Garantir la qualité de service temps réel selon l'approche (m,k)-firm

Jian Li

Institut National Polytechnique de Lorraine
Département de la formation doctorale en Informatique Ecole doctorale IAE+M

Garantir la qualité de service temps réel selon l'approche (m,k)-firm

THESE

Présentée et soutenue publiquement le 14 Février 2007 Pour l'obtention du

Doctorat de l'Institut National Polytechnique de Lorraine
(Spécialité Informatique)
Par

LI Jian

Composition du Jury

Président	Jean-Yves Marion, Prof. à LORIA-INPL
Rapporteurs	Pascale Minet, Chargé de recherche à INRIA Rocquencourt Pascal Lorenz, Prof. à l'Université de Haute Alsace
Examinateurs	Pascal Richard, Maître de conférence à LISI / ENSMA Man Lin, Associate Prof. à St. Francis Xavier University, Canada Françoise Simonot-Lion, Prof. à LORIA-INPL
Directeur de thèse	Ye-Qiong Song, Prof. à LORIA-INPL
Codirecteur de thèse	Nicolas Navet, Chargé de recherche à INRIA Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications - UMR 7503

Introduction générale

Aujourd'hui, la technologie temps réel est omniprésente, et de plus en plus d'infrastructures dépendent d'elle. Les domaines des applications typiques du calcul temps réel et de la communication temps réel incluent le contrôle des procédés industriels, la fabrication, l'avionique, la commande de trafic aérien, les multimédia, les télécommunications (l'autoroute de l'information), la télé-médicine et le soin intensif surveillé, la défense, etc.

Dans les systèmes de contrôle temps réel, les tâches sont habituellement périodiques et ils ont des contraintes de l'échéance, avant lesquelles chaque instance d'une tâche devrait accomplir son calcul. Dans les cas défavorables où il y a les composants en pannes, les techniques d'une reconfiguration s'applique pour restaurer des échecs de processeur; qui assignent toutes les tâches aux processeurs en état. Cette reconfiguration peut conduire à la surcharge de processeur à point qu'il n'est plus possible de satisfaire toutes les échéances des tâches. D'ailleurs, bien que la bande passante des réseaux d'aujourd'hui soit relativement abondante, l'apparition des nouvelles applications de l'Internet, telles que la transmission audio/vidéo multimédia, mènent au même problème des ressources limitées qu'avant.

Généralement les systèmes fonctionnent pendant de longues périodes dans des environnements non déterministes assujettis à des fautes, tant que possible, ils devraient pouvoir tolérer les fautes et continuer de fonctionner correctement. La dégradation contrôlée (graceful degradation) est une manière de fournir un niveau réduit de service plutôt que d'échouer complètement en cas de surcharge de système ou en cas de fautes inattendues. Par exemple, les flux multimédias ont habituellement les taux de transmission variables et peuvent tolérer des échéances ratées ou des paquets perdus à condition qu'ils soient espacés correctement dans le temps, ce fait est dû à la

redondance dans le code et la tolérance de perception humane. Jusqu'à maintenant, comment mesurer exactement le QoS résultante des applications multimédia reste encore une question ouverte.

Il existe de différentes contraintes temps réel selon les applications et surtout en termes de leur niveau de tolérance aux fautes temporelles. Formellement, la contrainte temps réel peut être classifiée dans le *temps réel dur (hard real time HRT)*, le *temps réel souple (soft real time SRT)*, et *le temps réel firm (firm real time)*. Un système temps réel dur exige de servir toutes les instances avant leurs échéances. Cette condition rigoureuse, d'une part, n'est pas nécessaire pour tous les systèmes puisqu'un certain nombre d'échéances ratées est tolérable pour certaines applications. D'un autre part, l'occurrence des fautes (par exemple l'échéance ratée, paquets perdus, etc.) ne peut pas toujours être évitée pour les systèmes temps réel adaptatifs parce que, essentiellement, les systèmes et ses environnements ne sont pas entièrement prévisibles à l'avance.

Par contre, les systèmes sous contraires temps réel souple (soft real-time SRT) peut accepter un certain nombre d'échéances ratées de temps en temps, qui au mieux est exprimé par des garanties probabilistes ou de statistiques. Cependant, la dégradation contrôlée exige non seulement la fiabilité mais également la disponibilité. Par exemple, beaucoup de fautes qui se produisent dans un intervalle court peuvent mener à une dégradation statistiquement acceptable pour quelques applications, néanmoins la densité des fautes peut être nuisible pour quelques autres applications.

Par conséquent, la contrainte temps réel firm (firm real-time : FRT) [Ramanathan95] s'avère intéressante pour éviter le cas où il y a un grand nombre de fautes consécutives dans un intervalle court. En particulier, la contrainte « (m, k)-firm » exige qu'au moins *m* instances devraient être finis avant leurs échéances parmi n'importe quel *k* instances consécutives.

Étroitement liée aux contraintes temps réel firm est « weakly hard real-time » (WHRT) qui met des restrictions sur le nombre d'échéances qui peuvent être ratées (ou doivent être rencontrées) dans un certain nombre d'échéances consécutives. Et quelque part, le temps réel « (m, k)-firm » a été suggéré pour être une sous-classe de

WHRT [Bernat01]. Bien qu'ils tous les deux contraindrent des échéances ratées à une limite précise, une différence inhérente existe entre les deux types de contraintes temps réel. En fait, le FRT suppose qu'il est inutile d'exécuter l'instance si elle ne peut pas être entièrement finie avant son échéance. Tandis que, sous WHRT, une instance est encore exécutée quand elle excède son échéance et peut causer la suspension d'elle-même. De notre point de vue, WHRT est une sous-classe de SRT. Réciproquement, FRT peut être considéré comme un ordonnancement actif aux fautes pour le system, qui jette l'instance quand elle n'est pas possible de finir avant son échéance. Ce rejet actif peut réduire la quantité de travail à l'avenir pour le processus d'ordonnancement, et le facilite, en comparaison de WHRT, pour ordonnancer les instances suivantes. En outre, FRT peut éviter la perte des ressources par l'exécution inutile des instances qui ne satisfont pas leurs contraintes.

Le but des systèmes temps réel adaptatifs est de fournir des garanties de performances acceptables a priori au niveau de système et de fournir la dégradation contrôlée en présence des fautes. Ceci exige un certain genre de déterminisme/prévisibilité, qui implique que, ayant prétentions de la quantité de travail et certaines tolérances aux fautes, on doit pourvoir dimensionner les ressources exigées au moment de la conception.

De notre point de vue, la contrainte (m, k)-firm fournit un cadre convenable et puissant pour indiquer le niveau de la tolérance aux fautes. Nous choisissons de concentrer cette thèse sur l'utilisation de la contraintes (m, k)-firm dans les systèmes temps réel adaptatifs. Le défi de la recherche est de développer les techniques efficaces de gestion de ressources en utilisant la contrainte (m, k)-firm et d'évaluer leurs exécutions dans le contexte de la transmission temps réel adaptative de systèmes de contrôle-commande et de multimédia. Traditionnellement, la garantie de la QoS (Quality of Service) temps réel est réalisée en réservant à l'avance les ressources selon le pires cas, appelé le surapprovisionnement, et il induit un taux d'utilisation des ressources basse. Évidemment, si faisable, il vaut mieux de réserver la ressource selon le taux moyen de la charge, et de jeter quelques demandes en cas de surcharges. Ce fait est plus efficace à l'utilisation de ressource plutôt que de réserver beaucoup plus de ressources en garantissant toutes les demandes dans le pire cas. Autrement dit que, le

problème principal est comment déterminer la ressource minimale requise pour la garantie déterministe sous la contrainte (m, k)-firm, ou comment servir au plus d'applications temps réel avec la capacité de ressource disponible.

Ce but de recherche se concentre sur la conception des réseaux de nouvelle génération (next generation network : NGN), en particulier, ressource et fonction de control d'admission (« resource and adimission control functions » RACF) qui permettront aux opérateurs de garantir la qualité de bout en bout pour des services multimédia, tels que VoIP et IPTV, etc. Notre recherche peut fournir à un opérateur la capacité de définir des règles pour les types de communication spécifiques, afin d'allouer au mieux des ressources de réseau.

Intuitivement, la ressource requise par un ensemble de tâches selon la contrainte (m, k)-firm devraient être moins que celle sans dégradation (HRT). Si la contrainte (m, k)-firm n'économise pas la ressource par rapport à HRT, elle perdra ses motivations et intérêts originaux. Par conséquent, la ressource requise constitue en le critère et la mesure les plus importants de l'efficacité d'ordonnancement.

Dans la première partie de cette thèse, nous introduisons l'état de l'art concernant algorithmes d'ordonnancements sous contrainte (m, k)-firm (modèle fixe et modèle dynamique) ainsi qu'une autre contrainte similaire, appelé DWCS (Dynamic window constraint scheduling). Nous analysons et comparons alors les approches d'ordonnancement et rappelons les conditions pour déterminer l'ordonnançabilité d'un ensemble de tâches. Ensuite, nous présentons les trois contributions principales de cette thèse.

La *première contribution* est une condition suffisante d'ordonnançabilité pour l'algorithme classique NP-DBP-EDF « Non preemptive – Distance based priority – earlieast deadline first [Hamanathan99] ». Ce travail est important puisqu'il n'y avait aucune condition suffisante d'approvisionnement avant pour l'algorithme dynamique d'ordonnancement sous la contraint (m, k)-firm.

La *deuxième contribution* est une novelle contrainte temps réel qui mène à une meilleure utilisation de ressource que (m, k)-firm. Pour cela nous présentons cette contribution selon les étapes suivantes :

- D'abord, selon les littératures existantes [George02] [Mok01] [Quan00] [Jeffay91], nous expliquons les raisons théoriques de l'utilisation basse de ressources dans l'analyse d'ordonnançabilité du pire cas [LiDEA03] [LiETFA06].
- Deuxièmement, nous donnons les orientations de recherche visant à augmentant le taux d'utilisation de ressources. En fait, comme nous avons déjà expliqué, tous les schémas d'ordonnancement dans l'état de l'art suivent une ou une autre orientation de recherche indiquée ici.
- Troisièmement, nous proposons une nouvel contrainte temps réel en relaxant (m, k)-firm pour défier la contrainte traditionnelle d'échéance sur les instances individuelles. Cette novelle contraint temps réel est appelé contrainte « relaxed (m, k)-firm notée R-(m,k)-firm », qui remplace l'échéance traditionnel de paquet (par exemple) par un facteur de délai d'un groupe de paquets (ou un group d'instances de tâche). Nous demontrons que cette contrainte est plus flexible et bien adaptée aux flux multimédias, tels que MPEG et MP3, etc. Enfin, une condition suffisant d'allocution de ressources est donnée pour un ensemble de tâches sous l'ordonnancement non préemptif à priorité fixe. Les simulations démontrent l'avantage en termes d'utilisation de ressources.

La *troisième contribution* est un nouveau mécanisme de gestion de file d'attente, appelé « *double leaks bucket* » (DLB), qui peut fournir la garantie déterministe de la contrainte R-(m, k)-firm. DLB peut être mis en application dans Diffserv et d'autres mécanismes de QoS, puisqu'il peut être simplement mis en application, en remplaçant le mécanisme largement employé « Leaky Bucket », pour gérer les transmissions multimédia temps réel sous la contrainte R-(m, k)-firm. Pour analyser ce nouveau mécanisme, nous nous servons de la théorie des files d'attente et « network calculus ».

L'applicabilité de notre nouvelle contrainte temps réel et le nouveau mécanisme sont démontrés sur la transmission de paquets pour les flux multimédias et sur un système de contrôle embarqué dans l’automobile.

Chapitre 1
Contexte et état de l'art

Dans ce chapitre, nous présentons d'abord le contexte général des approches temps réel courantes comprenant le modèle de tâches temps réel et le modèle de système d'analyse. Les définitions de tâches périodiques et sporadiques sont présentées en détail puisque leurs paramètres réguliers mènent habituellement à beaucoup de résultats déterministes. Ensuite, nous présentons l'état de l'art concernant la contrainte « (m, k)-firm » et les approches d'ordonnancement récentes à la garantie de la contrainte (m, k)-firm. La contrainte de fenêtre dynamique (window-contrainte), une contrainte semblable à (m, k)-firm, sera également présentée ainsi que ses approches d'ordonnancement dynamiques. En plus, une analyse et une comparaison profondes sont effectuées sur les divers algorithmes d'ordonnancement.

Modèle de système

Dans le système temps réel embarqué, les accès concourants de ressources se produisent dans un système, qui a une ressource limitée. Cette ressource pourrait être la capacité du processeur pour exécuter la demande d'exécution de tâches ou la bande passante de réseau pour la transmission de messages. Donc, le problème est comment ordonnancer ces demandes d'accès tandis que toujours garantir la performance temporelle de tâche. L'optimisation du taux d'utilisation de ressources est également importante puisque c'est un critère pour mesurer l'efficacité d'une méthode

d'ordonnancement. Dans la suite, quelques modèles d'ordonnancements et de tâche temps réel seront présentés.

Modèle MIQSS

Le modèle « multiple input queues single server » (MIQSS) peut être s'employé pour étudier une grande catégorie de systèmes d'informatique et de télé-communications, tels que WordFIP et applications de FDDI, CAN pour le système embarqué dans l'automobile, le système distribué du réseau de PLC (power line communication) et le système intégral de l'Ethernet communauté.

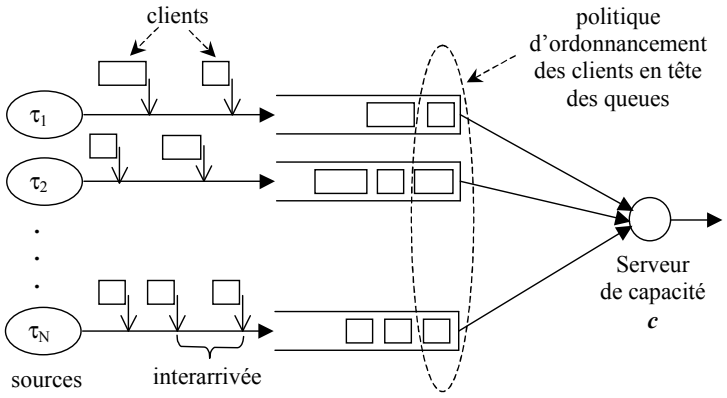


Figure 1: *Modèle MIQSS*

Le modèle proposé se compose de *N* sources en produisant les *N* flux de tâches τ_i ($i = 1, 2, \ldots n$) qui désirent d'être servi par un serveur unique. Chaque flux est constitué par une source et une file d'attente, où une instance générée par une tâche attend jusqu'à être choisi par le processeur. Le processeur choisit l'instance de la tête des files d'attente selon sa politique d'ordonnancement.

WFQ et CBQ

« Weighted Fair Qeueing » (WFQ) est une technique d'ordonnancement de paquet permettant des services garantis de bande passante. Le but de WFQ est de permettre à plusieurs sessions de partager le même lien. WFQ est une approximation du « Generalized Processor Sharing » (GPS) qui, comme nom suggère, est une géné-

ralisation du « Processor Sharing » (PS) [Kleinrock76]. Dans PS, chaque session a une file d'attente de FIFO (First In First Out) individuellement. À n'importe quelle moment, *N* sessions actives (celles avec les files d'attente non vides) sont servies simultanément, chacun à un taux de 1/N de la vitesse de lien (bande passante). Au contraire du PS, GPS permet à différentes sessions d'avoir différentes parts de service.

Le GPS possède plusieurs propriétés intéressantes. Grâce au fait que chaque session a sa propre file d'attente, seules les sessions mal-comportées (qui envoient beaucoup de données) sont pénalisées. En plus, le GPS permet à des sessions d'avoir des bandes passantes garanties assignées à elles. Parekh [Parekh92] a prouvé que quand un réseau emploie le GPS et une session est contrainte par « leaky bucket » la limite d'un retard de bout en bout peut être garanti.

Modèle de source

Les sources temps réel sont habituellement caractérisées par des tâches, telles que :

- Périodique et sporadique : la tâche périodique est une tâche temps réel, qui est activée (lancée) régulièrement aux taux fixes (périodiques). Normalement, une tâche périodique exige que son instance soit exécutée une fois par période. La tâche sporadique est une tâche temps réel qui est activée irrégulièrement borné par certains taux connu. Le taux borné est caractérisé par une période inter-arrivée minimale, c'est-à-dire, un intervalle de temps entre deux activations successives. La contrainte de temps est habituellement une échéance.

- (r, b)-borné : Généralement, une courbe d'arrivée sous (r, b)-borné [Leboutec02] [Chang00] permet à une source d'envoyer un e raffale de *b* à la fois, mais pas supérieur que *r* bit/s en moyenne. Les paramètres *b* et *r* s'appellent la raffale (par les unités des données) et le taux d'arrivé (en termes des unités des données par unité de temps) respectivement.

- Stochastique : un processus stochastique s'emploie, où les instances arrivent aléatoirement. Une distribution de Poisson est un exemple des instants d'arrivées des instances.

Paramètres des tâches périodiques et sporadiques

Dans cette section, nous présenterons les définitions de paramètres de la tâche périodique et sporadique en détail, ainsi que les définitions de l'ensemble de tâches concret et l'ensemble de tâche abstrait (non concret).

Définition traditionnelle

Beaucoup de résultats déterministes sont obtenue pour un ensemble de tâches au cas où l'échéance d'une tâche est égale à sa period. Dans ce que suit, le paramètre d'échéance sera ignoré puisqu'il égale à l'échéance. La tâche τ_i est formellement une paire (c_i, p_i), où :

c_i est le coût d'exécution : la quantité maximal de temps de processeur requise pour s'exécuter (le programme séquentiel de) la tâche τ_i sur un processeur unique consacré.

p_i est la période : l'intervalle minimal entre les instances d'une tâche

Dans tout ce document, nous supposons que le temps est discret et les tics d'horloge sont classés par les nombres naturels. Instances de tâches se produisent et exécutions de instance de tâche commencent et se terminent aux clic d'horloge ; chacun paramètre de c_i et de p_i est exprimé en termes d'un multiple (l'intervalle entre) des clic d'horloge. Si une tâche τ_i avec le coût d'exécution c_i est exécutée sans interruption sur un unique processeur au temps t, alors son exécution sera accomplie au temps $t+c_i$. Nous considérons deux paradigmes d'instance de tâches : périodique et sporadique. Si τ_i est périodique, la période p_i indique l'intervalle constant entre les instances. Si le τ_i est sporadique, p_i indique l'intervalle minimum entre les instances. La définition du comportement d'une tâche dépend qu'elle est périodique ou sporadique. Le ***comportement d'une tâche périodique*** $\tau_i = (c_i, p_i)$ est donné par les règles

suivantes pour l'instance et l'exécution de τ_i. Si $t_{i,k}$ est le temps où $k_{ième}$ instance de tâche τ_i arrive, alors :

- $(k+1)_{ième}$ instance de tâche τ_i se produira au temps $t_{i,k+1} = t_{i,k}+p_i$.
- $k_{ième}$ exécution de tâche τ_i doit commencer pas plus tôt que $t_{i,k}$ et être accompli pas plus tard que les échéances $t_{i,k}+p_i$. Ce fait exige que le processeur soit assignées c_i unités du temps à l'exécution de τ_i dans l'intervalle $[t_{i,k}, t_{i,k}+p_i]$

Le comportement d'une tâche sporadique est légèrement moins contraint que celle d'une tâche périodique. Le ***comportement d'une tâche sporadique*** $\tau_i = (c_i, p_i)$ est donné par les règles suivantes pour l'instance et l'exécution du τ_i, Si $t_{i,k}$ est le temps où $k_{ième}$ instance de tâche arrive, alors :

- le $(k+1)_{ième}$ instance de τ_i se produira pas plus tôt que le $t_{i,k}+p_i$, alors $t_{i,k+1} \geq t_{i,k}+p_i$.
- $k_{ième}$ exécution de tâche τ_i doit commencer pas plus tôt que $t_{i,k}$ et être accompli pas plus tard que l'échéance $t_{i,k}+p_i$.

Observer que les comportements de tâches périodiques et sporadiques diffèrent seulement dans la première règle. Nous supposons que les instances de la tâche sporadique sont indépendantes, et ce fait est dans le sens que le temps où une instance de tâche sporadique est invoquée selon seulement sa dernière instance et pas sur l'instance de toute autre tâche. Noter que le comportement des pires cas d'une tâche sporadique $\tau_i = (c_i, p_i)$ (le pire cas est dans le sens d'avoir besoin de plus de temps d'exécution de processeur), se produit quand τ_i est invoquée de chaque intervalle de p_i.

Ensemble de tâches concret et abstrait (non concret)

La difficulté de l'ordonnancement des tâches périodiques peut être affectée par les temps où les premières instances des tâches sont relancées [Jeffay91]. En outre, nous sommes intéressés par la prévisibilité de l'ordonnançabilité qui peut servir à dimensionner la ressource de réseau, ou évaluer la QoS avant la transmission. Par

conséquent, le but de l'étude sur l'ensemble de tâches avec les temps d'activation non déterminés, et nous présenterons les définitions des ensembles de tâches concrète et abstraite (non concrète) [Jeffay91].

Une tâche concrète est une paire (τ_i, r_i), où τ_i est une tâche, et r_i est un nombre entier non négatif qui exprime le temps de la première instance ou le temps d'activation de tâche. Le comportement de (τ_i, r_i) est le comportement du τ_i se produit au temps de r_i. Une fois que une tâche est activée, les instances arrivent répétitivement pour toujours.

Un ***ensemble de tâches périodiques (ou sporadiques) abstrait*** $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ est un ensemble de tâches classées de *1* à *n,* où pour chacun *i*, $1 \leq i \leq n$, $\tau_i = (c_i, p_i)$. Un ***ensemble de tâches périodiques (sporadiques) concret*** $\omega = \{(\tau_1, r_1), (\tau_2, r_2) ..., (\tau_n, r_n)\}$ est un ensemble de tâches concret classées de *1* à *n*, où le r_i est la temps d'activation de la tâche τ_i. Il y a une relation naturelle plusieurs-à-un entre la tâche concrète et la tâche τ_i abstraite (non concrète). Nous disons que la tâche abstraite τ_i génère une tâche concrète (τ_i, r_i), et une tâche concrète (τ_i, r_i) ***est générée*** de la tâche τ_i. Cette relation se prolonge naturellement en une relation entre l'ensemble de tâches concrètes et l'ensemble (non concret) de tâches abstraites. Donné un ensemble de tâche abstraite $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ et un ensemble concret de tâche $\omega = \{(\tau_1, r_1), (\tau_2, r_2) ..., (\tau_n, r_n)\}$, alors que l'ensemble abstrait Γ ***génère*** l'ensemble concret de tâche ω et ω ***est généré*** de Γ.

Noter qu'un ensemble de tâches abstraites est ordonnançable si et seulement si les tâches peuvent être ordonnancées pour n'import quel temps d'activation. En revanche, chaque membre d'ensemble de tâches concrètes a un temps d'activation donné, et ayant qu'un ensemble de tâches concrètes est ordonnançable peut prouver seulement qu'il est ordonnançable avec ses temps d'activation donnés. L'ordonnancement « earliest deadline first » (EDF) est considéré comme un algorithme réussi puisqu'il est optimal pour l'ensemble de tâches préemptibles ou non. L'ordonnançabilité d'un ensemble de tâche non préemptif peut être déterminé par le théorème suivant [Jeffay91].

Théorème 1 [Jeffay91] : sachant que $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ est un ensemble de tâches sporadiques ou périodiques classées dans l'ordre non décroissant selon la taille de la période, (c.-à-d., pour n'importe quelle paire de tâches τ_i et τ_j, si $i>j$, alors $p_i \leq p_j$). Si Γ est ordonnançable alors il y a des relations suivantes :

(1) $\sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1$

(2) $\forall i, 1< i \leq n; \forall L, p_1 < L < p_i$:

$$c_i + \sum_{j=1}^{i-1} \left\lfloor \frac{L-1}{p_j} \right\rfloor c_j \leq L$$

Si Γ satisfait les conditions (1) et (2), alors l'algorithme d'ordonnancement non préemptif EDF peut ordonnancer n'import quel ensemble de tâches concretes, périodiques ou sporadiques, généré de Γ.

Dans ce qui suit, nous présenterons le système temps réel avec tolérance aux fautes sous la contrainte (m, k)-firm. Nous nous intéressons à garantir le niveau de la tolérance aux fautes en cas de surcharges, et les stratégies d'évaluation de performances ou d'attribution de ressources seront focalisées dessus.

Définition du modèle (m, k)-firm

Temps réel (m, k)-firm est une des manières appropriées de concevoir le système temps réel adaptatif qui fournit la gestion dynamique de QoS (quality of service) [ARTIST03]. Formellement, un système offrant contrainte (m, k)-firm exige d'un QoS minimum de *m* parmi *k* échéances consécutives de se accomplir dans le cas pire. Dans des cas généraux, plus que *m* échéances sont respectées car le système ne fonctionne pas toujours dans le pire cas.

Jusqu'ici, nous caractérisons un tâche τ_i comme $\tau_i = \{p_i, d_i, c_i, m_i, k_i\}$, avec $i = 1, 2, ..., n$, représentant l'indice des sources. Une tâche peut être un flux de messages à transmettre ou un travail à faire dans un système embarqué. p_i : la période pendant laquelle l'instance se produit, d_i : l'échéance associée (elle peut être omise si elle est

égale au périodique), c_i : le temps de service (exécution) de tâche sur le serveur et m_i et k_i représentent la contrainte (m, k)-firm.

Une tâche sous la contrainte (m, k)-firm peut se trouver dans un des deux états suivants : normal et échec dynamique [Hamdaoui95]. Figure 2 présente le diagramme d'état transition pour (2.3)-firm : où 1 dénote qu'une échéance est satisfaite et 0 une échéance ratée, respectivement. Ces états sont évalués selon l'historique du système ; chaque état qui est normal ou en échec dynamique dépend su service des trois dernières échéances. Le fait que l'échéance prochaine sera ratée ou respectée causera le passage du système à un autre état.

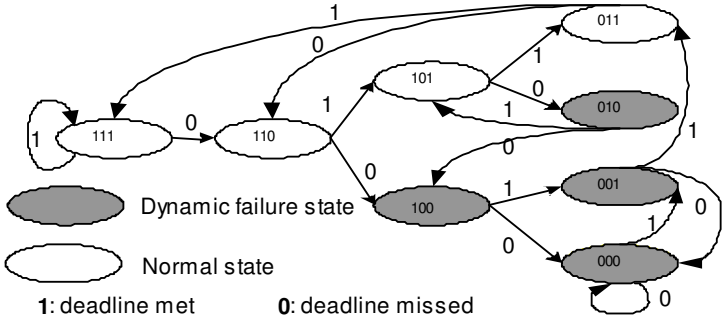


Figure 2 : diagramme de transition d'Etat avec (2.3)–firm

S'il y a moins de 2 échéances respectées, le système est dans l'état échec dynamique. Autrement, le système est dans l'état normal. Ce diagramme décrit la transition d'états concernant une tâche, cependant, comment ordonnancer parmi un ensemble des tâches sera discuté dans la section prochaine.

Après, nous nous limitons au cas de l'ordonnancement non préemptif dans un système ayant un processeur unique ; c'est-à-dire, nous supposons qu'un algorithme d'ordonnancement dans lequel une fois l'exécution d'une instance a commencé, il n'en interrompt pas pendant l'exécution. Nous limitons également l'ordonnancement sur un processeur unique sans temps d'oisif inséré ; ce qui signifie que l'algorithme d'ordonnancement ne permet pas au processeur d'être à vide s'il y a une instance qui a été invoquée mais n'a pas accompli. Pour éviter la redondance rédactionnelle, nous ne mentionnerons plus ces restrictions dans le reste du document.

Ordonnancement sous contraintes « (m, k)-firm »

Dans cette section, nous présenterons les travaux récents qui concernent l'ordonnancement d'un ensemble de tâches sous la contrainte (m, k)-firm. En même temps que les contraintes temps réel, les conditions suffisantes pour les divers algorithmes d'ordonnancements seront présentées. En outre, les discussions parmi les algorithmes des ordonnancements seront aussi effectuées.

