer journal*, Vol. 29, No.5, pp.390-395, Oct. 1986.

[15] Lehoczky J.P., "Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines", *IEEE RTSS*, pp.201-209, Los Alamitos (USA), 1990.

[16] Audsley, N., A. Burns, M. Richardson, K. Tindell and A. Wellings "Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling", Software Engineering Journal, 8(5), pp. 284-292, September 1993.

[17] Burns, A., M. Nicholson, K. Tindell and Zhang "Allocating and Scheduling Hard Real-time Tasks on a Point-to-Point Distributed System", *Workshop on parallel and Distributed Real-Time Systems*, pp. 11-20, Newport Beach California, April 13-15 1993.

[18] 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.

[19] Georges L., P. Mühlethaler and N. Rivierre, "A Few Results on Non-Preemptive Real-Time Scheduling", *INRIA research report n°RR3926*, may 2000.

[20] Georges L., N. Rivierre, M. Spuri, "Preemptive and Non-Preemptive Real-Time UniProcessor Scheduling", *INRIA research report n°RR INRIA2966*, Sept. 1996.

[21] Briand, P., M. Roy, *Meeting Deadlines in Hard Real-Time Systems, The Rate Monotonic Approach*, IEEE computer Society ISBN 0-8186-7406-7.

[22] Song, Y.Q., "Time Constrained Communication Over Switched Ethernet", *Proc. of IFAC Fet'2001*, pp.177-184, Nancy (France), Nov. 15-16, 2001.

[23] Song, Y.Q., A. Koubâa, and F. Simonot, "Switched Ethernet For Real-Time Industrial Communication: Modelling And Message Buffering Delay Evaluation", in *4th IEEE WFCS'2002*, pp27-35, August 28-30, 2002.

[24] Jasperneite J., P. Neumann, M. Theis, K. Watson "Deterministic Real-Time Communication with Switched Ethernet" in *4th IEEE WFCS*, Vasteras (Sweden), August 2002.

[25] Charara H., J.-L. Scharbarg, J. Ermont, and C. Fraboul, "Methods for bounding end-to-end delays on an AFDX network", 18th ECRTS, Dresde, Germany, July 2006.

[26] 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.

[27] 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.

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

[29] Mok, A. K. and Wang, W. R., "Window-Constrained Real-Time Periodic Task Scheduling", *22nd IEEE RTSS*, pp15-24, London, England, December, 2001.

[30] Li, J., *Sufficient Condition for Guaranteeing (m,k)-firm Real-Time Requirement Under NP-DBP-EDF Scheduling*, Technical report No. A03-R-452, Stage de DEA, LORIA, June, 2003.

[31] West, R., Zhang, Y., Schwan, K. and Poellabauer, C., "Dynamic Window-Constrained Scheduling of Real-Time Streams in Media Servers", *IEEE Transactions on Computer*s, Vol.53, NO. 6, pp. 744-759, June, 2004.

[32] Zhang, Y., West, R. and Qi, X., "A Virtual Deadline Scheduler for Window-Constrained Service Guarantees", *25th IEEE RTSS*, 2004.

[33] Jia, N., Hyon, E., Song, Y.Q., "Ordonnancement sous contraintes (m,k)-firm et combinatoire des mots", *RTS'2005*, Paris (France), April 2005.