Au niveau de priorité associée aux tâches, les algorithmes pour garantir la contrainte (m, k)-firm peuvent être classés dans *l'ordonnancement statique* ou *l'ordonnancement dynamique*, selon la stratégie d'ordonnancer. D'autre part, pour classer les instances à obligatoire ou facultative, on peut utiliser le pattern fixe ou pattern dynamique. Le pattern fixe ou dynamique convient à quelques applications spécialisées. Par exemple, pour le flux vidéo de MPEG, les paquets ont différents niveaux d'importance, c'est-à-dire quelques paquets peuvent être jetés et certains ne peuvent pas. Dans ce cas-ci, l'ordonnancement fixe devrait être exploité. Cependant, pour quelques applications, par exemple, le flux MP3, tous les paquets peuvent être considérés d'avoirs la même importance, donc le jette de paquet peut être faite dynamiquement seulement selon l'efficacité de l'utilisation de ressource.

Ordonnancement avec (m, k) pattern fixe

Le pattern fixe est un algorithme d'ordonnancement qui classe instances de tâche dans des obligatoire et des facultatives, tel que « skip-over », « sche_mkfirm », « deeply red pattern », EFP pattern, etc. Les instances obligatoires doivent être exécutées et accomplies avant leurs échéances, par contre les instances facultatives peuvent être fluxées sans exécution quand elles ne peuvent pas être accomplies avant leurs échéances. Quelque part, le pattern fixe sous contrainte (m, k)-firm s'appelle également (m, k)-pattern. (m, k)-pattern, dénoté par Π_i, est une séquence binaire $\Pi_i = \{\pi_{i1}, \pi_{i2}, \ldots \pi_{ik_i}\}$, qui dénote que $\tau_{i,j}$ est une instance obligatoire si $\pi_{ij}=1$ et elle est facultative si $\pi_{ij}=0$, et $\sum_{j=1}^{k_i} \pi_{ij} = m_i$.

Ordonnancement « skip-over »

Dans [Koren95], les auteurs considèrent le modèle « skip-over ». C'est un cas spécial d'ordonnancement du (m, k)-pattern où *m = k-1*. Clairement, c'est un cas spécial de (m, k)-pattern défini comme suit (6) :

$$\pi_{ij} = \begin{cases} 1 & 1 \le j < k_i - 1 \\ 0 & j = k_i \end{cases} \qquad j = 1, 2 \mathrm{L}\ \ k_i \qquad (1)$$

Dans [Koren95], l'instance obligatoire et facultative sont nommées comme « red task » et « blue task », respectivement. Le « red tasks only » et « blue when possible » sont une pattern statique et un pattern dynamique respectivement pour un ensemble de tâche sous la contrainte « skip-over ». Ils se diffèrent de l'intention de s'exécuter ou pas de l'instance facultative. L'algorithme de « red tasks only » n'ordonnance jamais une instance facultative, alors que l'algorithme de « bleu when possible » ordonnance l'instance facultative quand il n'empêchera pas les obligatoires.

Il est prouvé que le problème d'ordonnançabilité d'un ensemble de tâche périodique, avec le jette de temps en temps est NP-dur dans le sens faible. Une condition suffisante est présentée dans [Koren95] pour déterminer l'ordonnançabilité pour un ensemble de tâche de « red tasks only » sous priorité assignée selon « rate monotonic », qui est décrit comme suivant :

Théorème 2 [Koren95] : Donné un ensemble de tâches périodiques $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$, et où pour chacune i, $1 \le i \le n$, $\tau_i = (c_i, p_i, k_i)$, qui permet des sauts (skip), puis $L \ge \sum_{i=1}^{n} D(i,[0,L])$, pour tout L≥0 $D(i,[0,L]) = \left(\left\lfloor \frac{L}{p_i} \right\rfloor - \left\lfloor \frac{L}{p_i k_i} \right\rfloor \right) c_i$ est la condition suffisante d'ordonnancabilité de Γ.

Noter que cet algorithme ne peut pas être aisément appliqué au modèle (m, k), parce qu'il est juste un cas spécial de (m, k)-firm, comme (k-1, k)-firm.

Modèle Sche_mkfirm.

Dans [Ramanathan99], un (m, k)-pattern est proposé, qui définit les instances obligatoires avec le formulaire ci-dessous qui ordonnance les instances obligatoires

avec « rate monotonic », cet algorithme s'appelle « Sche_mkfirm », et est défini comme suit :

$$\pi_{ij} = \begin{cases} 1 & if \quad j = \left\lfloor \left\lceil \frac{j \times m_i}{k_i} \right\rceil \times \frac{k_i}{m_i} \right\rfloor \\ 0 & otherwise \end{cases} \qquad j = 1,2,\mathrm{L}\ k_i \tag{2}$$

Pour ceci (m, k)-pattern d'un tâche, son (m, k)-pattern est fixe dès que sa contrainte (m, k)-firm est définie. Une condition suffisante d'ordonnançabilité est donnée:

Théorème 3 [Ramanathan99] : considérons un ensemble de *n* tâches periodiques ou sporadiques $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$, où $\tau_i = (c_i, p_i, m_i, k_i)$, classées dans l'ordre non décroissant de période tel que $p_1 < p_2 < ... < p_n$. Définissons les termes ci-dessous :

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

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

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

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

Si pour un intervalle quelconque *t*, $min_{t \in R_i} W_i(t)/t \le 1$ pour tous $1 \le i \le n$, alors l'algorithme RM respect de façon déterministe toutes les contraintes (m_i, k_i)-firm.

Comme la définition, l'algorithme Sche_mkfirm associe toujours les premières instances des tâches en tant qu'obligatoires, alors le temps de réponse dans le pire cas de la chaque tâche est toujours la première instance de chacune. Ceci s'appelle « worst-case interference point » (WCIP) de tâche τ_i, qui est un intervalle du temps dans lequel le nombre d'instances obligatoires est le plus grand parmi tous les intervalles avec la même longueur. C'est juste la stratégie du théorème ci-dessus pour ga-

rantir chacune première instance de chaque tâche afin de donner la condition suffisante pour garantir la contrainte (m, k)-firm.

« Deeply-red (m, k) » pattern

Dans [Quan00], une (m, k)-pattern extrême appelé « deeply-red » (m, k)-pattern est donné, qui est défini comme suivant :

$$\pi_{ij}^{r} = \begin{cases} 1 & 1 \le j \le m_i \\ 0 & m_i < j \le k_i \end{cases} \qquad (3)$$

Le théorème 4 dans [Quan00] qui a prouvé ce « deeply-red » (m, k)-pattern est un cas critique comme suit :

Théorème 4 [Quan00] : pour un ensemble de tâches Γ avec $r_i=0$, si les instances obligatoires définies par « Deeply-red » (m, k)-pattern sont ordonnançable, les instances obligatoires dérivées de tous les autres (m, k)-pattern sont également ordonnançable.

Observer que « Deeply-red » (m,k)-pattern a centralisé le WCIP au temps d'activation et peut être considéré en tant que le pire cas du (m, k)-pattern. Si « Deeply-red » (m, k)-pattern est ordonnançable, tous les autres modèles tels que le pattern de « Sche_mkfirm », « skip-over » pattern sont faisables aussi. Cependant, ce Deeply-red (m,k)-pattern a une utilisation de ressource basse, en plus, dans beaucoup de cas, il exige la même ressource pour garantir un ensemble de tâches sous la contrainte (m, k)-firm que l'ensemble de tâches sous contrainte HRT. Ceci cause la perte d'efficacité pour « Deeply-red » (m,k)-pattern, qui sera discuté plus tard.

« Enhanced Fixed-Priority » (m,k)-pattern

Dans [Quan00], en décalant le pattern de Sche_mkfirm, un nouveau (m,k)-pattern appelé « Enhanced Fixed-Priority » a été donné comme ci-dessous :

$$\pi_{ij} = \begin{cases} 1 & if \quad j = \left\lfloor \left\lceil \frac{(j+s_i)\times m_i}{k_i} \right\rceil \times \frac{k_i}{m_i} \right\rfloor - s_i \\ 0 & otherwise \end{cases} \qquad j = 1,2,\mathrm{L}\ k_i \tag{4}$$

En fait, EFP (m,k)-pattern se produit en décalant à droit le pattern Sche_mkfirm, il fait quelques efforts pour éviter WCIPs de chaque tâche qui se concentre au temps de démmarage. La valeur s_i est configurée pour séparer au plus WCIPs parmi les tâches. Intuitivement, cet EFP (m,k)-pattern peut ne jamais être plus mauvais que le pattern Sche_mkfirm, et on lui a montré expérimentalement que l'algorithme EFP se comporte largement meilleur que l'algorithme Sche_mkfirm en termes d'ordonnançabilité du système. Cependant, la configuration prend un calcul compliqué et essaye toutes les possibilités, qui fait EFP (m,k)-patternnt non approprié aux applications temps réel adaptatives ou à la transmission en raison de l'environnement instable.

Ordonnancement (m, k) pattern dynamique

Noter que l'algorithme « blue when possible » pour la contrainte « skip-over » peut être considéré en tant qu'ordonnancement dynamique pour tâche « skippable », cependant, « Distance based priority » (DBP) est un algorithme général dynamique d'ordonnancement de (m, k)-firm.

DBP (priorité basée sur la distance à l'état d'échec)

DBP a été premièrement présenté par Hamdaoui et Ramanathan [Hamdaoui95] comme un mécanisme de priorité association dynamique pour les tâches sous la contrainte (m, k)-firm dans un système MIQSS.

L'idée fondamentale de l'algorithme DBP est tout à fait simple: le plus proche un flux à un état d'échec, le plus haute sa priorité est. Un état d'échec se produit quand la contraint (m, k)-firm d'un flux est violé, c.-à-d., il y a plus que *k-m* échéances ratées dans la dernière fenêtre de longueur *k* d'instances.

Pour connaître l'état actuel d'une tâche, nous devons examiner l'historique de l'exécution des dernières *k* instances. Si nous associons « *1* » à une instance avec

l'échéance respectée et « *0* » à une instance avec l'échéance ratée, comme (m, k)-pattern, cet historique alors est entièrement décrite par un mot de *k* binaires appelé la k-séquence.

La k-séquence est un mot de *k* bits classés du plus récent au plus ancien dans lequel chaque binaire garde la mémoire si l'échéance est ratée (bit = 0) ou respectée (bit =1), où le bit à extrémité gauche représente le plus ancien. Chaque nouvelle instance arrivée cause un décalage de tous bits vers la gauche, le bit à l'extrémité gauche va sortir du mot de la k-séquence et n'est plus considéré, alors que le bit à l'extrémité droite sera un « 1 » si l'échéance est respectée (c.-à-d. il a été servi à temps) ou un « 0 » autrement.

Figure 3 donne un exemple avec la contraint (3.5)-firm.

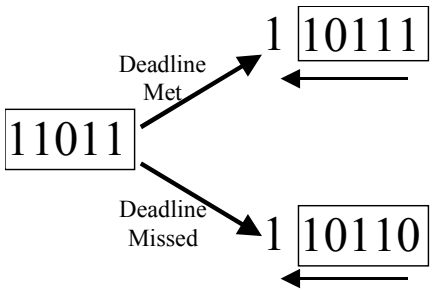


Figure 3: Évolution de k-séquence

Ainsi pour chaque flux, qui exige une contrainte (m_i, k_i)-firm, la priorité est assignée selon le nombre d'échéances ratées consécutives qui mène le flux à violer sa contrainte (m_i, k_i)-firm. Ce nombre d'échéances ratées est mentionné comme la distance de l'état actuel à l'état d'échec. Priorité assignée par DBP à une instance donnée par la distance de k-séquence courante à un état d'échec. Formellement, selon [Hamdaoui95] la priorité est évaluée comme suit :

Considérons que $s_j=\left(\delta^j_{i-k_j+1},\Lambda,\delta^j_{i-1},\delta^j_i\right)$ dénote l'état du *k* instances précédentes consécutives de la tâche τ_i, $l_j(n, s_j)$ dénote la position (de la droite) de la $n^{ième}$ échéance respectée (ou de 1) dans s_j, alors $(i+1)^{ième}$ instance de τ_j est associée à une priorité donnée par :

$$P\_DBP_{j,i+1} = k_j - l_j\left(m_j, s_j\right) + 1 \qquad (5)$$

Quand un flux est déjà dans l'état d'échec (c.-à-d., moins que *m 1s* dans la k-séquence), la priorité la plus élevée « 0 » est attribuée. Par exemple, considérons un flux avec la contrainte (3.5)-firm, l'instance courante $\tau_{j,i+1}$ est attribuée la priorité 2 si le mot de 5-séquence est (11011), et est attribuée la priorité 3 si 5-séquence courante est (10111).

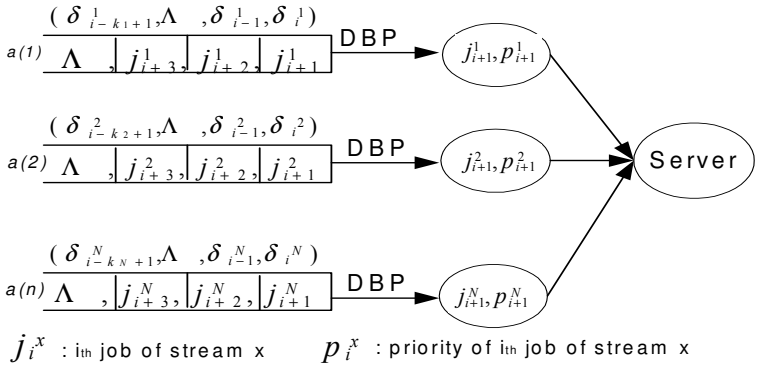


Figure 4 : DBP pour attribution de priorités à des instances en tête des files d'attente

Un des problèmes au sujet du DBP est qu'il attribue les priorités seulement en considérant la contrainte (m_i, k_i)-firm sans comparer aux autres tâches qui partagent le même serveur. Ce comportement d'autoréférence peut mener à une situation où plus d'un flux obtient la même priorité en même temps ; dans ce cas-ci, un autre algorithme pour choisir parmi eux devrait être défini. Dans la partie suivante, quand les instances de tâches au début des files d'attente ont la même priorité, EDF est employé par défaut.

Noter que les algorithmes (m, k)-firm dépendent seulement de la ration de *m* sur *k* de chaque tâche, sans considération des périodes et des temps d'exécution des tâches. En outre, seulement la soustraction de *k* et *m* est prise en compte, unilatéralement. Par exemple, les contraintes (2,3)-firm et (3,4)-firm seront attribuées avec la même priorité pour la première instance. Dans la section prochaine, la contrainte de fenêtre fait quelques efforts pour améliorer ceci en employant le quotient de la tolé-

rance à la perte dans une fenêtre fixe, mais ne prend pas en compte encore des périodes de tâches et des temps d'exécution.

Contrainte de fenêtre « Window constraint »

Similaire à la contrainte (m, k)-firm, une autre contrainte importante qui fait le objet de rejet sélectif dynamique, appelée contrainte de fenêtre (Window-contraint), sera présentée [West99]. La « Window-contrainte » est définie par une valeur $W_i = x_i/y_i$, où le numérateur de fenêtre, x_i, est le nombre de paquets qui peuvent être perdus ou transmis en tard dans chaque fenêtre fixe de taille y_i (le dénominateur de fenêtre), les paquets arrivés consécutivement du même flux τ_i. Par conséquent, pour les paquets arrivés du même flux τ_i, un minimum de y_i-x_i paquets doit être accompli leur services avant leurs échéances, autrement une violation de service se produit. À tout moment, tous les paquets dans le même flux, τ_i, ont la même contrainte de fenêtre, W_i. Ainsi chaque paquet dans un flux, τ_i a une échéance qui se décale par un intervalle fixe, p_i, de son prédécesseur. Après transmission d'un paquet du τ_i, l'ordonnanceur va modifier la contrainte de fenêtre de τ_i aussi pour les autres flux dont les échéances des paquets sont ratées. En conséquence, la contrainte de fenêtre d'origine d'un flux (τ_i), W_i, différera de sa contrainte de fenêtre courante, W_i'. Observer que la contrainte de fenêtre d'un flux peut également être considérée comme la tolérance aux pertes.

Comparaison entre (m, k)-firm et contrainte de fenêtre

La contrainte de fenêtre essaie explicitement de fournir la garantie de service dans une fenêtre fixée. C'est-à-dire, la contrainte de fenêtre suppose que la contrainte de fenêtre originale indiquée pour chaque flux et définit le requit de service sur les groupes d 'échéances non-superposée. Réciproquement, la contrainte (m, k)-firm exige sur tous k instances consécutives, qui signifie qu'une fenêtre glissante qui commence à n'importe quelle instance de tâches.

Cependant, il est possible de déterminer une contrainte de fenêtre glissante correspondante d'une contrainte de fenêtre fixe spécifiée par [West00] [West04], et

vice versa. Comme indiqué précédemment, la contrainte de fenêtre exige, qu'il n'y a pas plus de x_i échéances ratées dans une fenêtre fixe de y_i dans τ_i. Ce qui implique qu'il faut garantir le respect des échéances d'un minimum de $m_i = y_i - x_i$ sur une fenêtre fixe de $k_i = y_i$. S'il n'y pas plus de x_i échéances ratées dans une fenêtre fixe de y_i, il n'y a pas plus de $2x_i$ échéances ratées dans une fenêtre glissante de $y_i + x_i$. Donc, ceci signifie qu'en se respectant m_i échéances dans une fenêtre avec taille k_i, alors il est possible de transformer la contrainte de fenêtre x_i/y_i en (y_i-x_i, y_i+x_i)-firm, et transformer aussi la contrainte (m_i, k_i)-firm en contrainte de fenêtre de $2(k_i-m_i)/(2k_i - m_i)$. La preuve est omise pour la brièveté.

Bien que la contrainte de fenêtre et la contrainte (m, k)-firm peuvent être transformée mutuellement, elles sont essentiellement différentes. Brièvement, la contrainte de fenêtre est moins restreinte que la contrainte (m, k)-firm sous le même taux de pertes. Par exemple, un tâche sous (1, 2)-firm contrainte et une tâche sous la contrainte de fenêtre 1/2 peuvent tolérer la même perte de 50%, mais la trace de l'ordonnancement suivante peut être acceptée par la contrainte de fenêtre 1/2 mais pas par (1, 2)-firm contrainte (où la fenêtre est glissée à 2^{ieme} et 3^{ieme} instances). Cette trace d'ordonnancement peut seulement satisfaire la contrainte (1,3)-firm mais il n'est pas nécessaire pour la plupart d'endroit où la fenêtre glisse. En fait, ceci peut être obtenu facilement selon la méthode de transformation mentionnée ci-dessus.

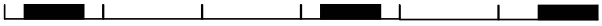


Figure 5 : contrainte de fenêtre 1/2 et contrainte (1, 3)-firm

Ordonnancement dynamique « Window-contrainte »

Dans cette section, nous présenterons les algorithmes d'ordonnancement sous contrainte de fenêtre. R. West a proposé deux versions d'ordonnacement pour la contrainte de fenêtre, qui s'appelle « Dynamic Window Constraint Scheduling » (DWCS). Afin de les distinguer, nous allons appeler en tant que DWCS-1 et DWCS-2. Ces deux algorithmes s'emploient dans le modèle MIQSS et donnent la priorité aux paquets basés sur la valeur courante de leur tolérance aux pertes. DWCS modifie la

priorité dynamiquement puisque la tolérance aux pertes d'un flux change après une échéance ratée ou respectée.

DWCS-1

Le tableau 1 montre les règles pour ordonnancer des paires de paquets parmi différents flux. Rappeler que tous les paquets dans le même flux sont alignés avec l'ordre d'arrivée. La priorité est attribuée à un flux selon la tolérance aux pertes, où $W_i = x_i/y_i$ est la tolérance aux pertes courante pour tous les paquets dans le flux τ_i. Si deux flux ont la même tolérance aux pertes différente de zéro, « earliest deadline first » s'emploie (le EDF). Si deux paquets ont la même tolérance aux pertes différente de zéro et la même échéance, la priorité est donnée au paquet qui a le perte-numérateur plus bas (x_i), car il peut tolérer moins de pertes consécutives. Si deux paquets ont la tolérance aux pertes égale à zéro et leurs perte-dénominateurs sont aussi zéro, ils sont ordonnancé par EDF, autrement ils sont ordonnancés selon perte dénominateur le plus élevé. Dans tous autres cas, la priorité est attribuée selon « First In First Out ».

Pairwise Packet Ordre (1)
La plus basse tolérance aux pertes d'abord
La même tolérance aux pertes différente de zéro, l'ordre de EDF
La même tolérance aux pertes et échéances différentes de zéro, ordre de le plus bas numérateur de contrainte de fenêtre d'abord
Tolérance aux pertes et dénominateurs sont zéro, l'ordre de EDF
La tolérance aux pertes est zéro, le dénominateur de contrainte de fenêtre le plus élevé d'abord
Tous autres cas : « first-come-first-serve »

Tableau 1 : Priorité DWCS-1 parmi des paires de paquets

Chaque fois qu'un paquet dans le flux τ_i est transmis, puis la tolérance aux pertes de τ_i est ajustée. De même, la tolérance aux pertes d'autres flux sont ajustée

seulement si tous paquets de ces flux ratent les échéances en raison de délais des files d'attente.

Comparaisons entre DBP, DWCS et EDF

On compare le taux de pertes et le effet de état d'échec parmi les politiques DBP, DWCS et EDF [Striegel03].

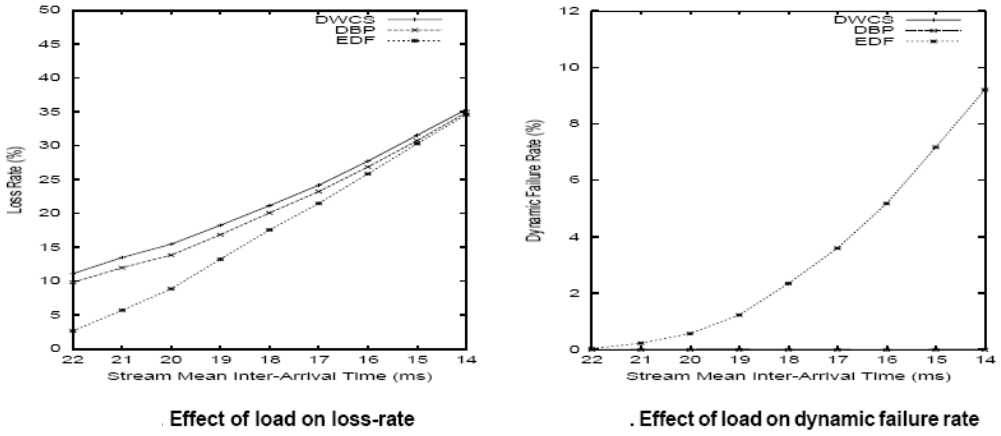


Figure 6 : comparaison de taux de taux de perteet d'échec parmi DWCS, DBP et EDF

Figure 6 démontre le taux de perte expérimental des paquets produit par les ordonnancements de EDF, de DBP, et de DWCS. Noter que le taux de perte contient tous les paquets perdus, et le taux d'échec dynamique mesure seulement ces paquets qui ont eu un impact négatif (violation de fenêtre de perte) sur la QoS perçue par clients. Il peut voir dans les graphiques, que DBP et DWCS sont meilleurs que EDF en termes de taux d'échec dynamique ; cependant, ils se comportent beaucoup mauvais en termes de taux de perte. Vu de Figure 7, taux de perte n'est pas nécessairement corrélés avec le taux d'échec dynamique. Noter que DBP et DWCS peuvent réduire le taux de l'état d'échec, mais ratent plus les échéances. C'est la la raison pour laquelle dans [West01], un nouvel ordonnancement de DWCS basé sur le temps réel dur a été proposé, et ce qui sera présenté dans la prochaine section.

DWCS-2 : ordonnancement HRT en utilisant DWCS

Malheureusement, au cas de sous charge, EDF respecte plus d'échéances. Cependant, l'algorithme DWCS-1 est encore meilleur que EDF dans des cas de surcharge où il est impossible de respecter toutes les échéances.

Dans [West00] [West04], on a proposé un autre algorithme d'ordonnancement dynamique de DWCS (Dynamic Window-Constraint Scheduling), tel que la garantie de temps réel dur puisse être faite sur des flux de paquet en tolérant *x* échéances ratées parmi des *y* instances. Le tableau 2 suivant décrit les règles de la version révisée de DWCS pour des paires de commande de paquets.

Pairwise Packet Ordering (2)
EDF (earliest deadline first)
Le même échéance, la plus basse tolérance aux pertes d'abord
La même tolérance aux pertes et échéances différentes de zéro, ordre du plus bas perte-numérateur d'abord
La même échéance et contraint-fenêtre est zéro, l'ordre de dénominateur du EDF d'abord
La même échéance et contraint-fenêtre n'est pas zéro, ayant perte-dénominateur le plus élevé d'abord
Tous autres cas : « first-come-first-serve »

Tableau 2 : DWCS-2

La priorité est attribuée aux paquets des flux selon les règles montrées dans le tableau 2. Ce tableau diffère de la tableau originale de DWCS-1 [West99]. La différence inhérente est que les deux premières lignes dans le tableau sont renversées : la table originale compare d'abord des paquets basés sur contrainte de fenêtre courantes donnant et attribue la priorité au paquet avec la contrainte de fenêtre le plus petit (numérique-évaluée). S'ils ont la même contrainte de fenêtre, le paquet avec l'échéance la plus tôt est choisie (EDF).

Ajustement de tolérance aux pertes de DWCS

On décrit maintenant comment des tolérances aux pertes sont ajustées. La tolérance aux pertes d'origine est dénotée par x_i/y_i pour tous les paquets de flux τ_i. La tolérance aux pertes courante est dénotée par x'_i/y'_i pour tous les paquets alignés de flux τ_i. On dénote le perte-numérateur courant avec x'_i, et on dénote le perte-numérateur courant avec x_i pour des paquets dans le flux τ_i. On dénote perte-dénominateurs courant et original avec y'_i et y_i, respectivement. Avant qu'un flux de paquet soit servi, ses tolérances aux pertes courantes et originales sont égales. Parmi tous les paquets dans une file d'attente d'un même flux, τ_i, le paquet le plus récemment transmis avant son échéance, on ajuste les numérateurs et les dénominateurs de perte comme suit :

(a) ajustement de contrainte de fenêtre quand un paquet dans τ_i est servi avant son échéance :

```
if (y'_i > x'_i) then y'_i = y'_i − 1;
else if (y'_i = x'_i) and (x'_i > 0) then
        x'_i = x'_i − 1; y'_i = y'_i − 1;
if (x'_i = y'_i = 0) or (τ_i is tagged) then
        x'_i = x_i; y'_i = y_i;
if (S_i is tagged) then reset tag;
```

Figure 7: modification de contrainte de fenêtre après le service d'un paquet

(b) ajustement de contrainte de fenêtre quand une échéance d'un paquet dans τ_j | j ≠i est ratée :

Le comportement d'un ou plusieurs paquets en retard, est ajustée suivant les indications dans Figure 8. En l'absence d'un essai de faisabilité, il est possible que les violations de contrainte de fenêtre puissent se produire. Une violation se produit réellement quand $W'_j = x'_j/y'_j \mid x'_j = 0$, et en plus une autre échéance de paquet dans τ_j est ratée. Avant que τ_j soit servie, x'_j reste zéro, alors que y'_j est augmenté par une constante, ε, chaque fois qu'un paquet dans τ_j rate son échéance. L'exception à cette règle

est quand $y_j = 0$ (et, plus spécifiquement, $W_j = 0/0$). Ce cas spécial permet DWCS à servir toujours des flux dans l'ordre du EDF, si une telle politique de service est désirée.

if ($x'_j > 0$) then
    $x'_j = x'_j - 1$; $y'_j = y'_j - 1$;
    if ($x'_j = y'_j = 0$) then $x'_j = x_j$; $y'_j = y_j$;
else if ($x'_j = 0$) and ($y_j > 0$) then
    $y'_j = y'_j + \varepsilon$;
Tag S_j with a violation;

Figure 8 : ajustement de contrainte de fenêtre après une échéance ratée

Condition suffisante pour DWCS-2

Pour DWCS-2, une condition suffisante et nécessaire a été donnée pour un ensemble de flux, qui peut fournir la garantie d'ordonnançabilité déterministe.

Théorème 5 [West04] : Considérer un ensemble de flux $\Gamma = \{\tau_1, ..., \tau_n\}$, où $\tau_i \in \Gamma$ est défini par le 3-tuple avec les mêmes contrainte de fenêtre « window-constraint » ($c_i = K$; $p_i = qK$; $w_i = x_i/y_i$). Cet ensemble de flux est ordonnançable par DWCS-2 si et seulement si le facteur d'utilisation, $U = \sum_{i=1}^{n} \frac{(y_i - x_i)}{q y_i} \leq 1.0$.

Commente DWCS-2 :

Au premier regard de la condition suffisante et nécessaire pour DWCS-2, il est très passionnant parce que l'utilisation de ressource peut arriver à 100%, tel que DWCS-2 est optimal pour l'ordonnancement avec le rejet sélectif de paquets dans la transmission en réseaux. Cependant, nous avons constaté que ce résultat est efficace seulement dans des cas spéciaux. Ses limitations et imperfections seront discutées dans les points suivants :

- vue du théorème, l'approche DWCS-2 exige que tous les flux doivent avoir le même temps d'exécution « unit size execution time), en tant que $c_i=K$, et la même période, en tant que $p_i=qK$. Cette restriction n'est pas appropriée aux communications de réseau, puisque les applications temps réel ont habituellement la taille de paquet différente.

- On démontre DWCS-2 dans tableau 2, qui mises à niveau EDF comme le première règle d'ordonnancement, tels que DWCS-2 s'invalide comme une modification du EDF. Ceci cause du fait que les règles au-dessous du EDF peuvent seulement être prises en compte en condition des échéances égales. Normalement, dans HRT-EDF ordonnancement, « rate monotonic » (RM) ou l'autre méthode de priorité fixe (PF) est employée pour associer la priorité en cas d'échéances égales. Pareillement, la contrainte-fenêtre (perte-taux) joue le même rôle que RM et PF, et elle pertes ses caractères du jette sélectifs. En fait, si une violation de contrainte-fenêtre se produit par EDF, cette violation se produit par DWCS-2 ordonnancement aussi.

- afin de valider l'ordonnancement DWCS-2, toutes les périodes de flux doivent être synchronisé. Autrement, les flux seront ordonnancés comme HRT sous EDF, puisque EDF est le premier dans les règles de ordonnancement de DWCS-2. Ce fait peut être montré par un exemple montré dans Figure 9 et Figure 10. Figure 9montre un cas où il y a deux flux paramétrés comme : c=K, p=K, la contrainte de fenêtre ½ et le temps d'activation synchronisé, alors la contrainte de fenêtre peuvent être satisfaite.

Figure 9 : HRT-DWCS est valide pour les flux synchronisés

Figure 10 : HRT-DWCS est invalide pour les flux non synchronisés

Cependant, Figure 10 montre que si le flux τ_1 est décalée un peu plus tôt que dans la Figure 9, alors les deux même flux seront programmés selon EDF, tel que τ_1 est toujours servie, τ_2 n'est jamais. La raison est que DWCS-2 modifie l'ordonnancement DWCS original en changeant les deux premières règles, afin d'améliorer le taux de échéance ratée. Noter que tous les flux dans les systèmes de communication ont gigues et les sources sont activées arbitrairement, donc la synchronisation ne peut pas être bien réalisée dans le vrai système. Par conséquent, la contrainte-fenêtre est rarement prise en compte par DWCS-2 pendant l'ordonnancement, alors que les flux sont programmés selon HRT-EDF.

Les points ci-dessus ont montré que DWCS-2 peut seulement fournir l'utilisation élevée dans quelques cas extrêmes spéciaux, et il peut être considéré comme un algorithme d'ordonnancement de temps réel dur. En fait, dans [West99], il est également identifié par l'auteur puisque DWCS-2 s'appelle en tant que l'ordonnancement temps réel dur en utilisant DWCS. Cependant, nous espérons l'algorithme d'ordonnancement qui devrait sélectivement ordonnancer les instances selon le niveau de la tolérance aux fautes en état de la surcharge, et ce caractère est perdu en mettant EDF en tant que la première règle. (D'ailleurs, dans le prochain chapitre, nous nous discuterons que la preuve de cette condition suffisante n'est pas correcte du tout, et le reconstruirons dans l'annexe A).

Conclusion

Dans ce chapitre, nous avons présenté les modèles basiques d'analyse et avons montré les résultats fondamentaux. La motivation de contrainte (m, k)-firm et contrainte de fenêtre est la fonction de rejet sélectif parmi les instances (paquets) en cas de surcharges. Observer que l'algorithme DWCS-2 peut réaliser l'utilisation de

ressource à 100%, qui semble optimale, mais elle est seulement valable sous certaines conditions extrêmes spéciales. Sachant que (m, k)-firm et contrainte de fenêtre sont les contraintes de dégradation contrôlée pour le système temps réel adaptatif, les conditions extrêmes exigées par DWCS-2 ne sont pas raisonnables et réalisables dans des vrais systèmes. Réciproquement, « skip-over », « deeply-red (m,k)-pattern », « Sche_mkfirm » et « EFP (m, k)-pattern » ordonnancent l'ensemble de tâches sans condition spéciale par rapport à DWCS-2, donc (m,k)-patterns ont gagné la généralité. D'ailleurs, les modèles « Sche_mkfirm » et « EFP (m,k)-pattern » sont fixés à l'avance pour que les instances à jeter soient fixées et en juste proportion espacée, ainsi que leur coûts de réalisation ne sera pas considérable. Cependant, l'utilisation de ressources de ces approches d'ordonnancement générales est très basse. Ce fait sera détaillé dans le chapitre 3 avec notre analyse et conclusion plus approfondies. Dans le chapitre prochain, nous allons rechercher dans les algorithmes d'ordonnancement dynamiques afin d'obtenir une utilisation de ressource plus élevée. Remarquer que DBP et DWCS-1 n'ont pas d'exigences sur les paramètres des ensembles de tâches (ou flux). Mais juste avant notre étude, il n'y a pas encore de conditions suffisantes sur DBP ou DWCS-1.

Chapitre 2

Condition suffisante d'ordonnançabilité sous NP-DBP-EDF

Dans le chapitre précédent, nous avons présenté deux méthodes d'ordonnancements générales, appelées DBP et DWCS-1, qui peuvent jeter les instances sélectivement. Dans ce chapitre, nous proposons une condition suffisante pour déterminer l'ordonnançabilité d'un ensemble de tâches à contrainte (m, k)-firm ordonnancées avec l'algorithme de fixation de priorités DBP.

Comme la contrainte (m,k)-firm peut être transformée en une contrainte « windows-constraint », et vice versa, nous nous concentrerons seulement sur comment garantir la contrainte (m,k)-firm avec DBP. L'efficacité de la condition suffisante sera examinée en termes de ressources requises par rapport à la contrainte HRT.

Motivation de NP-DBP-EDF

Normalement, les systèmes temps réel à contraintes strictes sont conçus et dimensionnés en considérant la situation qui conduit à la charge de travail maximale ce qui implique des exigences considérables en termes de ressources matérielles. Cependant, ce pire cas ne se produit que rarement et donc les ressources ne sont que rarement utilisées. Une solution est de concevoir le système en considérant un cas moyen. Cette solution peut convenir à une sous-classe des systèmes temps réel mous (SRT) qui exigent seulement une garantie statistique sur les échéances. Cependant, pour d'autres systèmes temps réel qui se trouvent dans le domaine du multimédia et du contrôle-commande, la seule garantie statistique des échéances peut être inacceptable. Nous avons donc besoin de spécifications sur la distribution des échéances ratés

[Bernat01] ; c'est pour cette raison que le modèle (m, k)-firm est employé [Hamdaoui95]. Typiquement, pour un même taux d'échéances ratées, une application temps réel peut tolérer des échéances ratées non consécutives bien mieux que des échéances ratées consécutivement. On dit qu'un système est sous la contrainte (m, k)-firm temps réel s'il exige la garantie que au moins *m* échéances soient respectées pour n'importe quel *k* instances consécutives d'une tâche.

Beaucoup de travaux ont proposés de nouveaux algorithmes d'ordonnancement pour garantir la contrainte (m, k)-firm [ZWang02]. Deux classes d'ordonnancement peuvent être distingué : l'ordonnancement dynamique et l'ordonnancement statique. DBP (Distance Based Priority) [Ramanathan99] et DWCS (Dynamique Window-constraint Scheduling) [West00] sont des ordonnancements dynamiques. L'affectation des priorités en ligne est effectuée selon l'état actuel du système. ERM (Enhanced Rate Monotonic) [Ramanathan99] et EFP (Enhanced Fixed-Priority) [Quan00] sont statiques puisqu'ils ordonnancent les instances hors-ligne en utilisant un pattern statique pour classer les échéances obligatoires et facultatives. Il faut noter que, comme pour le temps réel dur, une condition suffisante d'ordonnançabilité est naturellement nécessaire pour assurer une garantie déterministe de type (m, k)-firm. Il y a des conditions suffisantes pour ERM, EFP et DWCS [West00] [West04], mais il n'y a aucune condition similaire proposée pour DBP jusqu'à maintenant.

Dans ce chapitre, nous considérons seulement l'algorithme d'ordonnancement dynamique DBP pour un ensemble de tâche sous contraintes (m, k)-firm. Le système devrait pouvoir s'adapter à la variation de quantité de travail (par exemple dans les réseaux gérant la QdS par le contrôle d'admission) en profitant de la possibilité de jeter jusqu'à *k-m* instances consécutives pendant une période de surcharge du système. Par conséquent, dans ce contexte, l'ordonnancement hors-ligne n'est tout simplement pas approprié. En outre, une politique d'ordonnancement dynamique peut permettre une meilleure utilisation des ressources disponibles en général. Finalement, nous insistons sur l'importance de jeter instances des tâches qui dans tous les cas ne pourront pas être accomplies avant leur échéance par le système. En fait, la situation de surcharge cause certains non-respect d'échéances, et supprimer une partie des ins-

tances de tâche (de préférence celles avec des échéance ratées) peut permettre une performance meilleure du système. Dans notre travail, nous rejetons dynamiquement des instances et c'est ce qui diffère par rapport aux travaux classiques sans rejets (par exemple [Ramanathan95], [Bernat01], [Bernat03]).

Nous avons fait le choix d'une politique dynamique et étudierons DBP car c'est une technique efficace [Poggi03] mais pour laquelle il n'existe pas de condition suffisante d'ordonnançabilité ce qui est indispensable pour les applications visées. Pour DWCS, une telle condition a été proposée dans [West04]; mais son domaine d'application est très limité puisque les tâches doivent avoir le même temps d'exécution et le même taille des périodes. Nous le verrons dans la suite, nous pourrons trouver une condition plus générale en utilisant DBP. Comme nous voulons obtenir un résultat applicable à l'ordonnancement des tâches et à l'ordonnancement des paquets sur un réseau, nous nous limitons à l'ordonnancement non préemptif. Nous nous plaçons dans le cadre des études antérieures [Hamdaoui95], [West04], et considérons un ordonnancement à priorité avec un arbitrage EDF en cas de priorités égales.

Dans ce chapitre nous nous concentrons sur NP-DBP-EDF (Non preemptive – Distance Based Priority – Earliest Deadline First) et proposons une condition suffisante hors-ligne pour évaluer l'ordonnançabilité d'un ensemble de tâche sous l'algorithme DBP pour la contrainte (m, k)-firm. En outre, l'ordonnançabilité peut également être déterminée en-ligne dans un intervalle limité de temps si toutes les dates d'activation sont indiquées. Les simulations prouvent l'efficacité de la condition suffisante hors-ligne en termes de taux d'utilisation des ressources, pour une utilisation dans des système de contrôle-commande.

Bien que nous soyons principalement intéressés à prévoir l'ordonnançabilité a priori (ie. avant l'exécution du système) d'un ensemble de tâches, nous constatons que la condition suffisante en ligne pour déterminer l'ordonnançabilité de NP-DBP-EDF est beaucoup plus efficace que la condition hors-ligne pour ce qui est de l'utilisation des ressources. En particulier, l'ordonnancement DBP avec garantie hors-ligne peut dans certains cas exiger la même quantité de ressource que l'ordonnancement sous contrainte HRT (la raison théorique est montrée dans l'annexe

B). Par ailleurs, nous avons prouvé dans notre rapport [Li03] que le politique DBP peut se trouver dans un état d'échec (i.e. contrainte (m,k) non-respectée) même avec un taux utilisation de la ressource arbitrairement bas. Par conséquent, nous concluons qu'un faible taux d'utilisation des ressource est inévitable pour la contrainte (m, k)-firm dans le cas général. Ceci nous conduit à la recherche de la raison théorique de ce faible taux d'utilisation, afin de trouver une solution efficace.

Chapitre 3

Analyse des causes du faible taux d'utilisation des ressources

Dans les chapitres précédents, nous avons constaté que la contrainte (m, k)-firm n'est pas efficace en termes d'utilisation de la bande passante. Dans ce chapitre, on se concentrera sur les raisons théoriques de cette utilisation basse des ressources et on cherchera une contrainte plus efficace, compatible avec un rejet sélectif des paquets et les politiques d'allocation de ressource existantes.

D'abord, nous aurons une analyse approfondie de la condition suffisante d'utilisation de la ressource. Intuitivement, un ensemble de tâches sous la contrainte (m, k)-firm devrait soumettre une quantité de travail moindre, et exiger ainsi moins de ressources. Cependant, dans le chapitre précédent il est montré que la contrainte (m, k)-firm ne peut généralement pas économiser la ressource par rapport à HRT quand le but est de fournir des garanties déterministes. Encore plus défavorable, on sait que la contrainte (m, k)-firm peut toujours être violée pour un ensemble de tâches avec une quantité de travail arbitrairement basse [Quan00] [Mok01] [Li03].

Le problème général d'ordonnançabilité temps réel est prouvé comme NP-Difficile au sens fort (voir par exemple [Jeffay91] [Mok01] [Garey77,78]) et les travaux futurs dans ce domaine feront face à la question de l'indécidabilité. Cette question de la complexité algorithmique sera discutée dans la suite et les résultats principaux seront rappelés. Ensuite, nous décrivons les perspectives de recherches qui nous paraissent prometteuses pour les problèmes d'ordonnancement NP-Difficile. A la lumière de ces perspectives, nous récapitulons les stratégies existantes

d'ordonnancement temps réel qui ont pour objectif de réaliser une utilisation de ressource plus élevée.

Nous illustrons le phénomène d'utilisation faible des ressources dans le domaine de l'ordonnancement temps réel et mettons en évidence les raisons théoriques. Ces analyses montrent la difficulté des problèmes d'ordonnancement et des défis des travaux futurs. Nous dressons trois perspectives pour réaliser une utilisation de ressource plus élevée:

1) Ordonnancement avec solutions sous-optimales,
2) Modèles de tâches spécialisés,
3) Relaxation des contraintes en temps réel

Notre travail dans le chapitre 2 ainsi que les travaux dans [Quan00] ety [George00] ont suivi la première perspective qui a comme désavantage un temps de calcul hors-ligne ou en-ligne très important, et donc qui perd de son intérêt en pratique.

DWCS [West04] s'inscrit dans la deuxième perspective, et peut être considéré comme un mécanisme intéressant en termes d'utilisation de ressource, mais ses demandes extrêmes sur le modèle de tâche lui font perdre en généralité et perdre possibilité d'utilisation comme ordonnancement de paquet dans les réseaux.

Selon la troisième direction de recherche, dans le prochain chapitre, nous essayerons de relaxer les pires cas d'inter-arrivée des instances et de proposerons une nouvelle contrainte temps réel. Cette nouvelle contrainte temps réel constituera la contribution principale de cette thèse, en définissant une échéance sur un groupe d'instances plutôt que fournir la garantie sur l'échéance de chaque instance. En outre, on montrera que cette nouvelle contrainte temps réel conduit à une utilisation de ressource significativement plus élevée que les solutions existantes.

Chapitre 4
Relaxation de la contrainte (m, k)-firm

Dans ce chapitre, nous proposerons une version relaxée de la contrainte (m, k)-firm, avec laquelle le système temps réel peut réaliser une utilisation de ressource plus élevée.

Comme expliqué dans les chapitres précédents, l'avantage pratique de la contrainte (m, k)-firm est de servir plus de tâches avec une ressource limitée. Autrement dit l'intérêt des recherches est d'augmenter le facteur d'utilisation autant que possible. Malheureusement, jusqu'à maintenant il n'existe pas une politique d'ordonnancement non préemptive qui peut obtenir un taux d'utilisation intéressant pour un ensemble de tâches non-concrêt sous la contrainte (m, k)-firm dans le cas général.

En conséquence, nous sommes attachés à proposer une contrainte (m, k)-firm relaxée (« relaxed (m,k)-firm » ou R-(m, k)-firm) selon les perspectives de la recherche listées dans le chapitre précédent. Cette contrainte est bien adaptée aux transmissions de flux multimédia. En fait, le chapitre 3 est la motivation de notre proposition de contrainte R-(m, k)-firm et nous allons l'analyser dans les chapitres 4 et 5.

Dans ce chapitre, nous présentons d'abord la définition générale de la contrainte R-(m, k)-firm, qui peut être définie selon le concept de fenêtre fixe ou de fenêtre glissante. Nous proposerons une condition suffisante pour dimensionner la ressource système pour un ensemble de tâches, avec laquelle la version fenêtre fixe de la contrainte R-(m, k)-firm peut être garantie. Cette condition suffisante sera donnée sous la politique d'ordonnancement à priorités fixes. Les instances de la même tâche sont réparties entre instances obligatoires et instances facultatives selon le pattern Sche_mkfirm qui a été déjà présenté dans la section 4.1.2 du chapitre 1. Cette condi-

tion suffisante peut se mettre en pratique pour traiter la situation de surcharge quand il n'est pas possible de respecter toutes les échéances des instances. De plus, cette condition suffisante est intéressante pour déterminer le minimum de ressource requise pour un ensemble de tâches, tel que le système jette toutes les instances optionnelles et n'utiliser la ressource que pour servir les instances obligatoires. Si la condition suffisante est satisfaite et que les instances sont sélectivement jetées, le système peut garantie une QdS correspondant à la contrainte R-(m, k)-firm. Des simulations sont effectuées pour illustrer l'efficacité de cette proposition en comparaison avec la contrainte HRT et la contrainte (m, k)-firm originale.

Chapitre 5

Garantir la contrainte R-(m, k) -firm - gestion des files d'attente

Lorsque l'on définit l'architecture de réseau, il est nécessaire de trouver une méthode de calcul pour dimensionner la bande passante. Braden et al distinguent deux classes d'algorithmes concernant le contrôle de congestion dans les routeurs : des algorithmes d'ordonnancement et des algorithmes de gestion des files d'attente [Braden98] : « de façon schématique, les algorithmes de gestion de files d'attente g contrôle la longueur des files d'attente en rejetant des paquets si nécessaire. Par contre, les algorithmes d'ordonnancement déterminent quel paquet à envoyer par la suite et sont employés principalement pour contrôler l'attribution de la bande passante parmi les flux ».

Dans l'aspect d'ordonnancement temps réel, l'analyse sur la contrainte R-(m, k)-firm a été effectuée dans le chapitre 4. Dans ce chapitre, nous utiliserons la contrainte R-(m, k)-firm pour contrôler la congestion de réseau avec un algorithme de gestion de file d'attente.

Actuellement, les résolutions de congestion dans les réseaux s'adaptent mal aux transmissions multimédia puisqu'elles rejettent aléatoirement des paquets sans considération du problème de pertes consécutives, et il est considéré comme inutile de retransmettre les paquets avec contrainte temps réel, comme le fait TCP/IP. Nous employons la contrainte R-(m, k)-firm pour traiter le contrôle de congestion avec rejet sélectif des paquets de flux multimédia (r, b)-borné [LeBoutec02] en utilisant des théorèmes du Network Calculus. En outre, nous proposons un mécanisme de gestion actif de file d'attente, appelé « Double Leaks Bucket » (DLB), qui exécute dynami-

quement une transmission sélective des paquets et un mécanisme de rejet selon la situation de charge de réseau. On établit une condition suffisante de la configuration de DLB qui permet de fournir une garantie déterministe de contrainte R-(m, k)-firm.

CHAPITRE 6
Conclusion et travaux futurs

Conclusion générale

Cette thèse a traité des mécanismes, des politiques et des approches pour fournir une dégradation contrôlée (« graceful degradation ») de la qualité de service temps réel dans les systèmes distribués et les réseaux temps réel. Comme les systèmes temps réel adaptatifs et les flux multimédia temps réel ont une certaine tolérances aux pertes et aux retards, nous avons mis en oeuvre la contrainte (m, k)-firm pour évaluer, mesurer la dégradation gracieuse de la qualité de service, et pour rejeter sélectivement une certaine quantité de paquets en cas de surcharge.

Cette politique peut distribuer uniformément les instances rejetées pendant l'exécution de tâche et éviter des longues pertes consécutives. Le but de notre travail est de chercher une méthode efficace d'allocation de ressource qui peut garantir de façon déterministe la contrainte (m, k)-firm pour un ensemble de tâches temps réel exécutée sur une ressource limitée. Ce point de recherche contribue à l'amélioration des mécanismes de contrôle d'admission pour QdS garantie dans les réseaux et l'allocation de capacité de processeur dans les applications de contrôle-commande.

Comme mentionné avant, nous avons présenté les définitions générales de la contrainte (m, k)-firm et son modèle analytique, ainsi que l'état de l'art des techniques d'ordonnancement et leur condition suffisante d'ordonnançabilité sous contrainte (m, k)-firm.

Les contributions de recherches de cette thèse ont été présentées par étapes, et nous avons en déduit des résultats théoriques. Le chapitre 2 s'est concentré sur

l'algorithme dynamique d'ordonnancement DBP, parce qu'il est approprié aux applications temps réel adaptatives ayant des contraintes de type (m, k)-firm. On a proposé une condition suffisante hors ligne pour déterminer l'ordonnançabilité d'un ensemble de tâches et un intervalle temps suffisant pour vérifier l'ordonnançabilité en ligne sous l'algorithme d'ordonnancement NP-DBP-EDF. L'efficacité de cette condition suffisante est analysée avec l'exemple d'un système de contrôle embarqué dans un véhicule; cependant, l'utilisation de ressource peut-être très bas en cas de garantie déterministe. Encore pire, la condition de ressource pour un ensemble de tâches sous la contrainte (m, k)-firm déterministe n'est pas inférieure au cas de la contrainte HRT [LiTII06].

En conséquence, nous avons analysé les raisons du faible taux d'utilisation des ressources dans le domaine de l'ordonnancement temps réel d'un ensemble de tâches, en ne se limitant pas au cas de la contrainte (m, k)-firm. Après analyse, nous avons conclue que l'ordonnançabilité temps réel est principalement affectée par le temps inter-arrivée des instances des tâches pour des ensembles de tâches concrètes [Jeffay91].

Par ailleurs, le problème général de l'ordonnancement d'un ensemble de tâches temps réel sous contrainte HRT et contrainte (m, k)-firm a été prouvé comme NP-difficile au sens fort et l'utilisation de ressource peut être arbitrairement bas [Garey77] [George00] [Jeffay91] [Mok01] [Quan00] [LiDEA03].

En étudiant les divers travaux existants [West04] [Zhang04] [Quan00], nous avons déduit trois perspectives de recherche pour augmenter l'utilisation de ressource dans les systèmes temps réel : (1) des méthodes heuristiques avec des résultats sous- optimaux (2) définir des modèles de tâches spécialisés (3) relaxer la contrainte temps réel. La première méthode perd en intérêt pratique car elle requiert des tempss de calcul importants. La deuxième méthode exige de se restreindre à des modèles de tâches particuliers (par exemple, modèle de DWCS [West04]), qui ne sont pas forcément adaptées aux contraintes pratiques de l'ordonnancement dans les réseaux. La troisième perspective de recherche, qui nous paraît la plus prometteuse, constitue la contribution principale de cette thèse avec un modèle de contrainte appelé contrainte (m, k)-firm relaxé (« relaxed (m,k)-firm » ou « R-(m,k)-firm ») . R-(m, k)-firm a été démontré être une politique effi-

cace de dégradation contrôlée de QdS en termes d'utilisation des ressources. De plus, elle est très flexible et appropriée aux flux audiovisuels multimédia.

Puisque l'ordonnancement et la gestion de file d'attente sont les deux classes d'algorithmes utilisés dans les routeurs qui permettent de fournir une QdS en cas de surcharge de réseau (congestion), nous les avons tous les deux étudiées dans le cadre de la garantie R-(m, k)-firm.

Pour l'aspect ordonnancement temps réel, nous avons proposé une condition suffisante qui peut garantir de façon déterministe la contrainte R-(m, k)-firm pour un ensemble de tâche sous l'ordonnancement à priorité fixe [LiETFA06].

Dans l'aspect de gestion des files d'attente, nous avons proposé un mécanisme de gestion actif des files d'attente qui, en rejetant dynamiquement certains paquets en cas de congestion, peut garantir de façon déterministe la contrainte R-(m, k)-firm pour un flux d'entrée (r, b)-borné. En outre, ce mécanisme peut être mis en application dans Diffserv et ainsi fournir une garantie de transmission temps réel [LiAINA06].

Travaux futurs

Nous avons étudié l'ordonnancement temps réel avec dégradation de QdS pour les applications qui tolèrent certaines pertes. Des travaux futurs peuvent être effectués dans les points suivants

- un des défis est comment spécifier le degré de la tolérance aux pertes (ou le degré de la dégradation de qualité de service) pour une tâche concrète ou un flux multimédia. Idéalement, ce degré devrait être indiqué selon la perception humaine et les caractéristiques du flux temps réel. Un point positif est que le m et k de (m, k)-firm et de R-(m, k)-firm peut être choisie de façon libre tant que $m<k$. Les résultas de cette thèse peut donc fournir des théorèmes fondamentaux pour la garantie de QdS, et ce quel que soit le degré choisi.
- A court terme, nous envisageons d'implémenter le mécanisme DLB dans le routeur logiciel, module« Click », développé par le groupe LCS du MIT. Le routeur de « Click » est flexible, configurable, et nous permettra d'observer et d'analyser la stabilité de DLB pour gérer une QdS spécifiée avec R-(m,k)-firm.

- Nous devrons également étudier comment utiliser R-(m, k)-firm avec d'autres algorithmes d'ordonnancement que ceux à priorités fixes, car les applications ont parfois besoin de priorités dynamiques.

Institut National Polytechnique de Lorraine
Département de la formation doctorale en Informatique Ecole doctorale IAE+M

Guarantee Real-Time Quality of Service according to (m, k)-firm approach

A Thesis
Presented at February 14, 2007
In Partial Fulfilment
of the requirements for the Degree
Doctor of Philosophy

Doctorat de l'Institut National Polytechnique de Lorraine
(in Computer Science)
By

LI Jian

Defence committee:

President	Jean-Yves Marion, Professor at INPL
Reviewers:	Pascale Minet, Chargé de recherche at INRIA Rocquencourt Pascal Lorenz, Professo at l'Université de Haute Alsace
Examinators:	Pascal Richard, Maître de conférence at LISI / ENSMA Man Lin, Associate Prof. at St. Francis Xavier University, Canada Françoise Simonot-Lion, Professor at INPL
Supervisor:	Ye-Qiong Song, Professor at INPL
Co-supervisor:	Nicolas Navet, Chargé de recherche at INRIA Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications - UMR 7503

Acknowledge

I am deeply indebted to my advisor, Professor Ye-Qiong SONG, for providing me with support, advice and the freedom to pursue exciting and challenging work. I would like also thank him for his amiable helps and cares to my study, his help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this thesis. I have had the pleasure of working with him during my PhD.

I would like to acknowledge my co-supervisor, Nicolas NAVET, and the scientific leader of TRIO, Françoise SIMONOT-LION, who gave and confirmed this permission and encouraged me to go ahead with my thesis.

I'd like to extend my gratitude to all of my friends and colleges at TRIO project in LORIA, and I am grateful for all they have done. Finally, I would like to thank my committee members.

Content

List of Tables

List of Figures

OUTLINE

Real-time technology is becoming increasing pervasive, and more and more of the world's infrastructures depend on it. Typical fields of applications of real-time computing and real-time communication include process control, manufacturing, avionics, air traffic control, multimedia, telecommunications (the information super-highway), telemedicine and intensive-care monitoring, defense, etc.

In real-time control systems, tasks are usually periodic and they have deadline constraints, by which each instance of a task should complete its computation. In the adverse circumstances caused by component failure, techniques to recover from processor failures often involve a reconfiguration; in this case, all tasks are assigned to fault-free processors. This reconfiguration may result in processor overload such that it is no longer possible to meet the deadlines of all tasks. Moreover, although the current network bandwidth can be obtained at lower costs, the emergence of the Internet and new time-constrained applications, such as multimedia audio/video transmission, result in the same problem of limited resources as before.

In general, systems operate for long period of times in fault-prone nondeterministic environments, as far as possible, they should be able to tolerate fault and continue operating properly. **Graceful degradation** is a way to provide a reduced level of service rather than failing completely in case of system overload or unexpected faults. For instance, multimedia streams usually have variable transmission rates and can accept some adequately spaced missed deadlines or packet losses because of the redundancy in the coding scheme and the humane perception tolerance. So far, how to exactly quantify the resulting QoS of multimedia is still an open issue.

There are different real-time requirements according to the fault-tolerance level of the application. Formally, real-time constraint can be classified into hard real-time, soft real-time, and firm real-time. A hard real-time system requires finishing all instances before their deadlines. This rigorous requirement, on one hand, is not necessary for all systems since some deadline misses are negligible for the application. On the other hand, the occurrence of faults (e.g. missed deadlines, lost packets, etc) cannot always be avoided for adaptive real-time systems because, inherently, the systems and its environments are not fully predictable in advance.

On the contrary, *Soft Real-Time* (SRT) systems can accept some misses of deadlines *occasionally,* which is best expressed by probabilistic or statistics guarantees. However, graceful degradation requires not only the reliability but also the availability. For example, some faults that occur in a short interval may lead to a statistically acceptable degradation for some applications, while the high density of faults can be detrimental for some other applications.

Firm Real-Time constraint (FRT) was proposed to avoid the case where there are a large number of consecutive faults. In particular, (m,k)-firm constraint requires that at least *m* instances should be finished before their deadlines among any *k* consecutive ones.

Closely related to firm real-time constraints is *Weakly Hard Real-Time* (WHRT) constraint that puts restrictions on the number of deadlines that can be missed (or must be met) in a certain number of consecutive deadlines. And somewhere, (m,k)-firm real-time has been suggested to be a subclass of WHRT [Bernat01]. Although they both restrict missed deadlines to a precise bound, an inherent difference exists between the two types of real-time constraints. In fact, firm real-time assumes that it is worthless to run the instance if it cannot be fully finished before its deadline. While, under WHRT, an instance is still executed when it exceeds its deadline and can cause the suspension of itself. In our point of view, this makes WHRT be a subclass of SRT. Conversely, FRT can be considered as an active fault scheduling scheme, which drops the instance when it is not possible to finish it before its deadline. This active dropping can reduce the workload in the future scheduling process, and make it easier, in comparison with WHRT, to schedule the following instances. In addition, FRT can avoid the waste of resources by instances that do not meet their constraints.

The aim of adaptive real-time systems is to both provide a priori acceptable system-level performance guarantees and provide graceful degradation in the presence of faults. This requires some kind of determinism/predictability, which implies that, given certain assumptions about workload and failures, the required resources should be able to be dimensioned at the design time.

Since it is our belief that (m,k)-firm constraint provides a convenient and powerful framework to specify the level of fault-tolerance, we choose to focus this thesis on the use of (m,k)-firm constraints in adaptive real-time systems. The challenge of the research is to develop efficient resource management techniques using (m,k)-firm constraints and to evaluate their performances in the context of adaptive real-time control systems and multimedia transmission.

Traditionally, real-time QoS guarantees are achieved by reserving in advance the resources according to the worst-case execution pattern, called over-provisioning, and it induces a low utilization of the resources. Obviously, when feasible, it is better to reserve the resource according to the average transmission rate, and drop some packets in overload condition, rather than reserving much more resources to guarantee all packets in the worst-case. In other words, the key problem is under (m,k)-firm constraint how to determine the least resource requirement for the deterministic guarantees, or how to serve the most number of real-time applications with the available resource capacity.

This line of research is a key point in the design of Next Generation Networks (NGN), in particular, resource and admission control functions (RACF) which will enable the operators to guarantee end-to-end quality for multimedia services, such as VoIP and IPTV, etc. Our research may provide an operator with the ability to define rules for specific types of communication, in order to better allocate network resources.

Intuitively, the resource requirements according to (m,k)-firm constraint should be less than that without degradation (HRT). If (m,k)-firm constraint does not outperform HRT in that regard, it will lose its original motivation and interest. Therefore, Resource requirement will be the most important criteria and measure of the scheduling efficiency,

In the first part of this thesis, we review the state of the art concerning (m,k)-firm scheduling algorithms (fixed pattern and dynamic pattern) and a closely related constraint, called DWCS (Dynamic Window Constraint Scheduling). We then analyse and

compare these scheduling approaches and remind the existing schedulability conditions. Afterwards, we present the three main technical contributions of the thesis.

The first contribution is a sufficient schedulability condition for the classical NP-DBP-EDF (Non-Preemptive - Distance Based Priority - Earliest Deadline First [Hamanathan99]) algorithm. This work is worthy since there was no sufficient provisioning condition for dynamic (m,k)-firm scheduling algorithm before.

The second contribution is a novel real-time constraint that leads to a better resource utilization than (m,k)-firm.

- First, in the light of the existing literature [George02] [Mok01] [Quan00] [Jeffay91], we explain the theoretical reasons of low resource utilization in worst-case schedulability analysis [Li03] [LiETFA06].
- Second, we highlight the lines of research aimed at increasing the resource utilization rate. In fact, as we explain, all past scheduling schemes follow one or the others of these research fields.
- Third, we relax (m,k)-firm constraint to propose a new real-time scheme to challenge the traditional deadline constraint on individual instances. This novel real-time scheme is named the relaxed (m,k)-firm constraint, it replaces the traditional per packet (instance) deadline by a "delay factor" of a group of packets (instances). We explain that this constraint is more flexible and well suited to multimedia streams, such as MPEG, MP3, etc. At last, a sufficient resource allocation condition is given for a task set under fixed priority non-preemptive scheduling. Simulations demonstrate the advantage in terms of resource utilization.

The third contribution is a novel queue management mechanism, called double leaks bucket (DLB), which can deterministically guarantee R-(m,k)-firm constraint. DLB can be implemented in Diffserv and other QoS mechanisms since it can be simply implemented by replacing the widely-used leaky bucket to service the real-time multimedia transmissions with R-(m,k)-firm constraint. To analyze this new mechanism, we make use of queuing theory and network calculus.

The applicability of our novel real-time constraint and mechanisms is demonstrated on packet transmission for multimedia streams and on an in-vehicle embedded control system.

Chapter 1

General Introduction

In this chapter, we will first introduce the general context of current real-time schemes including the real-time task model and the analysis system model. The definitions of periodic and sporadic tasks are introduced in detail since their regular parameters usually lead to many deterministic results. Afterwards, we introduce the state of the art about (m,k)-firm constraint and recent scheduling approaches to guarantee (m,k)-firm constraint. Window-constraint, a similar constraint to (m,k)-firm, will be also introduced as well as its scheduling approaches dynamic window-constraint scheduling. In addition, a deeper analysis and comparison are carried out on the various scheduling algorithms.

1 System model

In a real-time embedded system, the concurrent sources access in a limited resource system. This resource could be the processor capacity for task execution demand or the network bandwidth for the message transmission. Therefore, the problem is how to schedule these access demands while still guaranteeing the timing performance. The optimization of resource utilization rate is also important since it is a criterion for efficiency of a scheduling method. In what follows, some real-time scheduling models and task models should be first introduced.

1.1 MIQSS model

Multiple input queues single server (MIQSS) model can be used to study a large category of computer and telecommunication systems, such as WordFIP and FDDI appli-

cations, CAN for the embedded system in automobile, distributed system of PLC (power line communication) network and integral system of Ethernet switch.

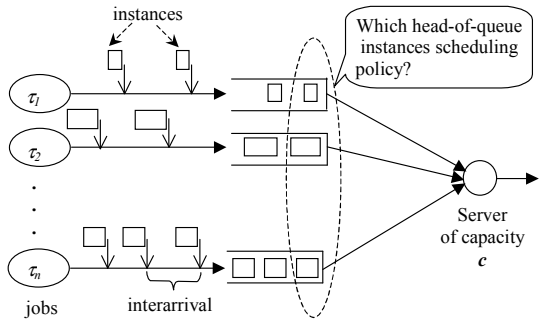


Fig. 1 MIQSS model

The proposed model is made up of N sources generating N streams of tasks τ_i (i = *1, 2, ...n*) attempting to be served by a single server. Each stream is formed by a source and a waiting queue, where a task issued from a source waits until chosen by the processor. The processor chooses tasks at the head of queues according to its scheduling policy.

1.2 WFQ and CBQ

Weighted Fair Queuing (WFQ) is a packet scheduling technique allowing guaranteed bandwidth services. The purpose of WFQ is to enable several sessions to share the same link. WFQ is an approximation of Generalized Processor Sharing (GPS) which, as the name suggests, is a generalization of Processor Sharing (PS) [Kleinrock6]. In PS each session has a separate FIFO queue. At any given time the N active sessions (the ones with non-empty queues) are serviced simultaneously, each at a rate of *1/N*:th of the link speed. Contrary to PS, GPS allows different sessions to have different service shares.

GPS have several nice properties. Since each session has its own queue, ill-behaved session (who are sending a lot of data) will only punish itself and not other sessions. Further, GPS allows sessions to have different guaranteed bandwidths allocated to them. Parekh [Parekh92] showed that when using a network with GPS switches and a session is constrained by leaky bucket model, with regard to the specification an end-to-end delay bound can be guaranteed.

1.3 Source model

The real-time sources are usually characterized as tasks, such as:

- **Periodic and sporadic**: Periodic tasks are real-time tasks, which are activated (released) regularly at fixed rates (periods). Normally, periodic tasks have constraints which indicate that instances of them must execute once per-period. Sporadic tasks are real-time tasks which are activated irregularly with some known bounded rate. The bounded rate is characterized by a minimum inter-arrival period, that is, a minimum interval of time between two successive activations. The time constraint is usually a deadline
- **(r, b)-bounded**: Generally, an affine arrival curve (*r,b*)-bounded [Leboutec02] [Chang00] allows a source to send *b* bits at once, but no more than *r* bit/s over the long run. Parameters *b* and *r* are called the burst tolerance (in units of data) and the arrival rate (in units of data per time unit), respectively.
- **Stochastic**: a stochastic process being called a random field, where the instances arrive randomly. A Poisson distribution is an example of arrival instances.

2 Parameters of periodic and sporadic tasks

In this section, we will introduce the periodic and sporadic task parameters definitions in detail, as well as the definitions of concrete task set and abstract task set.

2.1 Traditional definition

Since a lot of deterministic and successful results have been obtained for a task set anyone of which has the deadline equalling to the period size, in what follows, the parameter of deadline will be ignored since it equals to period. Therefore, a task is τ_i formally modelled by a pair (c_i, p_i) where:

- c_i is the computational cost: the maximum amount of processor time required to execute (the sequential program of) task τ_i on a dedicated unit-processor.
- p_i is the period: the minimal interval between instances of task τ_i.

Throughout this paper, we assume time is discrete and clock ticks are indexed by the natural numbers. Task instances occur and task executions begin at the clock ticks; each of the parameters c_i and p_i is expressed as a multiple of (the interval between) clock ticks. If a task τ_i with cost c_i begins execution at time t and is executed without interruption on a uniprocessor, then the execution is completed at time $t+c_i$. We consider two

paradigms of task instance: periodic and sporadic. If τ_i is ***periodic***, the period p_i specifies a *constant* interval between instances. If τ_i is ***sporadic***, p_i specifies a *minimum* interval between instances. The definition of the behaviour of a task depends on whether it is periodic or sporadic. The ***behaviour of a periodic task*** $\tau_i = (c_i, p_i)$ is given by the following rules for the instance and execution of τ_i. If $t_{i,k}$ is the time of the k_{th} instance of task τ_i, then:

- The $(k+1)_{th}$ instance of task τ_i will occur at time $t_{i,k+1} = t_{i,k}+p_i$..
- The k_{th} execution of task τ_i must begin no earlier than $t_{i,k}$ and be completed no later than the deadlines of $t_{i,k}+p_i$. This requires that c_i units of processor time be allocated to executions of τ_i in the interval $[t_{i,k}, t_{i,k}+p_i]$

The behaviour of a sporadic task is slightly less constrained than that of a periodic task. The ***behaviour of a sporadic task*** $\tau_i = (c_i, p_i)$ is given by the following rules for the instance and execution of τ_i. If $t_{i,k}$ is the time of the k_{th} instance of task τ_i, then:

- The $(k+1)_{\text{th}}$ instance of τ_i will occur no earlier than time $t_{i,k}+p_i$; thus $t_{i,k+1} \geq t_{i,k}+p_i$.
- The k_{th} execution of task τ_i must begin no earlier than $t_{i,k}$ and be completed no later than the *deadline* of $t_{i,k}+p_i$.

Thus the behaviours of periodic and sporadic tasks differ only in the first rule. We assume instances of sporadic tasks are independent in the sense that the time a sporadic task is invoked depends only upon the time of its last instance and not upon the instance times of any other task. Note that the worst-case behaviour of a sporadic task $\tau_i=(c_i, p_i)$ ("worst" in the sense of requiring the most processor time), occurs when τ_i behaves like a periodic task, that is, τ_i is invoked every p_i time steps.

2.2 Concrete and Abstract Task set

The difficulty of scheduling periodic tasks can be affected by the times that tasks are first invoked [Jeffay91]. In addition, we are interested in the predictability of schedulability which can service the network resource dimensioning or evaluate the QoS before the transmission. Therefore, the study concerns the task set with any release time, and concrete and abstract task set definitions should be introduced [Jeffay91].

A ***concrete task*** is a pair (τ_i, r_i), where τ_i is a task, and r_i is a non-negative integer that is the time of the first instance, or the *release time*, of τ_i. The ***behaviour*** of (τ_i, r_i) is the behavior of τ_i constrained further by the rule that the first instance of τ_i occurs at time r_i. Once released, tasks are invoked repeatedly forever.

A ***set of periodic*** (***sporadic***) ***tasks*** $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ is a set of tasks indexed from *1* to *n*, where for each *i*, $1 \leq i \leq n$, $\tau_i = (c_i, p_i)$. A ***concrete set of periodic*** (***sporadic***) ***tasks*** $\omega = \{(\tau_1, r_1), (\tau_2, r_2) ..., (\tau_n, r_n)\}$ is a set of concrete tasks indexed from *1* to *n*, where r_i is the release time of task τ_i. There is a natural many-to-one relation between concrete tasks and (non-concrete) tasks. We say the task τ_i ***generates*** a concrete task (τ_i, r_i) and a concrete task (τ_i, r_i) ***is generated from*** the task τ_i. This relation extends naturally to a relation between concrete task sets and abstract (non concrete) task sets. Let $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ be a task set and let $\omega = \{(\tau_1, r_1), (\tau_2, r_2) ..., (\tau_n, r_n)\}$ be a concrete task set, then the task set Γ ***generates*** the concrete task set ω and ω ***is generated from*** Γ.

Note that a task set is schedulable if and only if the tasks can be scheduled for *any* set of release times. In contrast, each member of a concrete task set has a specified release time, and showing that a concrete task set is schedulable only establishes that its specified release times can be accommodated. Earliest Deadline (EDF) First scheduling algorithm is considered as a successful algorithm since it is optimal for either preemptive or non-preemptive task set. A schedulability of a non-preemptive *general task* set can be decided by the following theorem given in [Jeffay91].

Theorem 1: [Jeffay91] Let $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ be a set of *sporadic or periodic* takss sorted in non-decreasing order by periods (*i.e.*, for any pair of tasks τ_i and τ_j, if $i > j$, then $p_i \leq p_j$). If Γ is schedulable then

(1) $\sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1$

(2) $\forall i, 1 < i \leq n; \forall L, p_1 < L < p_i$:

$$c_i + \sum_{j=1}^{i-1} \left\lfloor \frac{L-1}{p_j} \right\rfloor c_j \le L$$

And if Γ satisfies conditions (1) and (2) then the non-preemptive EDF scheduling algorithm will schedule any concrete set of periodic or sporadic tasks generated from Γ.

In the article [George00], a necessary and sufficient condition is introduced for a *concrete periodic task* set for non-idling, non-preemptive scheduling policy, which is given in the form of the following theorem:

Theorem 2 [George00]**:** Let Γ be a concrete periodic task set (defined as $\tau_i = (r_i, c_i, d_i, p_i)$ for $i=1...n$ with $0 < c_i \le d_i \le p_i$) Γ is feasible on one processor if and only if

(i) $$\sum_{i=1}^{n} \frac{c_i}{p_i} \le 1$$

(ii) a schedule exists where all deadlines in the interval [0, r+2P] are met for all the tasks in the periodic task set.

where P = least common multiple of $\{p_1, ..., p_n\}$ the periods of a task set Γ,

r = max $\{r_1, ..., r_n\}$

In what follows, we will introduce the fault-tolerant real-time system under (m,k)-firm constraint. The research interest is to guarantee the level of fault-tolerance in condition of overload, and the performance evaluation or resource allocation strategies will be focused on.

3 (m,k)-firm definition

(m,k)-firm real-time is one of the suitable ways to design the adaptive real-time system which provides dynamic QoS (Quality of Service) management [ARTIST03]. Formally, a system offering (m,k)-firm constraint requires a minimum QoS of *m* out of

any k consecutive deadlines to meet in the worst-case. In general cases, more than m deadlines are met as the system does not always run at the worst-case condition.

So far, we characterize a task τ_i by $\tau_i = \{p_i, d_i, c_i, m_i, k_i\}$, with $i = 1, 2, \ldots, n$ representing the index of sources. A task could be a stream of message to transmit and a job to do in an embedded system. p_i: the instance generating period, d_i the associated deadline (it can be omitted if the deadline equals to the period), c_i the task service time on the server and m_i and k_i denote the (m,k)-firm constraint.

A task under (m,k)-firm constraint can be found in one of the two following states: *normal* and *dynamic failure* [Hamdaoui95]. Fig. 2 shows the state transition diagram for (2,3)-firm: 1 denotes that a deadline is met and 0 denotes a deadlines missed, respectively. These states are evaluated according to past situation of the system; every state that is either normal or dynamic failure depends on the last three deadlines' services. And the next deadline's meet or miss will cause the system to transit to another state.

If there are more than 1 missed deadline, it is to say that the system is in dynamic failure state. Otherwise, the system is in normal state. This diagram describes the state-transition about a task, however, how to schedule among the tasks will be discussed in the next section.

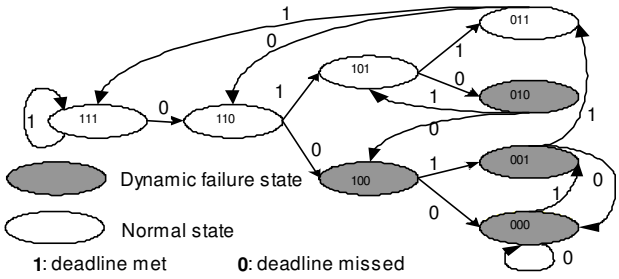


Fig. 2: State-transition diagram with (2,3)-firm

Afterwards, we restrict ourselves to the case of *non-preemptive* scheduling on a uniprocessor; that is to say, we assume a scheduling algorithm that does not interrupt the execution of any task once it has begun. We also restrict the scheduling on a uniprocessor *without inserting idle time*; which means that the scheduling algorithm does not permit the processor to be idle if there is a task that has been invoked but has not completed exe-

cution. To save space and avoid tedium, we will not mention these restrictions in the remainder of the paper.

4 (m,k)-firm scheduling

In this section, we will introduce the recent work that concerns scheduling of a task set under (m,k)-firm constraint. Together with the real-time constraints, the sufficient conditions on various scheduling will be introduced. In addition, the discussions amongst the scheduling algorithms will be carried out.

For the priority assignment among the tasks, the algorithms to guarantee (*m,k*)-firm constraint can be classed into *static scheduling* or *dynamic scheduling*, according to the schedule strategy decided on-line or off-line, respectively. In addition, the algorithm to selective drop packet can divided into *fixed pattern* and *dynamic pattern*, according to the selective dropping strategy among the instances of a task. Either fixed or dynamic pattern is suitable for some specialized applications. E.g., for the MPEG video stream, the packets have different levels of importance, that is to say some packets can be dropped and some cannot. In this case, fixed scheduling should be exploited. However, in some applications, e.g., for MP3 stream, all packets can be considered to have the same importance, so the dropping can be done *dynamically* only according to the efficiency of resource utilization.

4.1 (m,k) fixed pattern scheduling

Fixed pattern is a scheduling algorithm which classes the tasks' instances into mandatories and optionals, such as skip-over, sche_mkfirm, deeply red pattern, EFP pattern, etc. Mandatory ones must be executed and completed before their deadlines, while the optional ones can be discarded without execution when they cannot be completed before their deadlines. Somewhere, fixed pattern under (*m,k*)-firm constraint is also called *(m,k)-pattern*. (*m,k*)-pattern of task τ_i, denoted by Π_i, is a binary string $\Pi_i = \{\pi_{i1}, \pi_{i2}, \ldots \pi_{ik_i}\}$, which satisfies: $\tau_{i,j}$ is a mandatory instance if π_{ij}=1 and optional if π_{ij}=0, and $\sum_{j=1}^{k_i} \pi_{ij} = m_i$.

4.1.1 Skip-Over:

In [Koren95], the authors consider the '*skip-over*' model. It is a special case of the fixed-pattern (m,k) scheduling problem where $m_i = k_i$-*1 (i = 1…n)*. Clearly, it is a special case of (m,k)-pattern defined for task τ_i (6):

$$\pi_{ij} = \begin{cases} 1 & 1 \le j \le k_i - 1 \\ 0 & j = k_i \end{cases} \qquad (6)$$

In [Koren95], the mandatory and optional instances are denoted as red task and blue task, respectively. "*Red tasks only*" and "*blue when possible*" are a static scheduling and a dynamic scheduling respectively for a skippable task set. They differ from the intention to execute or not the optional instance. Red task only algorithm never schedule an optional instance, while bleu when possible does when it will not prevent the mandatory ones to meet their deadlines.

It is proven that to decide whether a set of periodic, occasionally skippable tasks is schedulable is NP-hard in the weak-sense. A sufficient condition is presented in [Koren95] to determine the schedulability for a red task only task set under rate monotonic priority assignment scheme, which is as follows:

Theorem 3 [Koren95]: Given a periodic task set $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$, and where for each i, $1 \le i \le n$, $\tau_i = (c_i, p_i, k_i)$ which allows skips, then, $L \ge \sum_{i=1}^{n} D(i,[0,L])$ for all $L \ge 0$, where $D(i,[0,L]) = \left(\left\lfloor \frac{L}{p_i} \right\rfloor - \left\lfloor \frac{L}{p_i k_i} \right\rfloor \right) c_i$ is a sufficient condition for the feasibility of Γ.

Note that this algorithm can not be readily applied to the (m,k) model, because it is only a special case of (*m,k*)-firm, such as (*k-1,k*)-firm.

4.1.2 Sche_mkfirm pattern.

In [Ramanathan99], one of (m,k)-pattern is proposed, called as *Sche_mkfirm algorithm*, which defines the mandatory instances as follows, and schedules the mandatory instances with Rate Monotonic:

$$\pi_{ij} = \begin{cases} 1 & if \quad j = \left\lfloor \left\lceil \frac{j \times m_i}{k_i} \right\rceil \times \frac{k_i}{m_i} \right\rfloor \\ 0 & otherwise \end{cases} \qquad j = 1,2,\mathrm{L}\ k_i \tag{7}$$

For this (m,k)-pattern of a task, its (*m, k*)-pattern is fixed once its (*m, k*)-firm constraint is defined. A sufficient condition is given that the schedulability is determined by following:

Theorem 4 *[Ramanathan99]:* Given $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$, where $\tau_i = (c_i, p_i, m_i, k_i)$, such that $p_1 < p_2 < ... < p_n$; let

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

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

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

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

For any time interval *t*, if $min_{t \in R_i} W_i(t)/t \le 1$ for all $1 \le i \le n$, then Algorithm Sched_mkfirm meets the (m_i,k_i)-firm guarantee requirement of each task τ_i.

As by definition, *Sche_mkfirm algorithm* associates always the first instance of any task as a mandatory, then the worst-case response time of every task are always obtained by the first instances of each. This is called as *worst-case interference point (WCIP)* of task τ_i, which defines a time interval in which the number of mandatory instances of τ_i is the largest among all time intervals with the same length. This is just the strategy of the above theorem to guarantee every first instances of each task so as to give the sufficient condition for (m,k)-firm constraint.

4.1.3 Deeply-red (m,k)-pattern

In [Quan00], an extreme (m,k)-pattern called *deeply-red (m,k)-pattern* was given, which is defined as:

$$\pi_{ij} = \begin{cases} 1 & 1 \le j \le m_i \\ 0 & m_i < j \le k_i \end{cases} \qquad (8)$$

The Theorem 5 in [Quan00] which proved this deeply-red (m,k)-pattern is the critical situation is now given:

Theorem 5 [Quan00]: for task set Γ with $r_i=0$, if the mandatory instances defined by the deeply-red (m,k)-pattern are schedulable, the mandatory instances derived from any other (m,k)-patterns are also schedulable.

Observe that Deeply-red (m,k)-pattern centralised the WCIP to the release time and can be considered as the worst-case of the (m,k)-pattern. If Deeply-red (m,k)-pattern is schedulable, all other patterns such as Sche_mkfirm pattern and Skip-over will be schedulable as well. However, this Deeply-red (m,k)-pattern has a low resource utilization, and even worse, in many cases it requires the same resource to serve a (m,k)-firm constrained task set than a HRT constrained task set. This causes the loss of efficiency for Deeply-red (m,k)-pattern, which will be discussed later on.

4.1.4 Enhanced Fixed-Priority (m,k)-pattern

In [Quan00], by rotating right *Sche_mkfirm pattern*, a new (m,k)-pattern called Enhanced Fixed-Priority (m,k)-pattern was given as follows:

$$\pi_{ij} = \begin{cases} 1 & if \quad j = \left\lfloor \left\lceil \dfrac{(j+s_i)\times m_i}{k_i} \right\rceil \times \dfrac{k_i}{m_i} \right\rfloor - s_i \\ 0 & otherwise \end{cases} \qquad (9)$$
$$j = 1, 2, \mathrm{L}\ k_i$$

In fact, EFP (m,k)-pattern produced offline from rotating right *Sche_mkfirm pattern*, it does some efforts to avoid the WCIPs of each task to overlap at the beginning time. The value of s_i is configured to separate the WCIPs of the considered tasks at most. Intuitively, this EFP (m,k)-pattern can never be worse than *Sche_mkfirm pattern*, and it

has been shown experimentally that EFP algorithm behaves significantly better than Sche_mkfirm algorithm in terms of system schedulability. However, the configuration takes a complicated calculation and tries all possibilities, which makes EFP (m,k)-pattern not suitable for adaptive real-time applications or network transmission because of the unstable environment.

4.2 (m,k) Dynamic scheduling pattern

Notice that the *blue when possible* algorithm for skip-over constraint task can be considered as a dynamic scheduling for skippable tasks, while Distance based priority (DBP) is a general dynamic (m,k)-firm scheduling algorithm.

4.2.1 DBP (Distance Based Priority):

DBP was firstly introduced by Hamdaoui and Ramanathan [Hamdaoui95], as a dynamic priority assignment mechanism for tasks under (m,k)-firm constraint in a MIQSS model.

The basic idea of DBP algorithm is quite simple and straightforward: the closer the stream to a failure state the higher its priority is. A failure state occurs when the stream's (m,k)-firm requirement is violated, i.e., there is more than *k-m* deadline misses within the last *k*-length window.

To know the current state of a task we should examine the execution historic of the last *k* instances. If we associate '*1*' to an instance with deadline met and '*0*' to an instance with deadline missed, similar to (m,k)-pattern this historic is then entirely described by a word of *k* bits called k-sequence.

The *k-sequence* is a word of *k* ordered bits in which each bit keeps memory of whether the deadline is missed (*bit= 0*) or met (*bit=1*). In k-sequence, the bit is ordered the most recent to the oldest task instance where the leftmost bit represents the oldest. Each new arrived instance causes a shift of all the bits towards left, the leftmost exits the word and is no longer considered, while the rightmost will be a *1* if the instance has met its deadline (*i.e.* it has been served within) or a *0* otherwise.

Fig. 3 gives an example with (3,5)-firm constraint.

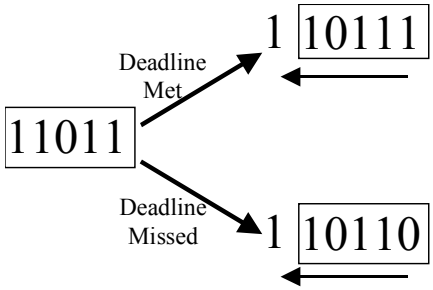


Fig. 3: **Evolution of the *k*-sequence**

Thus for each stream source τ_i, which requires a (m_i,k_i)-firm, the priority is assigned based on the number of consecutive deadline misses that leads the stream to violate its (m_i,k_i)-firm requirement. This number of missed deadlines is referred to as *distance to failure state* from current state. DBP assigns priority to a given instance by the *distance* from the current *k*-sequence to a failure state. Formally, according to [Hamdaoui95] priority is evaluated as follows:

Let $s_j=\left(\delta^j_{i-k_j+1},\Lambda,\delta^j_{i-1},\delta^j_i\right)$ denote the state of the previous *k* consecutive instances of τ_i, $l_j(n, s_j)$ denote the position (from the right) of the n^{th} meet (or *1*) in the s_j, then the priority of the $(i+1)^{th}$ instance of τ_j is given by :

$$P\_DBP_{j,i+1} = k_j - l_j\left(m_j, s_j\right) + 1 \qquad \textbf{(10)}$$

If a stream is already in failure state (i.e., less than *m 1s* in the k-sequence), the highest priority '0' is assigned. For example, considering a stream with (3,5)-firm constraint, the current instance $\tau_{j,i+1}$ is assigned the priority of *2* if current 5-sequence is (11011), and is set the priority of *3* if current *5*-sequence is (10111).

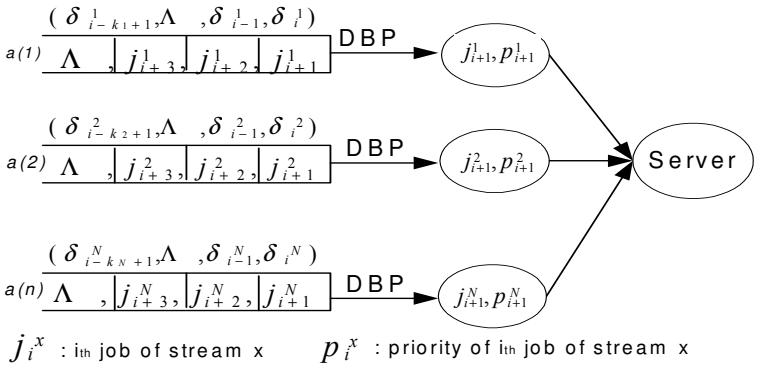


Fig. 4: DBP for priority assignment of head-of-queue's tasks

One of the problems about DBP is that it assigns priorities only considering one τ_i's (m_i,k_i)-firm constraint without comparing it to the others sharing the same server. This *self-reference* behaviour may lead to a situation where more than one stream gets the same priority at the same time; in this case, another algorithm to choose among them should be defined. In the later part, when head-of-queue's instances of different tasks have the same priority, EDF (Earliest Deadline First) is used by default.

Note that the introduced (m,k)-firm algorithms depend solely on the parameters of m over k of each task, regardless of task periods and execution times. Furthermore, only the subtraction of k and m is taken into account, so this results in the unilateralism. For example, (2,3)-firm and (3,4)-firm constraint will be assigned with the same priority for the first instance. In the next section, Dynamic Windows Constraint does some efforts to improve this by using the quotient of ration of loss-tolerance in a fixed window, but not yet taking into account the task periods and execution times.

5 Window Constraint

Similar to (m,k)-firm constraint, another important dynamic selective dropping constraint, so-called Window-Constraint will be introduced [West99]. Window-Constraint is defined by a value $W_i = x_i/y_i$, where the window-numerator, x_i, is the number of packets that can be lost or transmitted late for every fixed *window*, y_i (the window denominator), of consecutive packet arrivals in the same stream, τ_i. Hence, for every y_i packet arrivals in stream τ_i, a minimum of y_i-x_i packets must be scheduled for service by

their deadlines, otherwise a service violation occurs. At any time, all packets in the same stream, τ_i, have the same window-constraint, W_i, while each successive packet in a stream, τ_i, has a deadline that is offset by a fixed amount, p_i, from its predecessor. After servicing a packet from τ_i, the scheduler adjusts the window-constraint of τ_i and all other streams whose head packets have just missed their deadlines. Consequently, a stream τ_i's *original* window-constraint, W_i, will differ from its *current* window-constraint, W_i'. Observe that a stream's window-constraint can also be thought of as a *loss-tolerance*.

5.1 Comparison between (m,k)-firm and window-constraint

Window-constraint explicitly attempts to provide service guarantees over *fixed windows* of deadlines. That is, Window-constraint assumes the original window-constraint specified for each stream and defines service requirements over non-overlapping groups of deadlines. Conversely, (m,k)-firm constraint restricts on *any k consecutive instances*, which means a *sliding window* beginning from any instance of task sequence.

However, it is possible to determine a corresponding *sliding* constraint for a specified fixed window-constraint [West00] [West04], vice versa. As stated earlier, window-constraint requires to guarantee no more than x_i out of a fixed window of y_i deadlines is missed, for each task τ_i. This is the same as guaranteeing a minimum $m_i=y_i - x_i$ out of a fixed window of $k_i = y_i$ deadlines are met. It can be shown that, if no more than x_i deadlines are missed in a *fixed* window of y_i deadlines, then no more than $2x_i$ deadlines are missed in a *sliding* window of $y_i + x_i$ deadlines. Likewise, this means that by meeting m_i deadlines over fixed windows of size k_i, then it is possible to transfer x_i/y_i window constraint as (y_i-x_i, y_i+x_i)-firm constraint, and transfer (m_i,k_i)-firm constraint as $2(k_i-m_i)/(2k_i-m_i)$ window constraint. The proof is omitted for brevity.

Although window-constraint and (m,k)-firm constraint can be transformed to each other, they are essentially different. Briefly, window-constraint is less restricted than (m,k)-firm constraint under the same loss rate. For example, one task under (1,2)-firm constraint and 1/2 window-constraint can tolerate the same 50% loss, but the following scheduling trace can be accepted by 1/2 window-constraint but not by (1,2)-firm constraint (if the sliding window with size tow periods is slid on the 2nd and the 3rd in-

stances). Moreover, this scheduling trace can only be accepted by (1,3)-firm constraint but it is not necessary for some place where the window slide to (e.g. the sliding window with the size of three periods is slid on the 4^{th}, 5^{th} and 6^{th} instances). In fact, this can be easy get by the transformation method mentioned above.

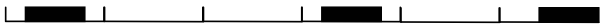


Fig. 5: 1/2 window constraint and (1,3)-firm constraint

Moreover, (m,k)-firm constraint is also proposed for the sliding window which slides period by period. Therefore, the window can not slide at [50, 66] to verify the (m,k)-firm constraint for task τ_1.

- **Performance difference between the fixed window and the sliding window.**

In fact, the sliding window means that the window slides period by period, while fixed window means that the window slides k-periods by k-periods. Moreover, the starting time of both fixed window and sliding window must be the beginning of period. We will use a concrete example to show sliding window performance in what follows.

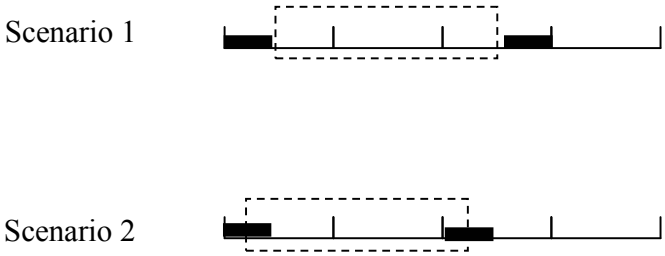


Fig. 6 : constraint on sliding window performance

Assume a task is under (1,2)-firm constraint, which means one deadline (assume that deadline=period) must be met among any two consecutive ones. And it is obvious that the two scenarios in Fig. 6 have satisfied (1,2)-firm constraint since one deadline is met among 2 consecutive ones. However, the window can not slide continuously since the workload arrives period by period for periodic task. This fact can be demonstrated as follows: if the window slides to the place shown in scenario 1, there is no instance execu-

tion. Furthermore, even the instance executions are well placed as in scenario 2, and when the window slides to the place shown in scenario 2, there are two halves of instance execution (it is not reasonable to say one instance is executed, only because of 0.5+0.5=1, which means 0.5 execution of the first instance and 0.5 execution of the third instance in the window). However, according to the (m,k)-firm definition, (1;2)-firm constraint has be satisfied in both scenarios, and this can prove that the sliding window of (m,k)-firm must slide period by period, and the window must start from the beginning of the period instead of anytime.

5.2 Dynamic Window-constraint Scheduling

In this section, we will introduce the scheduling algorithms for window-constraint. R. West proposed successively two versions so called dynamic window constraint scheduling (DWCS). Therefore, in order to distinguish them, we called DWCS-1 and DWCS-2. These two algorithms schedule in MIQSS model and give the precedence to the packets based on the current value of their loss-tolerance and deadlines. DWCS modifies the precedence dynamically since the loss-tolerance of a packet changes over time after a missed or met deadline.

5.2.1 DWCS-1

Table 1 shows the rules for ordering pairs of packets in different streams. Recall that all packets in the same stream are queued with arrival order. The precedence is given to the stream with the lowest loss-tolerance, where $W_i = x_i/y_i$ is the current loss-tolerance for all packets in stream i. If two packets have the same non-zero loss-tolerance, they are ordered earliest-deadline first (EDF). If two packets have the same non-zero loss-tolerance and the same deadline, they are ordered by the lowest loss-numerator x_i first, since fewer consecutive packet losses can be tolerated. If two packets have zero loss-tolerance and their loss-denominators are both zero, they are ordered by EDF, otherwise they are ordered highest loss-denominator first. In all the other cases, *First In First Out* will be the rule to assign the precedence.

Pairwise Packet Ordering (1)
Lowest loss-tolerance first
Same non-zero loss-tolerance, order EDF
Same non-zero loss-tolerance & deadlines, order lowest loss-numerator first
Zero loss-tolerance & denominators, order EDF
Zero loss-tolerance, order highest loss-denominator first
All other cases: first-come-first-serve

Table 1: DWCS-1 Precedence amongst pairs of packets

Every time a packet in stream τ_i is transmitted, then the loss-tolerance of τ_i is adjusted. Likewise, other streams' loss-tolerances are adjusted only if any of the packets in those streams miss their deadlines as a result of queuing delay.

5.2.2 DBP, DWCS, and EDF Comparisons

The comparison among DBP, DWCS and EDF is exploited on the scalability and QoS granularity that should be provided by the server [Striegel03].

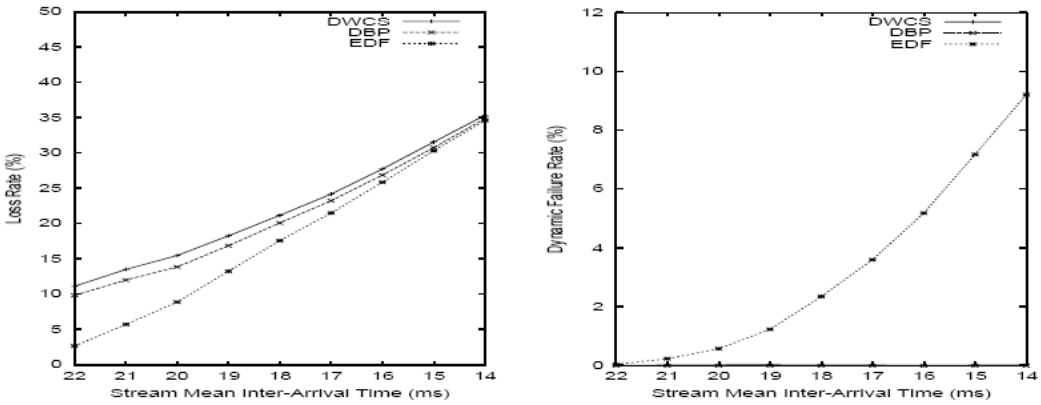


Fig. 7: loss-rate and failure rate comparison among DWCS, DBP and EDF

Fig. 7 shows the experimental packet loss-rate of the EDF, DBP, and DWCS schedulers. Whereas the loss-rate contains all packet losses experienced by the schedulers, dynamic failure rate measures only those packets that have a negative impact (violation of loss window) on the perceived QoS of the clients. From the graphs, it can be seen that the DBP and DWCS are better than EDF in terms of dynamic failure rate; however, they perform worse in terms of loss-rate. See from Fig. 7, loss-rate is not necessarily correlated with the dynamic failure rate.

Notice that DBP and DWCS can reduce the rate of the failure state, but miss more deadlines. This is the motivation why in [West01], a new DWCS scheduling based on hard real-time was proposed, that will be introduced in the next section.

5.2.3 DWCS-2: Hard Real-Time scheduling using DWCS

Unfortunately, in under-load situations, earliest deadline first (EDF) scheduling is often more likely meet deadlines. However, the DWCS-1 algorithm is still better than EDF in overload cases where it is impossible to meet all deadlines.

In [west00] [West04], another dynamic scheduling DWCS (Dynamic Window-Constrained Scheduling) algorithm was proposed, so that hard real-time guarantee can be made on packet streams in tolerating x missed deadlines among every y requests. The following Table 2 describes the rules of revised version of DWCS for ordering pairs of packets.

Pairwise Packet Ordering (2)
Earliest deadline first (EDF)
Equal deadlines, order lowest window-constraint first
Equal deadlines and zero window-constrain, order highest window-denominator first
Equal deadlines and equal non-zero window-constraint, order lowest windows-numerator first
All other cases: first-come-first-serve

Table 2: DWCS-2

Precedence is given to packets in streams according to the rules shown in Table 2. This table of precedence rules differs from the original table of DWCS-1 [West99]. The basic difference is that the top two lines in the table are reversed: the original table *first* compares packets based on their streams' current window-constraints, giving precedence to the packet in the stream with lowest (numeric-valued) window-constraint. If there are ties, the packet with the earliest deadline is chosen.

5.2.4 Loss-Tolerance Adjustment of DWCS

We now describe how loss-tolerances are adjusted. Let x_i/y_i denote the original loss-tolerance for all packets in stream τ_i. Let x'_i/y'_i denote the current loss-tolerance for all queued packets in stream τ_i. Let x'_i denote the current loss-numerator, while x_i is the original loss-numerator for packets in stream i. y'_i and y_i denote current and original loss-denominators, respectively. Before a packet stream is serviced, its current and original loss-tolerances are equal. For all buffered packets in the same stream τ_i as the packet most recently transmitted before its deadline, adjust the loss numerators and denominators as follows:

(A) **Window-constraint adjustment for a packet in τi serviced before its deadline:** in DWCS system, when a missed deadline causes the violation of window-constraint of a task, this task will be tagged and wait for the recover for the violation or reset of the tag.

```
if (y'i > x'i) then y'i = y'i − 1;
else if (y'i = x'i) and (x'i > 0) then
        x'i = x'i − 1; y'i = y'i − 1;
if (x'i = y'i = 0) or (τi is tagged) then
        x'i = xi; y'i = yi;
 if (τi is tagged) then reset tag;
```

Fig. 8: Window-constraint adjustment after a serviced packet

(B) **Window-constraint adjustment when a packet deadline of $\tau_j \mid j \neq i$ is missed:** comprising one or more late packets, is adjusted as shown in Fig. 9. In the absence of a feasibility test, it is possible that window-constraint violations can occur. A violation ac-

tually occurs when $W'_j = x'_j/y'_j \mid x'_j = 0$ and another packet in τ_j then misses its deadline. Before τ_j is serviced, x'_j remains zero, while y'_j is increased by a constant, ε, every time a packet in τ_j misses a deadline. The exception to this rule is when $y_j = 0$ (and, more specifically, $W_j = 0/0$). This special case allows DWCS to *always* service streams in EDF order, if such a service policy is desired.

```
if (x'_j > 0) then
        x'_j = x'_j − 1; y'_j = y'_j − 1;
        if (x'_j = y'_j = 0) then x'_j = x_j; y'_j = y_j;
else if (x'_j = 0) and (y_j > 0) then
        y'_j = y'_j + ε;
Tag τ_i with a violation;
```

Fig. 9: Window-constraint adjustment when deadline missed

5.2.5 Sufficient condition for DWCS-2

For DWCS-2, a sufficient and necessary condition has been given for a stream set, which can deterministically guarantee the schedulability.

Theorem 6 [West04]**:** Consider a set of n streams, $\Gamma = \{\tau_1, \ldots, \tau_n\}$, where $\tau_i \in \Gamma$ is defined by the 3-tuple with the same windows-constraint factors ($C_i = K$; $T_i = qK$; $w_i = x_i/y_i$). If the utilization factor, $U = \sum_{i=1}^{n} \frac{(y_i - x_i)}{qy_i} \leq 1.0$, then $x_i = y_i - 1$ maximizes n. And this stream set is schedulable by dynamic window-constraint scheduling (version 2).

5.2.6 Comments on DWCS-2:

At first glance, the sufficient and necessary condition for DWCS-2 is very exciting because the resource utilization can achieve the 100%, such that DWCS-2 is the optimal scheduling for selective packet dropping in network communication. However, we found that this result is effective only in special cases. Its limitations and shortcomings will be talked out in following points:

- Seen from the theorem, DWCS-2 approach requires that all streams must have the same unit-size execution time, $c_i=K$, and the same period, $p_i=qK$. This restriction is not suitable for network communications, since the real-time applications usually have the different packet sizes.
- See from Table 2, DWCS-2 upgrades EDF as the first scheduling rule such that it invalidates itself as a modification of EDF. This results in the effect that the rules below EDF can only be taken into account only in case of the same deadlines. Normally, in HRT-EDF scheduling, Rate Monotonic (RM) or other Fixed Priority (FP) scheduling method is used to assign the precedence in case of equal deadlines. Likewise, window-constraint (loss-rate) plays the same role as RM and FP, and losses its selective dropping characters. In fact, any violation of window-constraint caused by EDF will occur by DWCS-2 scheduling as well.
- In order to validate the DWCS scheduling, all stream periods must be synchronized. Otherwise, the streams will be scheduled as HRT under EDF scheduling, since EDF is the first one in the HRT-DWCS scheduling rules.
  This fact can be shown by an example shown in Fig. 10 and Fig. 11, where Fig. 10 shows a case where two streams (assume that c=K, p=K) with ½ window-constraint having synchronized release times, then the window-constraint can be satisfied.

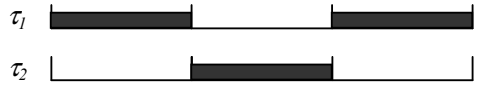


Fig. 10: HRT-DWCS is valid for synchronized streams

Fig. 11: HRT-DWCS is invalided for non-synchronized streams

However, Fig. 11 shows that if stream τ1 is released a little earlier, the same two streams will be scheduled according to EDF, and τ_1 is always serviced,

but τ_2 is never. The reason is that DWCS-2 modifies the original DWCS scheduling by changing the first two rules, which aims to ameliorate the miss deadline rate.

Notice that all streams in the communication systems have jitters and the streams started arbitrarily, so the synchronization cannot be well realized in real system. Hence, the window-constraint is rarely taken into account by DWCS-2 during scheduling, while the streams are scheduled according to HRT-EDF.

The above points showed that DWCS-2 can only provide the high utilization in some extreme special cases, and it can be considered as a hard real-time scheduling algorithm. In fact, in [West99], it is also recognized by the author and DWCS-2 is called as hard real-time scheduling using DWCS. However, the scheduling algorithm should selectively schedule the instances according to the level of tasks' fault tolerance in condition of overload, and this character is lost by updating EDF as the first rule. (Moreover, in the next chapter, we will discuss that the proof of the sufficient condition is not correct at all, and we will reconstruct it in appendix A).

6 Conclusion

In this chapter, we have introduced the basic analysis models and provided the fundamental information. The basic motivation of (m,k)-firm constraint and window-constraint is selectively dropping the instances (packets) in case of overload. Observe that DWCS-2 scheduling algorithm can achieve the 100% resource utilization, which seems optimal, but it is only available under a set of extreme conditions. Knowing that both (m,k)-firm and window-constraint are the graceful degradation constraints for the adaptive real-time system, the extreme conditions required by DWCS-2 are not reasonable and realizable in the actual systems. Conversely, skip-over pattern, deeply-red (m,k)-pattern, Sche_mkfirm pattern and EFP (m,k)-pattern schedule the task set without special requirement unlike DWCS-2, so these (m,k)-patterns gained the generality. Moreover, Sche_mkfirm pattern and EFP (m,k)-pattern are fixed in advance such that they are the most adequately spaced and the cost of realization will not be considerable.

However, the resource utilization of these general scheduling approaches is poor. This fact will be detailed in chapter 3 along with our deeper analysis and conclusion. In the next chapter, we will focus on the dynamic scheduling algorithm in order to gain higher resource utilization. Observe that DBP and original DWCS-1 are proposed without special requirement on the task set, but until now, there are not yet sufficient schedulability conditions on either DBP or DWCS-1.

Chapter 2

NP-DBP-EDF Sufficient Condition

In the previous chapter, we have introduced two general selective instance-dropping scheduling methods, named as DBP and DWCS-1. In this chapter, we will study the sufficient condition to determine the schedulability of a task set under DBP scheduling algorithm for (m,k)-firm constraint. Recall that (m,k)-firm constraint can be transferred to window-constraint, vice versa, so we will only focus on DBP scheduling for guaranteeing (m,k)-firm constraint. Note that (m,k)-firm constraint is employed in case of overload, such that a task set with loss-tolerance under (m,k)-firm should intuitively require less resource capacity than without loss-tolerance under HRT. The efficiency of the sufficient condition will be examined in terms the resource requirement in comparison with HRT.

1 Motivation of NP-DBP-EDF

It is well known that real-time systems designed according to the worst-case condition (case of hard-real time system design) often result in large resource requirement. As at run time the system is seldom in a worst-case condition, large amounts of system resources are under-utilized. One solution is to design the system based on an average case. This solution can be suitable for a subclass of soft real-time systems requiring only statistic deadline guarantee. However, for other real-time systems such as those found in

multimedia and the automatic control domain, providing only statistic deadline guarantee can be unacceptable. A more precise specification on how the deadline misses are distributed in time is necessary [Bernat01] and this can be done using the (m,k)-firm model [Hamdaoui95]. Typically, for the same deadline miss ratio, a real-time application better tolerates non-consecutive deadline misses than consecutive ones. A system is said under (m,k)-firm real-time constraint if it requires the guarantee of the deadline meet of at least *m* out of any *k* consecutive instances of a recurrent task.

Much previous work has dealt with new scheduling algorithms integrating the additional (m,k)-firm constraint [ZWang02]. Two families can be found: *dynamic* and *static*. DBP (Distance Based Priority) [Ramanathan99] and DWCS (Dynamic Window-Constrained Scheduling) [West00] are dynamic. The priority assignment done on-line is based on the current state of the system. ERM (Enhanced Rate Monotonic) [Ramanathan99] and EFP (Enhanced Fixed-Priority) [Quan00] are static as the scheduling is done off-line using a static deadline miss pattern. Finally, note that, as for hard real-time, sufficient conditions of feasibility are obviously required in order to ensure a *deterministic (m,k)-firm guarantee*. There are sufficient conditions for ERM, EFP and DWCS [West00] [West04], but no such condition has been investigated for DBP.

In this chapter, we only consider the dynamic (m,k)-firm scheduling algorithms for the following expects. The system should be able to adapt to workload variation (e.g. in networks handling QoS with connection admission control) by taking advantage of the possibility to *discard* until *k-m* out of *k* consecutive instances during system overload periods. Therefore, in this context, off-line scheduling is simply not suitable. Furthermore, the use of a dynamic scheduling policy could allow a better exploitation of the available resources in general. Finally, we insist on the importance of discarding the instances of tasks whose deadlines cannot be met by the system. In fact, an overload situation leads to deadline misses, and only discarding part of some tasks instances (preferably those with missed deadlines) can provide better performance. Scheduling with dynamic instance

drops makes our work different to the classic scheduling studies without drops (e.g. [Ramanathn95], [Bernat01], [Bernat03]).

Once we have established the focus on a dynamic policy, we have to justify the choice of DBP in our work as opposed to DWCS. We recall that for the targeted applications, we have to exhibit at least a sufficient feasibility condition. For DWCS such a condition was established in [West04]; but it has a limited application region since the tasks must be with the same service time and the same periods. That is why, although DBP itself could be improved [Poggi03], we investigated this scheduling in order to find a more general condition. Moreover, as we would like to obtain a result applicable to both task scheduling and network packet scheduling, we further restrict ourselves to *non-preemptive scheduling*. As proposed in former studies [Hamdaoui95], [West04], we consider EDF for equal priority cases.

Above all, in this chapter we only focus on NP-DBP-EDF (Non-Preemptive - Distance Based Priority – Earliest Deadline First).

2 NP-DBP-EDF scheduling

In [Poggi03], a necessary condition has been proposed for NP-DBP-EDF scheduling. However, this necessary condition only tells us that meeting all (m_i, k_i)-firm constraints are impossible if the server capacity is below a certain threshold. It does not tell us what the sufficient server capacity is for meeting all (m_i, k_i)-firm constraints. Therefore, for providing deterministic guarantee, a sufficient condition is fundamental.

2.1 NP-DBP-EDF scheduling algorithm

Under the NP-DBP-EDF scheduling policy, at time t, the instance, which is being executed in the unique server, has highest priority because of the non pre-emption. The priority is assigned to the instances waiting for execution at the head of the queues which have the smallest DBP value. In case of the same DBP value, EDF is used.

We note by $DBP_j(t)$ the DBP value of task τ_j at time point t. Instances of a same task are stored in a FIFO queue. The instances with $DBP_j(t) = 1$ (*for j = 1, 2, n*) must be

executed before their deadlines, otherwise their (m_j,k_j)-firm constraint guarantee will be violated. Instances with $DBP_j(t) > 1$ will be executed if they can have the server and the operation could be completed before the deadline, otherwise they are discarded. The fact of discarding task instances makes the following schedulability analysis different to the classic ones (e.g. those found in [Bernat01], [Bernat03]).

2.2 Busy period and workload evaluation

We define **DBP=*X* busy period** as the time interval during which the server is occupied by, and only by, the instances of tasks whose DBP value is equal to ***X***.

Obviously, any missed deadline of DBP=1 instance will violate the (m,k)-firm constraint. This is the reason why, afterwards, we will only focus on the worst-case processor demand related to DBP=1 busy period.

According to NP-DBP-EDF scheduling, except for the running instance (non preemption), DBP=1 instances have the highest priority. Then, once there are DBP=1 instances access requirements, they should be executed immediately or simply wait until the end of the executing instance. These two cases will be analyzed in what follows, and we will show how to calculate the total workload of the task set in a time interval started form the occurance of DBP=1 instances. The workload situations will be demonstrated in Fig. 12 and Fig. 14 respectively, and the strategies of the workload calculation will be used in the proof of Theorem 7.

First, we consider the situation that, at one time point, DBP=1 instance appears and can have the server immediately. The workload is calculated in the following DBP=1 busy period, giving the worst-case possibility. Let t_0 be the starting time of this DBP=1 busy period in this situation, t_d the ending time, and let $L = t_d - t_0$ be the length of the DBP=1 busy period.

Tasks are divided into two sets: one is for the tasks whose DBP=1 instances start from the beginning time of DBP=1 busy period, denoted by U. The other set is $\Gamma - U$.

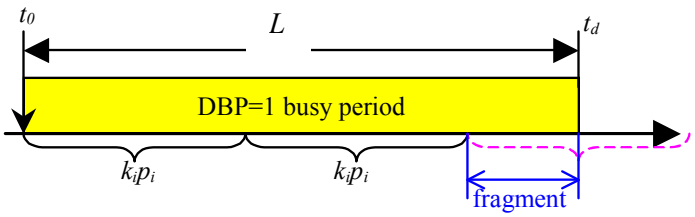


Fig. 12: Workload of DBP=1 instances starting at time point t0.

As shown in Fig. 12, it is given that in every interval k_ip_i for the task $\tau_i \in U$, there are m_i and only m_i instances of task τ_i with DBP=1. Only these instances can be executed and meet their deadlines. This generates a workload of:

$$W_U^1 = \sum_{i \in U} \left(\left\lfloor \frac{L}{k_i p_i} \right\rfloor m_i \right) c_i \qquad (11)$$

But, in general, interval L is not an integer multiple of k_ip_i. So, in the *fragment* (the residue of L divided by k_ip_i), for a task τ_i, the number of possible instances is bounded by m_i. This results in the following term:

$$W_U^2 = \sum_{i \in U} Min\left(\left\lfloor \frac{L - k_i p_i \left\lfloor \frac{L}{k_i p_i} \right\rfloor}{p_i} \right\rfloor, m_i \right) c_i \qquad (12)$$

By using equations (11) and (12), the workload caused by all tasks in U is then:

$$W_U = W_U^1 + W_U^2 \qquad (13)$$

The workload caused by the second part ($\tau_j \in \Gamma - U$) is calculated as follows. Tasks not included in set U have their DBP value greater than 1 at the time point t_0. It is clear that in DBP=1 busy period, only DBP=1 instances can be executed. So, Fig. 13 shows how a task τ_j starts to generate the workload from t_0 in DBP=1 busy period.

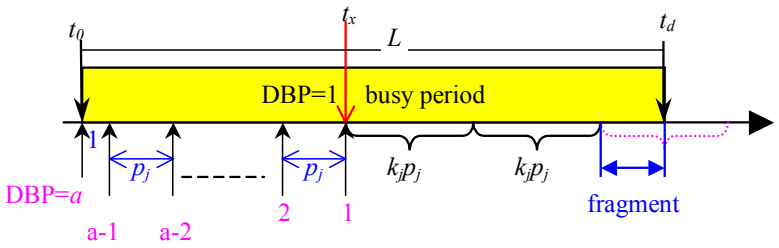


Fig. 13: Workload of instances whose DBP>1 at time point t_0.

Assume that, at t_0, the task τ_j has $DBP=a$ ($a>1$). The worst-case is the following situation: after one clock tick, this DBP value will be decreased by one, and then, after every period p_j, the DBP value will be minus one. No instance is executed in the interval $[t_0,t_x]$, where t_x is the time at which τ_j has its DBP=1. We note $l = t_x - t_0$. The worst case corresponds to:

$$l=1+(DBP_j(t_0)-2)p_j \qquad (14)$$

For the interval after t_x, the workload evaluation is similar to the one used for the set U. According to (11) and (12), we obtain:

$$W^1_{\Gamma-U} = \sum_{j\in\Gamma-U}\left(\left\lfloor\frac{L-l}{k_jp_j}\right\rfloor m_j\right)c_j \qquad (15)$$

$$W^2_{\Gamma-U} = \sum_{j\in\Gamma-U} Min\left(\left\lfloor\frac{(L-l)-k_jp_j\left\lfloor\frac{L-l}{k_jp_j}\right\rfloor}{p_j}\right\rfloor, m_j\right)c_j \qquad (16)$$

Formula (17) expresses the total workload of the tasks in the set ($\Gamma-U$).

$$W_{\Gamma-U} = W^1_{\Gamma-U} + W^2_{\Gamma\text{-}U} \qquad (17)$$

By using equation (13) and (17), the total workload of a DBP=1 busy period is:

$$W = W_U + W_{\Gamma\text{-}U} \qquad (18)$$

Now, we consider the second situation where a task with DBP=1 is blocked by the running instance of a task whose DBP>1(non pre-emption) as shown in Fig. 14. DBP=1 instance generates a workload starting at time point t_y, but a DBP>1 instance of τ_i has occupied the server at t_j and is still being executed. Because of non pre-emption, DBP=1 instances can only be executed after the completion of the DBP>1 instance.

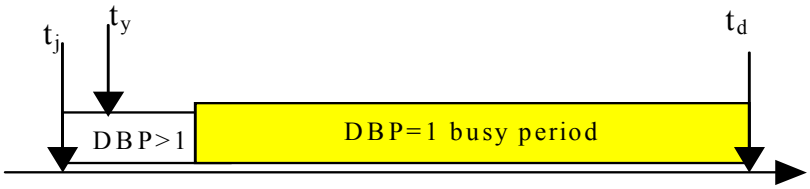


Fig. 14: DBP=1 busy period blocked by DBP>1 instance

Then, we calculate the tight upper bound of the workload in the interval $L = [t_j, t_d]$ (see Fig. 14). Obviously, at the completion of an instance of τ_i, the DBP of the next instance is still greater than 1 and it will not be executed in DBP=1 busy period. After p_i (at time t_r), τ_i also generates a workload only if DBP=1. So, the worst-case is when τ_i has DBP=2 at time t_j. For whatever (m_i,k_i)-firm constraint of τ_i, the worst-case is at time point t_r, the DBP value changes to 1 as in Fig. 15.

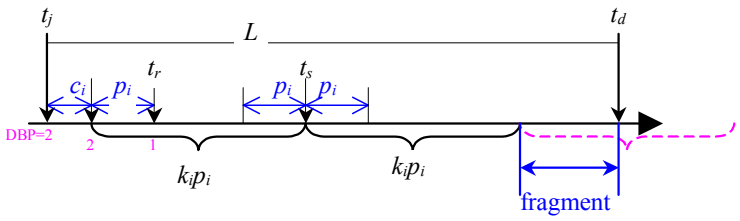


Fig. 15: DBP>1 instance causes workload in DBP=1 busy period

DBP=1 workload of τ_i in $L= [t_j, t_d]$ is calculated as follows. Let $t_s = t_j+c_i+k_ip_i$. Then we can predict that the instance invocated within $[t_s, t_s+p_i]$ has DBP>1. Otherwise, if this instance has DBP=1, there will be m_i-*1* DBP=1 instances in $[t_r, t_s]$ (as in every k_ip_i there are, at most, m_i DBP=1 instances in DBP=1 busy period). Moreover, as the instance within $[t_r-p_i, t_r]$ is a DBP>1 instance and it has not been executed, then in $[t_r-p_i, t_s] = k_ip_i$ there are only m_i-*1* DBP=1 instances. However, this violates the (m_i,k_i)-firm. So, we can

conclude that the instance invocated within $[t_s, t_s+p_i]$ is a DBP>1 instance and will not be executed. We can extend this to the instances invocated in $\left[t_s+k_ip_i, t_s+(k_i+1)p_i\right]$, etc.

With the same reasoning, we can predict that the instance invocated in $[t_s-p_i, t_s]$ has DBP=1. Otherwise, in $[t_r, t_s-p_i]$, there will be m_i DBP=1 instances. This is in violation of the fact that in $[t_j, t_j+c_i]$, there is already one executed instance, and there will be only m_i-1 DBP=1 instances in $[t_r-p_i, t_s-p_i]$. This reasoning can be extended to instances in $\left[t_s+k_ip_i, t_s+(k_i-1)p_i\right]$ as well.

For τ_i, after the first DBP>1 instance is executed and in $[t_j+c_i, t_d]$, it generates a workload of m_ic_i in every k_ip_i; this is given in equation (19):

$$W_{\tau_i}^1 = \left(\left\lfloor \frac{L-c_i}{k_ip_i} \right\rfloor^+ m_i+1 \right) c_i \qquad (19)$$

where $x^+ = max(0,x)$.

In the fragment (Fig. 15), the τ_i workload is given by:

$$W_{\tau_i}^2 = Min\left(\frac{L-c_i-\left\lfloor \frac{L-c_i}{k_ip_i} \right\rfloor k_ip_i}{p_i} - 1, m_i-1 \right)^+ c_i \qquad (20)$$

So, the total workload caused by τ_i is $W_{\tau_i} = W_{\tau_i}^1 + W_{\tau_i}^2$.

Now, we have to take into account all other tasks with DBP=1 at time t_y. The worst-case is for t_y, such as $[t_j, t_y] = 1$ (see Fig. 16). In $[t_y, t_d]$, we calculate the same as in equations (11) to (17). Some DBP=1 instances occur, at the earliest, at time t_y, while other tasks may have DBP=1 instances after t_y (as shown in Fig. 17).

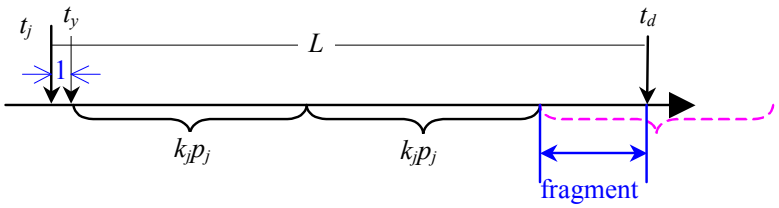


Fig. 16: DBP=1 instances starting time sub-situation-1

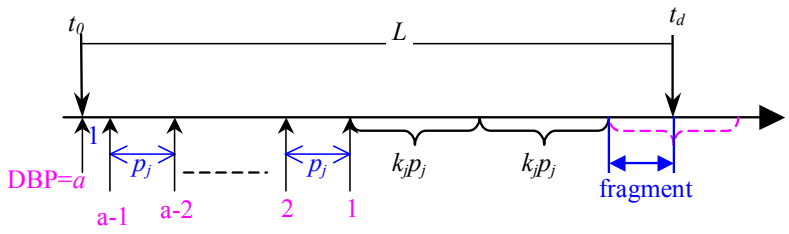


Fig. 17: DBP=1 instances starting time sub-situation-2

Tasks τ_j $(j \neq i)$ with DBP=1 starting, at the earliest, at t_0+l (with $l=1+(DBP_j(t_j)-2)p_j$). The method used to calculate equations (15) and (16) is applied to obtain the total work-load caused by τ_j , and gives formula (21):

$$W_{\Gamma-\tau_i} = \sum_{\tau_j \in \Gamma-\tau_i} \left(\left\lfloor \frac{L-l}{k_j p_j} \right\rfloor m_j c_j + Min\left(\left\lfloor \frac{(L-l)-k_j p_j \left\lfloor \frac{L-l}{k_j p_j} \right\rfloor}{p_j} \right\rfloor, m_j \right) c_j \right) \qquad (21)$$

Then, the total workload is:

$$W' = W_{\tau_i} + W_{\Gamma-\tau_i} \qquad (22)$$

2.3 NP-DBP-EDF Sufficient Condition theorem

Theorem 7: Let Γ be a set of periodic tasks, $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ $(n>1)$, where $\tau_i=(c_i, p_i, m_i, k_i)$, $d_i = p_i$ If the task set Γ satisfies the following conditions C_1 and C_2 in any time interval, then NP-DBP-EDF will schedule any concrete set of periodic tasks generated from Γ, i.e. there will not be any violation of the (m_i, k_i)-firm constraints.

C_1: for any arbitrary time length L: $W \leq L$

C_2: $\forall i, \forall L, L > min_i(p_i)$: $W' \leq L$

where W is given by equation (18) and W' by equation (22).

Proof:

The proof is by contradiction. Assume the contrary, i.e., that Γ satisfies condition C1 and C2 from the theorem, and that there is a concrete set of periodic tasks ωs generated from Γ, such that a task in ω_s falls into failure state, i.e. ω_s has violated the (m, k)-firm guarantee.

We analyze the process of falling into the failure state. Intuitively, the violation of the (m, k)-firm guarantee will happen after a time interval from time 0. Let t_d be the earliest time point where ω_s falls into failure state.

Obviously, only **DBP=1 busy period** leads to (m,k)-firm violation. Starting time t_d we work our way backwards to discover which cases occurred relative to this last DBP=1 busy period, knowing that, for all the possibilities, there are only three cases we could find:

Case 1) DBP=1 busy period starts from an idle time, and all executed tasks have the deadlines before t_d.

Case 2) DBP=1 busy period is blocked by a DBP>1 instance, and all executed tasks have deadlines before t_d.

Case 3) There are some task instances which have deadlines after t_d.

Case 1: In this case, we go backward from t_d to the starting time of this DBP=1 busy period.

This situation happens because, before this DBP=1 busy period, all of the workload has been completed or some workload remains which could not be finished before its deadline and it can be discarded in its tolerable region. So, there is an idle time between this DBP=1 busy period and the previous DBP busy period. (The critical situation is that a DBP=1 instance starts just at the end of the previous busy period, i.e., this idle time is 0. But this does not influence our analysis.) Let L be the length from the starting time t_0 of DBP=1 busy period to the violation time point t_d.

As there is no other idle time in this DBP=1 busy period $[t_0, t_d]$, the total workload in DBP=1 busy period is presented by formula (18). Moreover, since the system falls into a failure state at t_d, we can say that the total workload is definitely greater than the time

interval L, (i.e., $W>L$). However, this contradicts condition C_1 and it establishes the theorem for Case 1.

Case 2: In this case, we go back to the start time of the instance which blocked the last DBP=1 busy period. Let L be the time length from the violation time t_d to the identified time point.

In the interval L, the worst-case of the workload is presented by the formula (22). Since in L there is no idle time (otherwise, our analysis of Case 1 would be directly applied to the interval), and the system falls into a failure state at time point t_d, so $W' > L$. It contradicts our condition C_2 and establishes the theorem for Case 2.

Case 3: In this case, we go back to the last task which had an instance occurring prior to t_d with a deadline after t_d.

Let τ_i be this last task and L ($min_{j\neq i}(p_j)<L<p_i$) be this time length, shown in Fig. 18.

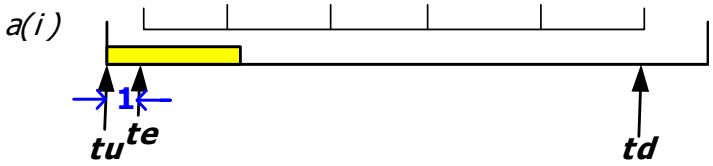


Fig. 18: DBP=1 busy period is blocked by a instance with the deadline after t_d

The workload of task τ_i in $[t_u,t_d]$ is c_i. All the other tasks τ_j ($\tau_j \in \Gamma - \tau_i$) have their deadlines before t_d can add the workload to this period, and their DBP value is superior to 1 at time t_u. Formula (21) gives the workload of τ_j .

Note that the determined time length is limited with $min_{j\neq i}(p_j)<L<p_i$, and if we use this special value in formula (19) and (20), we can derive that $W\tau_i=c_i$. Assuming that a concrete task set leads to a failure state and that there is no idle time in L, we obtain $W' > L$; it contradicts the condition C_2 as well. This establishes the theorem for Case 3.

End of Proof.

Corollary 1: Let Γ be a set of sporadic tasks, $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ ($n>1$), where $\tau_i=(c_i, p_i, m_i, k_i)$, $d_i = p_i$. If the task set Γ satisfies conditions C_1 and C_2 in any time interval, then NP-DBP-EDF will be able to schedule any concrete set of *sporadic tasks* generated from Γ, i.e. there will be not violation of the (m_i, k_i)-firm constraints.

Proof: As the worst case behavior of a sporadic task ("worst" in the sense of requiring the most processing time) occurs when τ_i behaves like a periodic task, that is, τ_i is invocated every p_i time step. Remember that a sporadic task can behave as periodic task. Therefore, as long as condition C_1 and C_2 are satisfied, NP-DBP-EDF algorithm can schedule any concrete set generated from a periodic task set. As we have defined, the arrival curve and the workload of any sporadic task set are always inferior to the periodic concrete set. Whenever a failure state happens, the two conditions have been violated. So, the conditions are also sufficient to guarantee that NP-DBP-EDF will be able to schedule any concrete set generated from a sporadic task set.

End of proof.

2.3.1 Sufficient verification length

As has been shown, in our sufficient condition for NP-DBP-EDF scheduling, all $DBP_j(t)$ are a function of time. Therefore, an interval is necessary to indicate the time evaluation domain. That is to say, we need a ***sufficient length*** for terminating the verification of our sufficient condition.

First, we explain the following definitions:

1) all possible DBP values for one task τ_i with (m,k)-firm constraint:

All DBP values appearing in the scheduling sequence are limited to a natural number in $[0, k_i-m_i+1]$, but not every value will appear in a concrete situation. Because the system falls into a failure state when DBP=0 instance appears, the successful sequence under consideration (no failure state contained sequence) contains DBP values which are limited in $[1, k_i-m_i+1]$. A task, τ_i, has k_i-m_i+1 DBP values in a successful scheduling sequence. In any k_i-m_i+2 instances of τ_i, there must be at least two instances which have the same DBP value in a successful sequence.

2) for a task set with n tasks, *at one time point*, it has $\prod_{i=1}^{n}(k_i-m_i+1)$ successful DBP configuration possibilities.

3) for a strict periodic task set, the inter-distribution of the instances reappears after each LCM (Least Common Multiple) of $\{p_1,...,p_n\}$. Suppose that $t_1,t_2,...t_x$ $(x\in N)$ are the time points with interval LCM, i.e., $t_{i+1} = t_i+LCM\ i\in(1...x)$. Not considering the concrete possibilities, at all time points of $t_1,t_2,...,t_x$, there are at most $\prod_{i=1}^{n}(k_i-m_i+1)$ possible successful DBP configurations. So, in $x=\prod_{i=1}^{n}(k_i-m_i+1)+1$, there must be at least two time points where all instances of the tasks have the same DBP values. And our scheduling can repeat on with the same successive scheduling from the two time points, because at each time point $t_1,t_2,...,t_x$ the inter-distribution of the instances is always the same.

Finally, we can conclude that the sufficient length L_{max} for terminating the verification of our sufficient condition (only for strict periodic task set) is:

$$L_{max} = r + \left(\prod_{i=1}^{n}(k_i-m_i+1)+1\right)\mathrm{LCM} \qquad (23)$$

where r is the last release time.

Obviously, this is a sufficient but not necessary length, because we are considering it from an aspect of permutation. Once at a time point the DBP values of all instances are the reappearance of DBP values, which occurred at a certain LCMs before, the schedulability can already be given. Since in this case, the following sequence will be the iteration of the sequence which took place between the two time points. In practice the test can stop earlier as soon as the repetition occurs for the first time at a multiple of LCM time point.

2.4 Applications of the sufficient condition

In this section, we apply our sufficient condition to dimension the sufficient server capacity for guaranteeing the (m,k)-firm constraint in contrast to that of (k,k)-firm (i.e. HRT). In practice, the dimensioning can be done not only off-line but also on-line. For example, a network supporting real-time QoS should be based on the sufficient con-

dition to decide the acceptance or rejection of a new task (or stream) in its connection admission control procedure; an adaptive real-time system could go from a nominal mode corresponding to (k,k)-firm to a degraded mode, still ensuring (m,k)-firm constraint with the presence of some resource failures.

2.4.1 Bandwidth dimensioning for graceful degradation of QoS

Consider the following networked control system (Fig. 19) where four sensors are connected to a controller via an intranet.

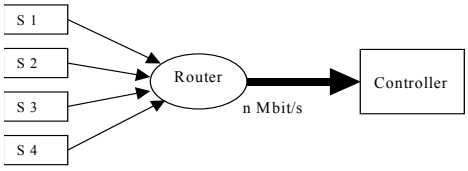


Fig. 19: Application model

Assuming Intserv/RSVP is used at the entrance router[1], the router should reserve a certain bandwidth for guaranteeing real-time QoS. The data packets of sensors should be transmitted to the controller or discarded (during peak network traffic load period but within the (m,k)-firm constraint tolerance region) before the next packet arrives from the same sensor (i.e., deadline is equal to period). The configuration in terms of inter-arrival time, packet size and specified (m,k)-firm constraints is given in Table 3.

	Packet size (kbit)	**Interarrival time (ms)**	**(m,k)-firm constraint**
S1	8	12	(2,5)
S2	8	20	(4,5)
S3	1	5	(1,4)
S4	4	6	(1,5)

Table 3: Parameters of the sources

[1] Despite the scalability problem, we still believe that IntServ QoS architecture can be used within an IP network for factory communication needs since, with its limited size, the scalability problem is not critical.

We will demonstrate the difference in terms of the minimum bandwidth that the router must reserve in order to deterministically guarantee (k,k)-firm and (m,k)-firm constraints. We start our scenario with the worst-case DBP values.

We calculated the cumulative workload during a sufficient schedulability analysis length of *(m_i,k_i)-firm* ($i = 1, 2, 3, 4$). Fig. 20 shows how the router load changes according to the time length L (here for the concrete task set, $L_{max}=9600ms$). The upper curve corresponds to the maximum workload of (k,k)-firm and the lower one corresponds to that of (m,k)-firm. This figure shows that on the average, as shown in the slopes of the lines, (k,k)-firm requires a bandwidth of 1.93 Mbit/s, while (m,k)-firm only requires 0.8 Mbit/s.

Now, to determine the sufficient bandwidth we need to find the greatest upper bound slope. This is given by the highest value of the workload divided by time length L. For both (k,k)-firm and (m,k)-firm cases in our example, these values are found at the beginning of our calculation. Fig. 21 shows the cumulative workload in units of kbit during the first 20ms (a detailed initial view of Fig. 20).

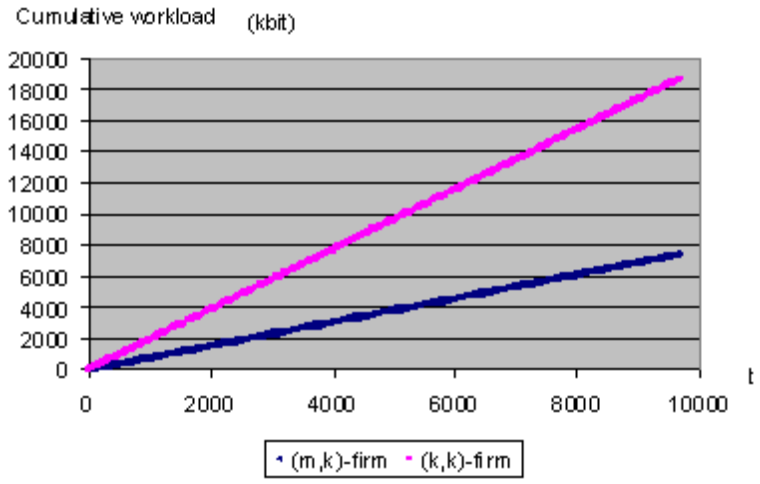


Fig. 20: Router load in average sense

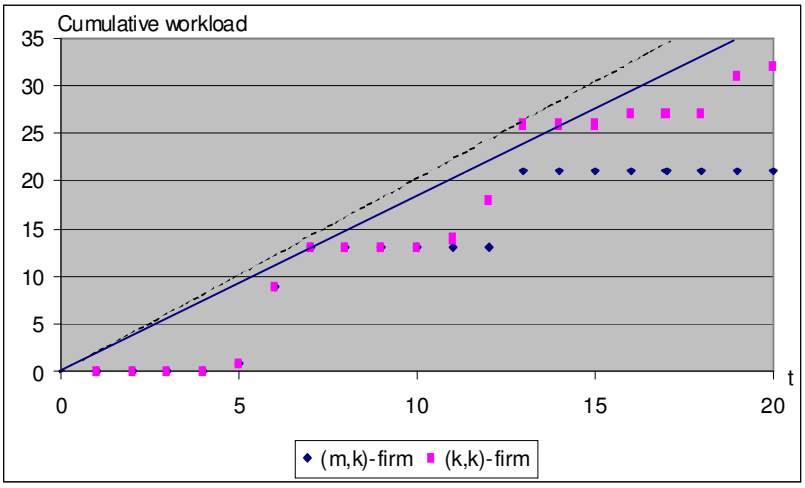


Fig. 21: Deterministically guarantee dimensioning

To guarantee (k,k)-firm, or HRT constraint, Jeffay's conditions must be satisfied. The minimum bandwidth for HRT guarantee is presented by the dotted line upper bounding the (k,k)-firm workload curve (Fig. 21). So, the sufficient bandwidth dimensioned by Jeffay's condition is 2Mbit/s.

For our sufficient conditions of NP-DBP-EDF, there are some concrete parameters such as the set *U* and DBP(t). We do not take into account the actual distribution of the tasks, which can be either inside or outside the set *U*, but include all DBP values starting with the worst values, without considering their roles in a concrete situation (at a critical time point, DBP=2 and changes to DBP=1 after only one time-click). The sufficient bandwidth for (m,k)-firm constraint is presented by the continuous line (Fig. 21). The sufficient bandwidth dimensioned by our sufficient condition is 1.857Mbit/s instead of 2Mbit/s. As calculated, even with the arbitrarily selected parameters, our sufficient condition can still economise 7.15% of the bandwidth.

Using WCIP [Quan00], we can easily understand that the worst execution interference with (m,k)-firm constraint could support the most mandatory consecutive *m* instances among the consecutive *k* instances of another task. That is why the simulation result of our sufficient condition was not dramatically decreased from that of Jeffay's condition in terms of the sufficient bandwidth. But, if we calculate in a concrete situation until we get the sufficient length shown in formula (23), we will have a much smaller sufficient bandwidth. However, for the absolute guarantee in pre-dimensioning, we must

take into account the worst-case router load, which takes place at the initial time region, as shown in Fig. 21.

2.4.2 Overload management in automotive control applications

In this part we show how our sufficient condition can help the dimensioning of the processor capacity in an automotive control system for making it fault-tolerant while using reduced resources.

In in-vehicle embedded system design, the current trend is to use generic processors to replace the specific ones [Wilwert05]. To achieve this goal, OSEK is defined by carmakers and the ECU (Electronic Control Unit) suppliers as the standard operating system [OSEK01]. Moreover, the effort to establish a common platform for supporting portable software modules is continuing inside the AUTOSAR consortium [http://www.autosar.org/]. One of the objectives is to be able to run a car function (e.g. engine control, ABS, etc.) over any generic processor, thus ensuring fault-tolerance when the same function is replicated on more than one processor. All the ECUs are interconnected via a bus (e.g. CAN [RV94] or FlexRay [Flexray19] in the near future).

For making the system fault-tolerant, the classical approach consists in reserving the sufficient spare capacity so that the tasks can be reassigned or re-executed on fault-free processors upon failure detection; without violating any deadlines (i.e. (k,k)-firm). As indicated in our introduction, the drawback of this approach is that the system resources are often underutilized when no faults are present. For the automotive industry where the cost constraints are omnipresent, this approach has not always been acceptable. The approach based on the (m,k)-firm model is more suitable. It consists in invoking an overload management technique upon detection of a failure [Ramanathan99]. Following this approach the system can still work with the presence of some processor failures without necessarily reserving as many resources as used in the classic approach.

The simulation is implemented by taking a case study similar to that of Ramanathan [Ramanathan99], in which the author has shown that the control laws of the automotive control applications can tolerate some deadline misses specified by the (m,k) patterns, without leading to a dangerous situation for the vehicle. Based on our experience in automotive systems [Wilwert05], [Wilwert03], we add another argument that most control loops are based on over-sampling input data (sensor data) to increase dependability.

The occasional loss of some input data will not automatically lead to a dangerous situation.

We then consider a system (Fig. 22) composed of four control functions: cruise control, traction control, braking control and engine control. At first, all four functions are implemented on the four ECU of the system, but only one function is running on each ECU. In case of failure of an ECU, the corresponding function it ensures is woken up on one of the remaining ECU, thus tolerating an ECU failure.

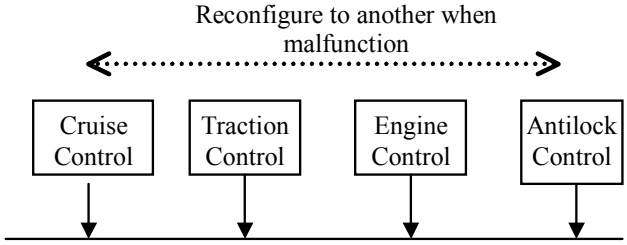


Fig. 22: Vehicle control system model

In what follows, we just consider the extreme case of three simultaneous ECU failures. Our goal is to dimension the processor capacity of an ECU to continue to guarantee meeting of the (m,k)-firm constraint of the four functions. The deadline miss tolerated by each function is assumed to be as given in Table 4.

	Execution time (ms)	Task period (ms)	(m,k)-firm constraint
Antilock control	2	20	(1,4)
Traction control	6	30	(1,4)
Engine control	5	50	(1,4)
Cruise control	6	100	(2,3)

Table 4: Task parameters of control system in vehicle

The target (m,k)-firm constraint for each function can be obtained either by following the control law stability/tolerance study method of [Ramanathn99] or by measuring and simulating the car situations in presence of failures (fault injection) [Wilwert03].

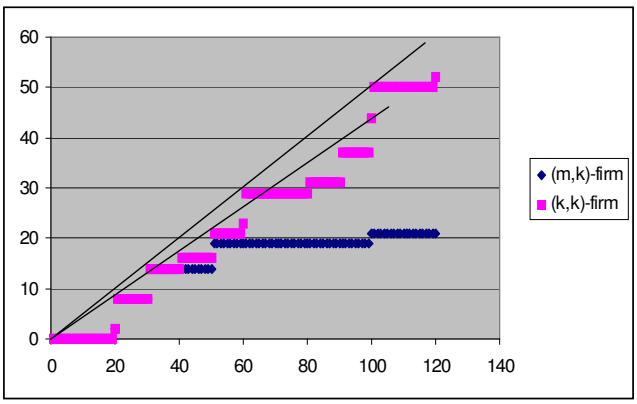


Fig. 23: Workload of (k,k)-firm in contrast of (m,k)-firm for dimensioning system sufficient capacity

The upper line with the slope value 0.495 is the sufficient capacity for the HRT measured by Jeffay's condition. The lower one with the slope value 0.42 is the sufficient capacity for the fault tolerant system in the form of (m,k)-firm. This represents a saving ratio of 15%.

2.4.3 Discussion on the limits of the deterministic (m,k)-firm guarantee

As one can see from the above examples, the advantage of using (m,k)-firm, compared with (k,k)-firm, is not always noticeable. In fact, our sufficient condition and that of Jeffay can even be overlapped in some situations, thus forcing the service of all k tasks even though the system is only under (m,k)-firm constraint. To understand that, let us first take the following numerical example given in Table 5. Four streams (tasks) with (mi,ki)-firm constraints should be executed by the MIQSS model server.

	(m,k)-firm constraint	Processing Time	Period/ Deadline
Stream 1	(2,5)	8	12
Stream 2	(4,5)	10	20
Stream 3	(3,6)	2	5
Stream 4	(1,5)	4	6

Table 5: Parameters of Periodic Task Set

Fig. 24 and Fig. 25 give the cumulative processor demand in time for, respectively conditions C_1 and C_2 in the case of HRT and (m,k)-firm of the concrete task set in Table 5. The x-coordinate represents the time interval (L), and the y-coordinate represents the processor demand which must be executed before the end of L. So, we calculate the changes of this processor demand according to the length of the time interval. In Fig. 24 and Fig. 25 the upper trapezium (solid line) represents the result of HRT under NP-EDF, and the lower one (dashed line) that with (m,k)-firm constraint under NP-DBP-EDF. To start simulation, we assum the worst case for (m,k)-firm by setting all $DBP_i(t) = 1$.

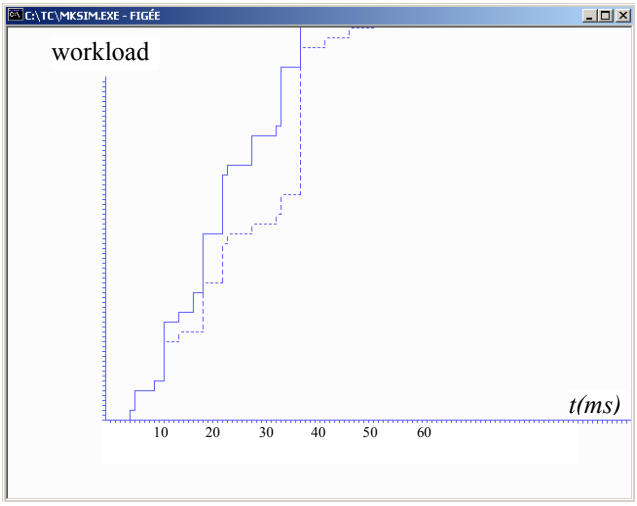


Fig. 24: Difference between conditions 1

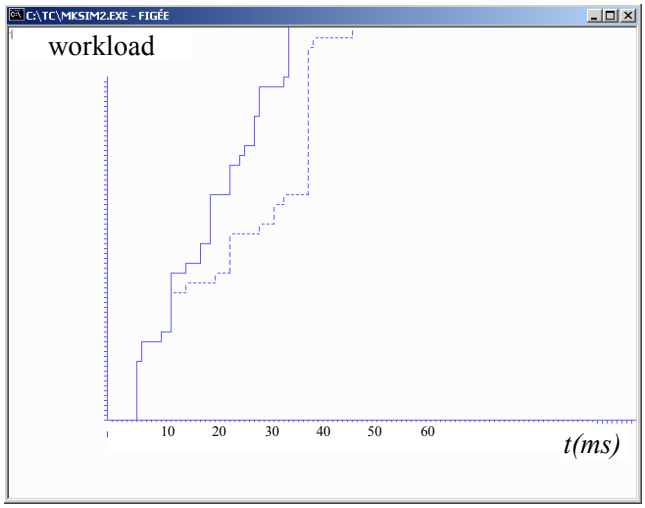


Fig. 25: Difference between conditions 2

From Fig. 24 and Fig. 25, we can see that there is an overlap at the initial time.In fact, condition C_1 of our theorem can be transformed to be like the condition (1) in Jeffay's theorem. Assuming that all tasks of all sources are within the set U (the worst case), and the interval L is less than $min(m_i \cdot p_i)$, we get $\left\lfloor \frac{L}{k_i p_i} \right\rfloor = 0$ and the term $Min\left(\left\lfloor \frac{L - k_i p_i \left\lfloor \frac{L}{k_i p_i} \right\rfloor}{p_i} \right\rfloor, m_i \right) c_i = \left\lfloor \frac{L}{p_i} \right\rfloor c_i$.

The condition C_1 of our theorem has been transformed to the condition (1) in Jeffay's theorem.

Condition C_2 of our theorem can also be transformed to be like condition (2) in Jeffay's theorem.

As we have interpreted, the sufficient condition of (m,k)-firm is always under the bound of Jeffay's theorem, and can reach the bound of Jeffay's theorem with some assumptions and forced evaluation conditions. Notice that these assumptions and forced evaluation conditions can be either realized or not in concrete situations. However, this limits the advantage of using the (m,k)-firm tolerance compared with a system only requiring statistic (m,k)-firm guarantee.

Actually, all (m,k)-firm scheduling algorithms including fixed pattern and dynamic algorithms can achieve the same resource requirement by comparison of HRT in some extreme cases, such that (m,k)-firm losses it basic motivation. In the next chapter, we will deeply analysis theoretically how and why these algorithms can require the same resource capacity

3 Conclusion

In this chapter, we first explained how (m,k)-firm model can be used to define the graceful degradation of real-time QoS, thus allowing the fault-tolerance, and then addressed the problem of the deterministic guarantee of (m,k)-firm real-time requirements for a set of periodic or sporadic tasks sharing a common server.

We addressed the problem of the deterministic guarantee of (m,k)-firm real-time requirements for a set of periodic or sporadic tasks sharing a common server. DBP has been chosen for its interesting feature of dynamically assigning priorities based on the

previous history of the system (k-sequence). This makes it suitable for QoS management in adaptive real-time systems and networks. We have proposed a sufficient condition to off-line evaluate the schedulability of task set under DBP scheduling algorithm for (m,k)-firm constraint. In addition, the schedulability can also be determined online within a limited time interval if all release times are given. Simulations approve the off-line sufficient condition in a network control system and in an automatic control system.

Although we are interested in predicting the schedulability of the task set, it is found that the online sufficient condition to determine the NP-DBP-EDF schedulability is much more efficient in terms of resource requirement than the off-line one. Additionally, DBP scheduler could requires the same amount of resource in comparison of HRT (theoretical reason is shown in Appendix B). Moreover, it has been proven in our report [Li03] that DBP scheduling may fail into failure state even with arbitrary low utilization. Therefore, together with the results in chapter 1, it is demonstrated that the low utilization is inevitable for (m,k)-firm constrain in general case. This leads us to research in the theoretical reason of low utilization and find the effective resolution.

Chapter 3

Challenge in Low Utilization Resolution

As observed from the results in previous chapters, the (m,k)-firm constraint is not efficient in terms of bandwidth utilization. This chapter will focus on the theoretical reasons of this low resource utilization and seek an effective solution for selective loss constraint and resource allocation.

First, we will have a deep analysis on the elements by which resource requirement is decided. Intuitively, a task set with a (m,k)-firm constraint should induce less workload and thus require less resources. However, in the previous chapter it has been highlighted that (m,k)-firm constraint is generally not able to save the resource with regard to HRT when the aim is to provide deterministic guarantees. Even worse, the (m,k)-firm constraint can always be violated for a task set with arbitrarily low workload [Quan00] [Mok01] [Li03].

The most general real-time schedulability problems have been shown to be NP-Hard in the strong sense (see for instance [Jeffay91] [Mok01] [Garey77,78]) and the future work in this field will face the intractability issue. This complexity issue will be discussed in the following and the results will be reminded. Afterwards, we outline the promising research perspectives which we believe for the NP-hard scheduling problems. In the light of these perspectives, we summarize the existing strategies in real-time scheduling that are aimed to achieve higher resource utilization.

1 Workload, resource requirement and utilization

Admission control and resource reservation can provide Quality of Service (QoS) for real-time traffics in the network. These protocols should primarily dimension the required resource for a real-time stream according to its characteristics. Recall that, in this thesis, techniques originating from the field of real-time systems are used to analyse and handle multimedia transmissions in order to provide deterministic guarantees rather than statistic or best effort guarantees. Hence, predictability is necessary in resource management, which means the pre-dimensioning of the least resource requirement for a given task set. That is to say, for a given non-concrete task set, it is necessary to find the resource requirement such that all generated concrete scenarios will be schedulable. However, if one looks at the existing schemes (DBP, DWCS, etc), deterministic guarantees for streams with real-time QoS are only achieved at the expense of a considerable over-provisioning of resource allocation [Koubaa04a] [Mok01].

Traditional MIQSS systems possess a server with a fixed capacity, such that the execution time of every task is also fixed. In this case, the research focuses on the schedulability determination as well as the worst-case response time evaluation. In this section, we will analyse what is the least resource requirement needed for guaranteeing a task set with a certain amount of workload. Precisely, the server capacity is not determined beforehand. This implies to rework the traditional periodic/sporadic task model (i.e. the computation time varies depending on the server processing capacity).

1.1 Task re-model according to workload

Recall that real-time task τ_i is typically modelled as (c_i, p_i, d_i, m_i, k_i), where c_i is the execution time of an instance on a server with a given capacity or the transmission time in a router with a given bandwidth. However, if the resource capacity is unknown, then c_i is a variable. In fact, a stream sends a packet to the network with a quantity of work B_i (the number of bits of a packet in a multimedia transmission), which is always a constant. Let R denote the required resource (processor capacity or bandwidth). Obviously, for the packet to be processed by the server within time c_i, the following relation holds:

$$c_i = B_i / R. \tag{24}$$

The *workload* of periodic task set $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ under HRT constraint and (m,k)-firm constraint are given by formulae (or formulas) (25) and (26), respectively:

$$\sum_{i=1}^{n} \frac{B_i}{p_i} \tag{25}$$

$$\sum_{i=1}^{n} \frac{m_i}{k_i} \frac{B_i}{p_i} \tag{26}$$

Note that, with sporadic tasks, the next instance's arrival is not predictable, since there is a just a restriction on the minimum inter-arrival time. Therefore, sporadic task's arrival rate (with the same other parameters) is always below the periodic task's arrival rate, such that sporadic task's workload is always smaller than (25) and (26).

In order to compare the efficiency of different scheduling policies, we also consider the *utilization* of the system resource which is defined as the ratio of the workload over the resource requirement. Therefore, the utilization under HRT constraint and (m,k) constraint are expressed respectively as:

$$\left(\sum_{i=1}^{n} \frac{B_i}{p_i} \right) / R = \sum_{i=1}^{n} \frac{c_i}{p_i} \tag{27}$$

$$\left(\sum_{i=1}^{n} \frac{m_i}{k_i} \frac{B_i}{p_i} \right) / R = \sum_{i=1}^{n} \frac{m_i}{k_i} \frac{c_i}{p_i} \tag{28}$$

Observe that both the right hand sides of (27) and (28) have been proposed somewhere [West04] [Koubaa04a,b] [Quan00] as utilization of the system. An effective scheduling algorithm should guarantee the real-time constraints of a task set with the smallest amount of resource, R.

2 Problem definition

In what follows, we will discuss the relationship among the resource requirements, workload and utilization. For a given task set, the workload is already fixed, but the resource requirement is difficult to determine. After dimensioning resource for the

given task set, the utilization can be easily obtained according to (27) and (28). Notice that the utilization cannot be greater than 100%, which is a necessary condition for all scheduling algorithms. Hence, the research objective is to achieve utilization as close as 100%.

Unfortunately, the results are usually not exciting in terms of utilization until now, since deterministic guarantee can only be achieved by an enormous over-provisioned resource reservation (this causes the low utilization). Moreover, all current scheduling algorithms for (m,k)-firm constraint also lead to a poor utilization ratio.

2.1 Relation between workload and resource requirement

As yet, the current network protocols allocate the resources according to the workload of the streams, which is fine to provide statistical guarantees with respect to (soft) real-time constraints. However, bandwidth alone is not enough to provide deterministic guarantees.

In what follows, we will explain that the resource requirement for deterministic guarantees does not only depend on the task set workload. The other elements which affect the temporal correctness in non-preemptive scheduling are highlighted on some obvious examples. The research is also carried out for HRT scheduling, since it is the basic model for (m,k)-firm window-constraint without taking into account the loss tolerance.

The following three observed (negative) properties explain the reasons which cause the low utilization. Each property is then illustrated with a typical example. All the properties have their theoretical supports [LiRTNS06] [Jeffay91] [George00] [Mok01] [Li03], which will be detailed in the next section.

2.1.1 Low utilization phenomenon 1

Property 1: the resource requirement depends not only on the workload of the task set, but also on the inter-distribution of tasks' instances (release times distribution).

Illustration 1: for the same non-concrete task set (the same period and execution time), the tasks might be schedulable in one scenario with certain release times but not be schedulable in another scenario.

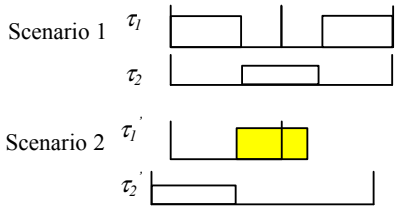


Fig. 26: Low Utilization Phenomenon 1

Fig. 26, if two HRT tasks are released at the same time, then they are schedulable as shown in concrete scenario 1. However, if τ_2' is released a little earlier as shown in scenario 2, then they are no longer schedulable. The process that advances the arrival time of the first instance of τ_2 with regard to τ_1 is a change in the inter-distribution of tasks instances. Clearly, in Illustration 1, the concrete task sets have the same workload, but the schedulability depends on the precise concrete scenario (different inter-distribution of tasks' instances). To let scenario 2 lead to a feasible schedule, more resources are needed and the utilization is thus reduced.

This phenomenon can also illustrate that for periodic task set [Jeffay91] the difficulty of scheduling tasks can be affected by the times that the tasks are released. However, the sporadic task's next instance has just been restricted by the minimum inter-arrival interval, and the worst-case inter-distribution is always subject to occur, such that the utilization could always be low.

2.1.2 Low utilization phenomenon 2

Property 2: A task set with less workload might be more difficult to be scheduled.

Similarly, we will use an example to demonstrate Property 2.

Illustration 2: see Fig. 27, two scenarios are shown for the two different task sets, τ2' in scenario 2 has doubled period and 1.1 time of execution time by comparison with τ_2 in scenario 1. Meanwhile, τ_1 *and* τ_1' have the same parameters. Then, the workload of the task set in scenario 2 is inferior to that of the task set in scenario 1. However, in Fig. 27, task set in scenario 1 is schedulable while the task set in scenario 2 is not. This demonstrates that a task set with less workload could be more difficult to be schedulable.

In addition, it is also proven that a task set with an arbitrarily low workload could still be non-schedulable [Li03] [Mok01].

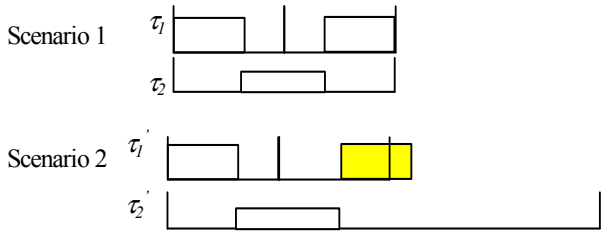


Fig. 27: Low Utilization Phenomenon 2

(m,k)-firm real-time is proposed to tolerate some deadline misses, so that it is expected to resolve the over-provisioning problem. However, another phenomenon will show the limits of the (m,k)-firm constraint in terms of resource utilization.

2.1.3 Low utilization phenomenon 3

Property 3: taking into account the worst-case, a task set under (m,k)-firm constraint cannot economize the resource in comparison with the same task set under HRT in general case.

Illustration 3: in Fig. 28, a task set of two tasks, with (2,3)-firm constraint, is not schedulable in the scenario. With whatever scheduling approach, one mandatory instance of the two tasks is invocated to be finished in the same period. Therefore, if (2,3)-firm constraint is satisfied, (3,3)-firm constraint will also be satisfied. This fact obliges to reserve resource according to HRT constraint in order to guarantee an (m,k)-firm constraint, which is quite undesirable.

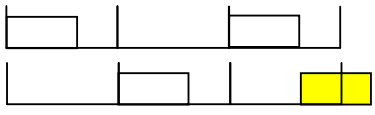


Fig. 28: Low Utilization Phenomenon 3

In fact, we have proven that it is always possible to find a (m,k)-firm constraint

set for a given task set, such that the resource requirement for (m,k)-firm constraint is not less than the HRT constraint (this will be discussed in detail in next section).

Above all, it is incorrect to conclude that over-provisioning problem is inevitable in order to get deterministic guarantee. Moreover, with some loss tolerance such as (m,k)-firm constraint, the resource reservation also cannot be improved for deterministic guarantee. Afterwards, it is necessary to know the theoretical reasons.

2.2 Low utilization theoretical reasons

In this section, we will introduce the theoretical reasons which cause the low utilization for either HRT or (m,k)-firm constraint. These theoretical reasons can explain the dependent elements of low utilization, which are introduced in the previous section.

2.2.1 Low utilization in HRT

Recall that in [Jeffay91], a sufficient schedulability condition is given for a general set of periodic or sporadic tasks under EDF scheduling for HRT constraint.

Theorem 1 [Jeffay91]**:** Let $\Gamma=\{\tau_1, \tau_2, ..., \tau_n\}$ be a set of *sporadic or periodic* tasks sorted in non-decreasing order by periods (*i.e.*, for any pair of tasks τ_i and τ_j, if $i < j$, then $p_i \leq p_j$). If Γ satisfies conditions (1) and (2), then the non-preemptive EDF scheduling algorithm will schedule any concrete set of periodic or sporadic tasks generated from Γ.

(1) $$\sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1$$

(2) $\forall i,\ 1 < i \leq n;\ \forall L,\ p_1 < L < p_i$:

$$c_i + \sum_{j=1}^{i-1} \left\lfloor \frac{L-1}{p_j} \right\rfloor c_j \leq L$$

Jeffay also proposed that this sufficient condition is necessary as well for a sporadic task set, while it is not a necessary condition for a periodic task set. Observe that periodic task set has more workload than a sporadic task set with the same parameters,

but periodic task set tends to require less resource than sporadic task set. These imply the following impact between workload and system resource requirement.

Theorem 8: With the same parameters, a periodic task set and a sporadic task set have follows attributes:

$W(periodic) \geq W(sporadic)$

$R(periodic) \leq R(sporadic)$

where *W(periodic)* and *W(sporadic)* stand for the workload submitted to the system by periodic and sporadic task set, respectively while *R(periodic)* and *R(sporadic)* stand for the requirement of server capacity (resource requirement) to deterministically guarantee periodic task set deadlines and sporadic task set deadlines respectively .

At a glance of Jeffay's theorem, the first condition denotes the maximum workload of task set, which can be either periodic or sporadic. However, this is not the only condition that decides the feasibility. [George00] indicates how to find the worst-case response time during a critical inter-distribution of instance.

Theorem 9: [George96]**:** For non-preemptive EDF, the worst-case response time of a task τ_i is found in a deadline busy period for τ_i in which τ_i has an instance released at time a (and possibly others released before), all tasks with relative deadline smaller than or equal to $a+d_i$ are released from time $t = 0$ at their maximum rate, and finally a further task with relative deadline greater than $a+d_i$, if any, has an instance released at time $t=-1$.

In fact, George's theorem (Theorem 9) describes a critical inter-distribution of tasks' instances, in which the worst-case response time occurs. Additionally, the most highest capacity of the server is necessary when the worst-case occurs, and it is just required by the second condition of Jeffay's theorem (Theorem 1) under EDF scheduling.

Moreover, this fact exists as for other Fixed Priority scheduling schemes such as rate monotonic, deadline monotonic, etc.

Theorem 10: [George96] Let $\Gamma=\{\tau_1, \tau_2, ..., \tau_n\}$ be a general task set with arbitrary fixed priorities, then the worst-case response time of τ_i is found where all tasks with higher priority are released synchronously from time $t = 0$, and the longest task which has lower priority, if any, at time $t=-1$.

The upper theorem can explain that the resource requirement depends not only on the workload of task set, but also the inter-distribution of tasks' instances (Property 1). Since a concrete periodic task set could probably avoid upper critical inter-distribution, while a sporadic task set (either concrete or non-concrete task set) always suffers from the worst-case inter-distribution. So, it can explain why a periodic task set has a higher workload but tends to require less resource.

So far, it seems interesting to find how much resource can be economized for a concrete periodic task set in comparison with the sporadic task set. Unfortunately, it has been proven that non-preemptive scheduling of a concrete periodic task is NP-hard in the strong sense [Jeffay91]. Therefore, it is obliged to reserve the resource according to the sporadic task set, which causes the considerable over-provision and the low utilization factor.

2.2.2 Low utilization in (m,k)-firm constraint

The considerable over-provisioning is inevitable in HRT. In this section, we will demonstrate that the over-provisioning for (m,k)-firm task set is more serious than HRT.

A task under (m,k)-firm constraint inherits the HRT task model by putting on (m,k)-firm constraint, such that a task set under (m,k)-firm constraint submits less workload to the system in comparison with a task set under HRT constraint. Intuitively, a task set under (m,k)-firm constraint should require less resource than HRT. Unfortunately, we have proven in [Li03] that if we can configure the (m,k)-firm constraint of a given task set (the (m_i,k_i)-firm constraints for every task are modifiable), where $m_i < k_i$, then there is always a (m_i,k_i)-firm constraint set for every task for which the resource requirement for (m,k)-firm constraint equals that for HRT (see Appendix B). In this case, (m,k)-firm con-

straint loses its basic interest. In fact, since (*m,k*)-firm and window-constraint can be transformed to each other, this pessimistic result exists equally in window-constraint.

Furthermore, it has been proven in [Li03] [Mok01], that, even under arbitrarily low workload, a non-concrete task set under (m,k)-firm constraint or window-constraint could turn into failure state.

Theorem 11 [Mok01]: Given an arbitrary non-negative real number u and a natural number g, there exists a unit-size Window-Constrained task set Γ, such that the aggregate utilization rate of Γ is less than or equal to u, and Γ is not schedulable by EDF based on window on g processors.

This result means the arbitrary low utilization is not evitable. Then, it is natural to find the smaller resource requirement for a task set under given (m,k)-firm or window-constraint in concrete situations, but this problem has been proven as NP-hard in strong sense.

Observe that a sufficient schedulability condition, given in [West04] for DWCS-2, has improved the utilization factor to 100%, which seems incompatible with the above results. We found that this sufficient and necessary condition is effective only in some special cases, which have been talked about after the presentation of DWCS-2 (in [West04]). For example, tasks must be synchronized and must have the same periods and unit size execution time. In addition, we reconstruct the theorem since the proof is incorrect (see Appendix A). This theorem illustrates the property that P⊂NP, this will be analyzed with more details in Section 3.

2.3 Summary of low utilization causes

As mentioned above, although sporadic task set has less workload than periodic task set, it requires more resource than periodic task set for the purpose of deterministic guarantees.

Moreover, it has been proven that finding the exact resource requirement of concrete periodic task set is a problem of NP-hard in strong sense. In other words, getting the sufficient and necessary condition for periodic task set cannot be expected.

More generally, [Quan00] and [Mok01] have proven that the schedulability of (m,k)-firm and window-constraint are problems of NP-hard in strong sense. These pessimistic results show that the deterministic guarantee for (m,k)-firm must be provisioned according to HRT. Fig. 29 shows a sketch relationship among workload, potential resource requirement and deterministic guarantee resource.

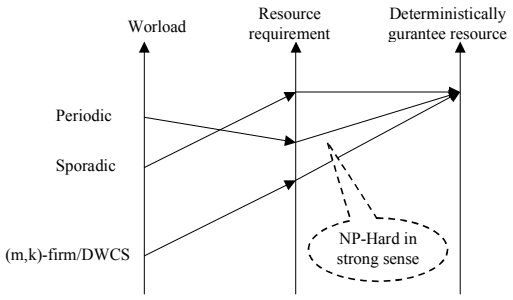


Fig. 29: Sketch relationship between workload and resource requirement

The left-most vertical line shows periodic, sporadic and (m,k)-firm task sets in the descending order of workload quantity. The middle vertical line shows that the periodic task set with the highest workload potentially requires less resource, while sporadic (m,k)-firm task set with the least workload potentially requires the most resources.

Unfortunately, as the potentially less resource requirement is proven as a NP-hard problem, deterministic guarantee resource will be obliged to be provisioned according to the worst-case (the same as the requirement for sporadic task set). Accordingly, the right-most vertical line shows the deterministic guarantee resource requirement

Therefore, the periodic yields a higher utilization than the sporadic while (m,k)-firm has the least one. Globally, current real-time system's relationship (or, correlation) between workload and utilization has been shown. Notice that it is of course a negative result to see that a system cannot benefit from a task set with less workload, and for any task set it must dimension a much greater resource to obtain deterministic guarantee.

3 Analysis in Complexity

In this section, we will resume the complexity of schedulability for a real-time task set. The results may have been introduced before, but a recall is necessary at this

point. In addition, the deeper analysis will be conducted for the perspectives of future work.

3.1 NP-hard in real-time

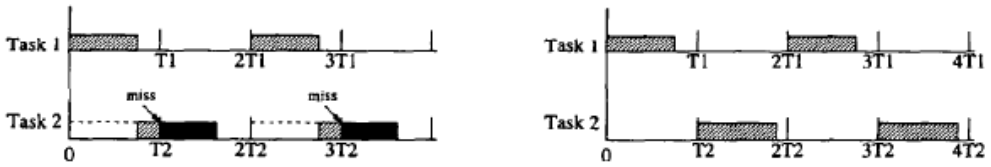
In HRT, the problem of schedulability under non-preemption to guarantee all deadlines is always an interesting problem. This problem has attracted research since the 1970's. In [Garey1977], it has been proved that the problem of schedulability for a task set under non-preemption is NP-hard in the strong sense, which corresponds to most usual context in real-time.

Intuitively, when the schedule takes into account the rejection of some instances, the problem will become more complicated. Actually, in [Quan00] and [Mok01] it has been proven that schedulability of (m,k)-firm and window-constraint are also problems of NP-hard in strong sense.

Faced to this low utilization problem and intractability (due to NP-hard), three research directions are possible. As previously mentioned, the results exist since 1970's, and some perspective directions have been already demonstrated as well.

3.1.1 First research direction

The first perspective research direction is to heuristic methods. The most important point about heuristic approach is that it involves giving up the certainty of achieving a solution, and provides an algorithmic program in order to achieve a much better solution on taking advantage of the possibility.



(a) Using 10 as the (m,k)-patterns for both Task 1 and Task 2 results in an infeasible schedule

(b) Using 10 as the (m,k)-pattern for Task 1 and 01 for Task 2 results in a feasible schedule

Fig. 30: Different (m,k)-patterns for the same task set lead to different scheduling result

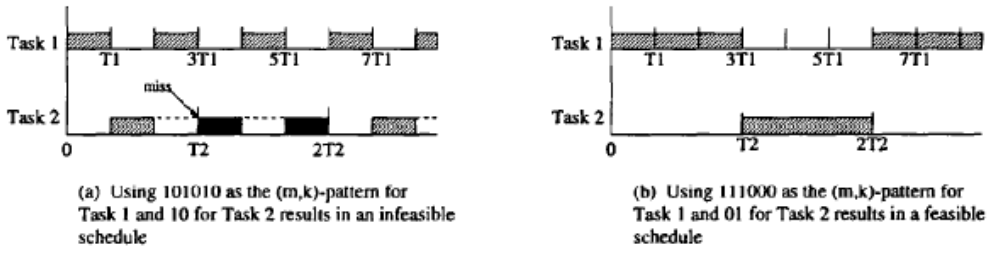


Fig. 31: Evenly distributed mandatory instances may not always improve the schedulability

For better fixing the problem we want to deal with, let's consider an example. Fig. 30 and Fig. 31 show some examples that the different patterns lead to different schedulability for a given task set, and even distribution of mandatory instances is not always better than others. In general, the general non-preemptive periodic task set scheduling under (m,k)-firm constraint has been proven as a problem of NP-hard in strong sense [Garey77] [Jeffy91]. Therefore, there is no optimal pattern for a given task set.

However, the Enhanced Fixed Priority pattern [Quan00], introduced in section 4.1.4 of chapter 1, can be considered as one way to deal with this NP-hard problem. Two heuristic methods have been used by the authors to resolve the following problem, which only give of course sub-optimal solutions.

Firstly let's recall that in [Quan00], Enhanced Fixed-Priority (m,k)-pattern is proposed by rotating right *Sche_mkfirm pattern* with the adding factor s_i:

$$\pi_{ij} = \begin{cases} 1 & if \quad j = \left\lfloor \left\lceil \frac{(j+s_i)\times m_i}{k_i} \right\rceil \times \frac{k_i}{m_i} \right\rfloor - s_i \\ 0 & otherwise \end{cases} \qquad j = 1,2,\mathrm{L}\ k_i$$

sche_mkfirm pattern always assigns the first instance as mandatory, such that the Worst Case Interference Point (WCIP) can occur among the first instances of tasks. Based on sche_mkfirm, EFP (m,k)-pattern does some efforts to rotate the mandatory instances of tasks in order to avoid the WCIPs. The possible values of s_i are ($0...k_i$-1) for task τ_i, and the problem is how to configure every s_i for task set for separating the WCIPs.

Therefore, EFP algorithm behaves significantly better than Sche_mkfirm algorithm in terms of system resource utilisation. However, EFP the combination to examine

all possible values of s_i is exponential. So the problem the authors in [Quan00] have had to resolve is how to find quickly the values of s_i which give satisfying sub-optimal scheduling.

We will not detailed in our thesis these heuristic configurations of s_i, because the heuristic algorithms are generally not suitable for on-line QoS control as with long computing time, and it is not suitable for off-line pre-dimensioning since some parameters cannot be determined for non-concrete tasks, for example the release times can only be given in concrete cases.

3.1.2 Second research direction

The second potential research direction is to work on the best way to *specify the task parameters*. As known, a general problem of NP-hard in strong sense can still be investigated according to tentative views of the NP world such as P⊆NP.

The general non-preemptive periodic task set scheduling is proven as NP-hard by mapping to the 3-partition problem, which is defined as follows:

3-Partition: It is defined as a finite set A of $3m$ elements, a bound $B \in Z^+$, and a "size" $s(a) \in Z^+$ for each $a \in A$, such that each $s(a)$ satisfies B/4 < s(a) < B/2 and such that $\sum_{a \in A} s(a) = mB$. Partition A into m disjoint sets $S_1, S_2, \ldots S_m$ such that, for $1 \le i \le m$, $\sum_{a \in Si} s(a) = B$.

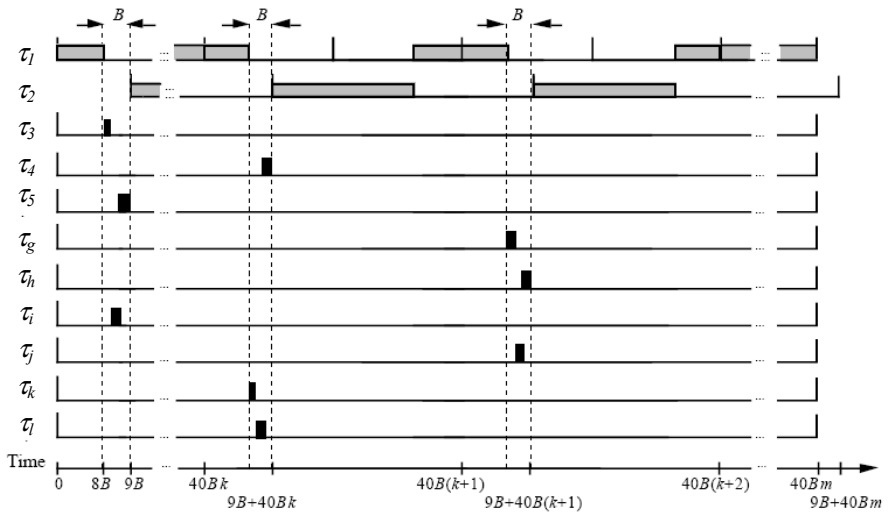


Fig. 32: Mapping non-preemptive periodic task set scheduling problem to 3-Partition problem

The transformation is performed as follows [Jeffay91]. Let $A = \{a_1, a_2, a_3, ..., a_{3m}\}$, $B \in Z^+$, and $s(a_1), s(a_2), s(a_3), ..., s(a_{3m}) \in Z^+$, constitute an arbitrary instance of the 3-Partition problem. We create an instance of the scheduling of non preemptive concrete periodic task problem by constructing a set ω_p of $n = 3m + 2$ concrete periodic tasks. Let $\tau_p = \{\tau_1, \tau_2, ..., \tau_{3m+2}\}$, where ($\tau = (c,p)$)

$\tau_1 = ((8B, 20B),$

$\tau_2 = ((23B, 40B)$, and

$\forall j, 3 \leq j \leq 3m+2$: $R_j = 0$;

be a set of concrete sporadic tasks. The construction of the set ω_p can clearly be done in polynomial time with the largest number created in the new problem instance being 40Bm. See from Fig. 32, the task τ_1 and τ_2 have the highest priority, and they occupy the scheduling time remaining m idle blocks with the size of 1. Thus the task set scheduling can be transformed to 3-Parition problem. The reader can refer to [Jeffay91] for the details of the mapping procedure.

Obviously, if every element size is fixed as B/3 in 3-partition problem, it is easy to satisfy the problem by any partition method. Hence, the polynomial resolutions are proposed according to this unit size strategy. In HRT, if all task lengths are 1, or preemptions are allowed, or all release times are 0, the general problem can be solved in polynomial time to meet all deadlines under non-preemptive scheduling [Lawler73], [Lageweg76].

Note that unit size computation time '1' breaks the difference between preemption and non-preemption. Meanwhile, it is known that EDF is optimal for preemption scheduling and the utilization can reach 100%. Furthermore, as long as all tasks have equal length [Carlier78] [Simons78] [Garey78] or preemptions are allowed [Blazewicz76], it can also be solved in polynomial time even if release times and deadlines are allowed to be arbitrary rationales.

The authors of DWCS-2 [West00] [West04] followed the same strategy of unit size computing time by specialying that all the tasks must have the same transmission time and the same period, then the utilization factor can achieve 100%. However, this very particular task model cannot be directly applied to the multimedia transmission, where each multimedia stream could have different packet size and period. Moreover, to

satisfy the sufficient condition, all task periods must be synchronised, which cannot be realised in real multimedia transmission networks because the jitters is not evitable.

The result in [West04] seems interesting and correct, but we found that the proof of the theorem is incorrect. Moreover, in [West04] the release times are synchronized at time 0 in order to prove the theorem. To fix the problem, we have given a complete proof and shown that it is still tenable for arbitrary release time (Appendix A).

However, for the general task model, DWCS or DBP scheduling can fall into failure state even with arbitrarily low processor utilization [Mok01] [Li03]. Actually, R. West followed the strategy proposed in 1970's. For example, the task set schedulability problem can be solved in polynomial time so long as all tasks have equal length as addressed in [Carlier78], [Simons78], [Garey78]. Therefore, DWCS can achieve 100% utilization with graceful degradation [West04] according to this polynomial problem task model.

3.1.3 Third research direction

The third way is to modify and relax the real-time constraint so as to achieve high utilization.

As known, the deadlines associated to every instance of task complicate the scheduling problem. In fact, the individual deadline approach is not the holy grail of real-time communication, and there have been a lot of works which tried to relax the deadlines to gain higher utilization factor, such as frame-based model [Liberato99], Pinwheel scheduling [Hotel89] and virtual deadline window scheduling [Zhang04], etc.

The frame-based model defines a set of tasks, which is to execute within each frame (time window) and is to complete before the end of frame. The problem is to schedule the task set in a single frame with deadline D.

Frame-based model

In a frame-based model, a schedule is built for the length of a frame and each task is assigned processing time in this schedule. Because one can specify the earliest and latest times, each task can be scheduled within a frame. The frame-based model allows multiple instances of periodic tasks with small periods to be included in multiple non-overlapping time intervals within the frame by treating each instance as a separate aperi-

odic task. This method can schedule any set of periodic tasks, if the frame length is a multiple of every period in the set.

A special case of Frame-based model is proposed in [Liberto99]. It assumes that all task instances are stored in a single queue at the beginning of the frame (i.e. released at the same time at the beginning of the time window). In this case, the utilization of processor can reach 100%, and can be resolved in polynomial time. Clearly, if the total tasks' instances size is inferior or equal to the frame size, any non-idle scheduling can treat with them without any constraint violation. The scheduling in a frame can be repeated to do with infinite task instances.

Note that frame-based model removes the deadlines of instances to provide a scheduling in the same frame size for all tasks, in other words, it deal with the periodic task set as aperiodic task set. It is suitable for some kinds of real-time applications, but various multimedia applications are not suitably analyzed by the frame-based model. Anyway, it is a successful research way and inspiring method.

Pinwheel

In our opinion, Pinwheel model [Hotel89] is more interesting to reserve different quantity of time in different frame size for each task. The generalized pinwheel scheduling problem is an offline scheduling for satellite-based communication as follows: Given a multiset $\{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$ of ordered pairs of positive integers, determine an infinite sequence over the symbols $\{1, 2, 3, ..., n\}$ such that, for each i, $1 \leq i \leq n$, the Pinwheel constraint requires that any subsequence of b_i consecutive intergers contains at least a_i times i . Differing from frame-based model, Pinwheel guarantees the execution time in a sliding window.

Pinwheel is designed for satellite-based communication model which only concerns unit size execution time. They cannot service current diverse multimedia applications. Together with Pinwheel model, frame-based model can find optimal scheduling scheme, and their utilization factor can reach 100%.

Instead of schedulability and high utilization, the research of Pinwheel focuses on the fairness allocation for each task on the same pinwheel. Intuitively, if the tasks scheduled in a window are bundled up to the front or to the end of the window, the schedule does not satisfy the property of proportional progress.

For example in Fig. 33, given a multiset of *{(2, 6),(2, 6),(2,6)},* the scheduling for the three tasks in the first pinwheel is considered as more fairly allocated than the second pinwheel. Then, the P-fairness constraint is proposed to restrict the fair allocation rather than the research on the optimisation and the utilization.

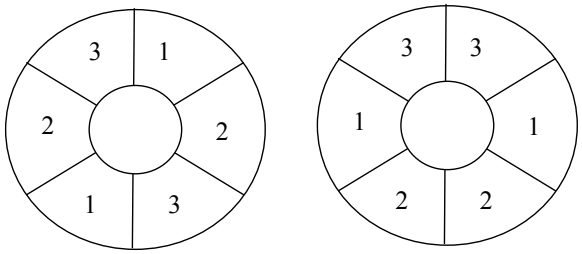


Fig. 33: Fairness of Pinwheel

P-fairness (m,k)-firm scheduling

The concept of Pfairness was introduced in [Baruah96], in the context of constructing periodic schedules for a system of periodic tasks on several identical processors, where the multiprocessor periodic scheduling problem [Liu69]. Theorem 12 that follows was proven by means of some fairly involved network-flow constructions, and by using the Integer Flow Theorem [Ford62]

P-fairness is a strict proportionate progress property. A schedule is *Pfair* if and only if the schedule satisfies not only the constraints defined in the definition of unit-size schedule, but also the following extra constraint: For every task τ_i and every integer l, the l_{th} instance scheduled for task τ_i is scheduled in the l_{th} *Pfair scope* of τ_i, which is defined as:

$$\left[r_i + \left\lfloor \frac{l \cdot b_i}{a_i} \right\rfloor \cdot p_i, r_i + \left\lceil \frac{(l+1) \cdot b_i}{a_i} \right\rceil \cdot p_i \right] \qquad (29)$$

The expression $r_i + \left\lceil \frac{(l+1) \cdot b_i}{a_i} \right\rceil \cdot p_i$ will be referred as the *deadline* of the l_{th} Pfair scope of task τ_i.

Wang and Mok modified window-constraint value $W_i=x_i/y_i$, the tolerable loss in the window, with the value of the "must execute instance" number in the window. In or-

der to avoid the confusion, we name it as (m,k)-window, which requires to execute at least *m* instance before their deadlines in the fixed no-overlapping window of *k*. Known that (m,k)-window differs from (m,k)-firm constraint, since (m,k)-window constraint requires m instances must be schedule in any k consecutive one, which implies a sliding window.

Theorem 12 [Mok01]**:** If $c_i = 1$ for all $\tau_i \in \Gamma$, *and* $\sum_{i\in\Gamma} \frac{m_i}{k_i \cdot p_i} \le g$ where g is the number of processors*, then there exists a Pfair (m,k)*-window constraint schedule *for* Γ.

In [West04], DWCS is proposed as an optimal scheduling algorithm for the special task set, in which all tasks must have the unit size execution time and the same period (original proof is wrong, correct proof is given in appendix A). Furthermore, in [Mok01], the authors proposed some other special cases for Pfair window-constraint scheduling, expressed by theorems 13 and 14.

Theorem 13 [Mok01]: If $c_i = 1$ and k_i is a multiple of m*i* for each task τ_i, there exists an optimal online Pfair (m,k)-window scheduling algorithm *A*, and the worst case online computational cost of A is *O(n)* per time slot.

Theorem 14 [Mok01]**:** If, for each task τ_i, $r_i = 0$, $c_i = 1$, and $p_i = p$, where p is a constant, there exists an online optimal Window-Constrained P-fair scheduling algorithm A and the worst case online computational cost of A is *O(n)* per time slot**.**

Similar to *P-fair scope*, another online schedule called *virtual deadline scheduling* (VDS) [Zhang04], which varies the latest scope according to the feedback of histories.

Virtual deadline

Zhang and West [Zhang04] proposed a relaxed window constraint to gain in utilization. It allows task instances to be serviced after their deadlines, as long as it can guarantee a minimum fraction of service to a task in a fixed window. Its scheduling mecha-

nism lengthens the instances' deadlines, and the deadlines are modified according to the execution time. Moreover, like DWCS, virtual deadline scheduling requires specially that the tasks should have the unit size execution time and their period must be the multiple of the execution time.

VDS derives "virtual deadlines" for each instance from the corresponding (m,k)-window and request period. The task instance with the earliest virtual deadline is scheduled first. A task's virtual deadline with respect to real-time, t, is shown in equation (30). The start time of current request period at time t is denoted by $ts_i(t)$. In effect, this can be considered as the arrival time of the latest instance of task τ_i. Similarly, (m'_i, k'_i) represents dynamic current loss tolerance in the rest of window at time t (the *current (m,k)*-window constraint). This implies that window-constraints change dynamically, depending on whether or not an instance is serviced by its deadline.

$$Vd_i(t) = \frac{k'_i p_i}{m'_i} + ts_i(t) \qquad when \;\; m'_i > 0 \qquad (30)$$

The exact rules that control the dynamic adjustment of window-constraints will be described later on. At this point, it is worth outlining the intuition behind a task's virtual deadline. If at time t, τ_i' s current (m,k)-window constraint is (m'_i, k'_i), then $m_i - m'_i$ out of $k_i - k'_i$ job instances have been serviced in the current window. There are still m'_i job instances that need to be serviced in the next $k'_i p_i$ time units. If one instance of τ_i is serviced every interval of length $\frac{k'_i p_i}{m'_i}$, then m'_i job instances will be serviced in the current remaining window time, $k'_i p_i$. This assures proportional fairness guarantees to τ_i with respect to other window-constrained tasks. Additionally, the delay bound is minimized, by preventing at least m_i instances of τ_i being serviced in a single burst at the end of a given real-time window.

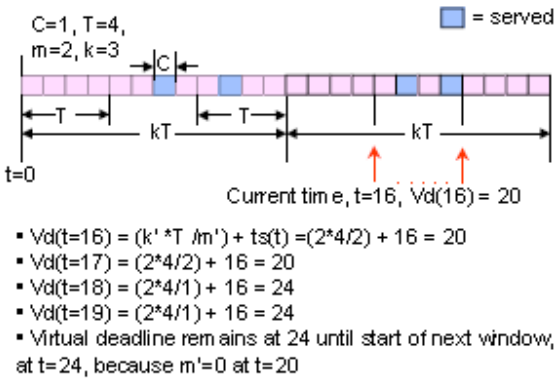


Fig. 34: Virtual Deadline Calculation

Fig. 34 gives an example of the virtual deadline calculation (we use the original figure in [Zhang04], where T denotes the period P). We can see that, if a task's current (m,k)-window constraint does not change within a request period, its virtual deadline will not change either. This example corresponds to the relaxed window-constrained model, where more than one instance can be served in one request period.

4 Conclusion of perspectives

In this chapter, we have illustrated the low resource utilization phenomenon in real-time scheduling area and have highlighted the theoretical reasons. These analyses showed the difficulty of the schedulability and challenges of the future work, and then we conducted three perspectives to make the real-time system achieve higher resource utilization:

1) Sub-optimal scheduling
2) Specifying the task parameters
3) Relaxation of real-time constraint

In fact, our work in the chapter 2 and the other works in [Quan00] [George00] just followed the first perspective. However, the first perspective research way needs the long computation time or needs long on-line verifying time, such that it losses the practical interests for the adaptive real-time system and networking resource allocation mechanism. DWCS [West04] was proposed according to the second perspective, and can be

considered as an interesting mechanism in terms of resource utilization. But its extreme requirements on the task model make it loss generality and loss suitability to act as a network packet scheduling scheme. According to the third direction of research, in the next chapter, we will try to relax the worst-case inter-distribution of instances and propose a new real-time constraint. This novel real-time constraint will construct the major contribution of this thesis, which defines a deadline on a group of instance rather than provide guarantee for each instance's deadline. In addition, this novel real-time constraint will be shown the significant high resource utilization in aspect of both real-time scheduling and active queue management.

Chapter 4

Proposition of Relaxed (m,k)-firm constraint

In this chapter, we will propose a relaxed version of (m,k)-firm constraint, under which the real-time system can achieve high utilization.

As mentioned, the practical advantage of (m,k)-firm constraint is to service more tasks with limited resource, and in other words, the research interest is to increase the utilization factor of a task set as much as possible. Unfortunately, until now there does not exist a non pre-emptive scheduling system which can get an interesting utilization gain for a non-concrete task set under (m,k)-firm constraint.

Consequently, we are motivated to proposed relaxed (m,k)-firm constraint according to the perspective research ways in the previous chapter. This relaxed (m,k)-firm constraint is well suited to the requirements of multimedia flows. In fact, the whole chapter 3 could be the motivation of our proposition of this R-(m,k)-firm constraint. This R-(m,k)-firm constraint construct our major contribution, and it will analyzed continuously in the chapter 4 and chapter 5.

In this chapter, we first introduce the general definition of R-(m,k)-firm constraint, which can be classed into *fixed window version* and *sliding window version*. In addition, we will give a sufficient condition to dimension the system resource requirement for a task set which can guarantee the fixed window version of R-(m,k)-firm constraint. This sufficient condition will be given under fixed priority scheduling; moreover, the instances of the same task are classed into mandatories and optionals according to *Sche_mkfirm* pattern which has already been introduced in section 4.1.2 of chapter 1.

This sufficient condition can be used to deal with the overload situation when it is not possible to meet all instance deadlines. Moreover, this sufficient condition is interesting to dimension the least resource requirement for the given task set, such that the system will drop all optional instances and can economise the resource to serve other work even the system has enough resource to meet all deadlines. As long as this sufficient condition is satisfied and the instances of every task are selectively dropped, the QoS of the system can be degraded to the R-(m,k)-firm constraint. Finally, the simulations are carried out to illustrate the successful effect on the scheduling with high utilization in comparison with HRT and original (m,k)-firm constraint.

1 R-(m,k)-firm scheme

All discussions in the previous chapter about the fatal low utilization motivate us to search a more flexible and suitable for multimedia real-time transmission. It is known that the real-time constraint is not a model without thinking of concrete application. Therefore, we propose a relaxed constraint of (m,k)-firm constraint according to the multimedia audio/video streams.

1.1 Definition of R-(m,k)-firm QoS constraint

Definition of R-(m,k)-firm constraint:
In a time interval [*s, t*] (with $t\text{-}s \geq l \geq 0$), a task submits *k* instances to a server. The server should finish the execution of at least *m* among them (*in order or not)* before time $t+\Delta$, where Δ is the maximum tolerable delay caused by the server for the group of *k* instances.

The following points can help to understand this definition. For instance, an R-(3,5)-firm constraint is shown in Fig. 35.

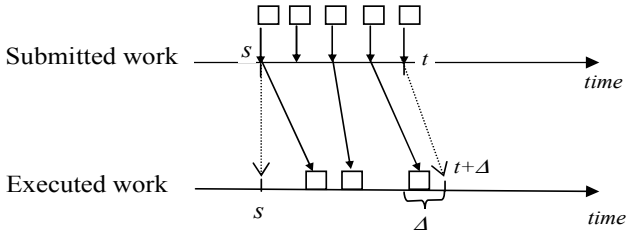


Fig. 35: **R-(3,5)-firm constraint**

1) In a certain time interval *[s,t]*, the stream can at most generate *k* instances. In other words, time point *t* is the end of generations of k^{th} packets. Obviously, for a periodic stream under R-(3,5)-firm, *[s,t] = 4p*, such as in Fig. 35.
2) A task τ_i can either be modelled by a periodic or sporadic source as (c_i, p_i, m_i, k_i) or by a (r_i,b_i)-bounded source under *(m_i,k_i)-firm* constraint but with Δ as the deadline of any group of *k* consecutive instances starting at time *s*.
3) The constraint is given for each task: every task can require its own constraint without considering others. The system should guarantee R-(m_i,k_i)-firm constraint for each task independently.
4) The real-time constraint includes two factors: **(m,k) factor** and **delay factor**. These two factors can be explained as a *fixed window* or *sliding window*. On the one hand, (m,k) factor applies to the sliding window, which means the *k* instances are counted from any one. On the other hand, it can also apply to the non-overlapped fixed windows, which means that the *k* instances are counted from 0^{th}, k^{th}, $2k^{th}$, etc (as defined in DWCS [West04]).
5) (m,k) factor of constraint: at least *m* among *k* consecutive instances should be executed within the delay constraint Δ.
6) Delay factor Δ: this factor assures that (m,k) factor must be realised before a maximum delay after the end of the release of the k^{th} instance starting from time *s*. For the sliding window requirement, it requires that from anytime one task generates *k* instances in a time interval *l*, and the system should assure the execution of at least *m* instances before $l+\Delta$. On the other hand, the fixed window just requires that after the release of a task, at least *m* instances are executed among the first *k* instances, as well as the *m* instances among $(k+1)^{th}$ to $2k^{th}$ instances, and so on.

1.2 R-(m,k)-firm for end-to-end QoS in network

After having understood the R-(m,k)-firm constraint, we give an example of its use in networks. As shown in Fig. 36, the source has a sending queue, and it sends the packets from the head of the queue through the network. The destination has also a corresponding receive queue, and adds the received packets to the end. Suppose in the time interval [*s, t*], *k* packets have been sent to the destination, the QoS provided by the network must assure the destination to receive at least *m* packets among the *k* for adding into its receive queue before $t+\Delta$. There are *k-m* discardable packets in any *k* consecutive ones.

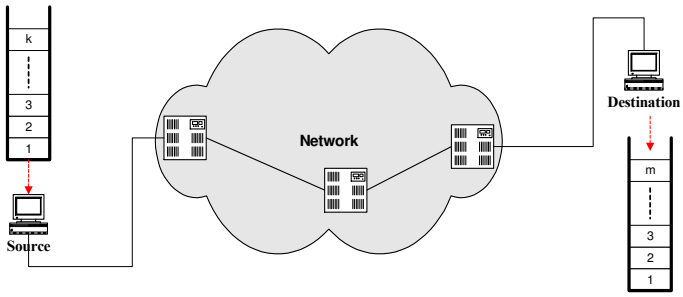


Fig. 36: Per Flow QoS

Actually, R-(m,k)-firm constraint replaces the conventional real-time constraint on the individual packet deadline with a delay factor on the end of a group packets.

1.3 Demonstration of R-(m,k)-firm advantages by virtual scheduling pattern

Fig. 37 shows the advantage of the R-(m,k)-firm constraint in contrast with the conventional per packet deadline constraint. Each block stands for a packet.

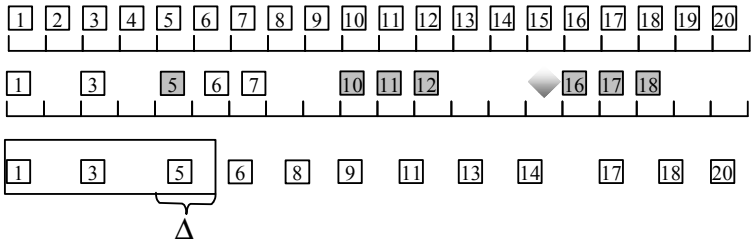


Fig. 37: **Example of a virtual scheduling pattern under (3,5)-firm and R-(3,5)-firm constraints**

The fist scenario (first line) shows a periodic packet stream that sends a packet at each beginning of the period (let P stand for the period time). The second scenario (2nd line) shows a scheduling under (3,5)-firm constraint. In the second scenario, the server is obliged to serve all grey blocks. Otherwise, the system falls into failure state, such as the 15^{th} block. Once the system falls into failure state (i.e. the (m,k)-firm constraint is violated), the server is still obliged to serve the next 3 packets (16^{th}, 17^{th}, 18^{th}) to restore system. Note that this obliged service of packets will cause high resource requirement (or low utilization) forming the so-called interference point [Quan00], [Jeffay91]. And the well-distributed interference points could reduce the resource requirement [Jeffay91]. However, the optimal distribution of those interference points is an NP-hard problem in strong sense, which makes it impossible to always reduce the over-provisioning problem. The third scenario shows a sequence under R-(3,5)-firm constraint, whose window size is configured according to R-(3,5)-firm constraint as $5p+\Delta$. Readers can verify by sliding the window, and will find that there are always at least 3 packets in the window no matter which beginning of period it slides to. Observe that although there are a lot of deadline misses, the sequence can still be accepted by R-(3,5)-firm.

1.4 R-(m,k)-firm is flexible and adaptive

It is obvious that the R-(m,k)-firm scheme is more flexible than the (m,k)-firm one. Although there are some deadline misses, it can be acceptable for a real-time multimedia communication such as VoIP, VoD, as well as some networked control systems where over-sampled data are transmitted by a network.

In previous section, we only showed a simple example scenario under R-(3,5)-firm and (3,5)-firm constraints, but it is not to say that all transmissions could tolerate the loss rate of 40%. After all, *m* and *k* of R-(m,k)-firm constraint can be configured as any natural number. Their values should be carefully chosen according to the specified real-time communication requirements of a given application. This point will not be further discussed here as it is beyond the scope of this work.

In fact satisfying R-(m,k)-firm constraint is not a trivial problem since the (m,k) factor and the delay factor are two antagonistic factors for a given server. On the one

hand, serving more instances (or packets) favours the (m,k) factor but leads to more delay. On the other hand, dropping more instances (or packets) may reduce the delay factor but risk to jeopardize the (m,k) factor.

1.5 Limitation of utilization gain for deterministic (m,k)-firm guarantee

The initial motivation of (m,k)-firm constraint is that the deadlines of at least *m* out of any *k* consecutive packets must be guaranteed. Similar to (m,k)-firm constraint, Dynamic Window-Constraint Scheduling (DWCS) [West04] can accept certain of deadline misses such that *x* is the maximum acceptable packet loss in a fixed given window *y* packets.

(m,k)-firm and DWCS constraints just take into account the amount of the deadline met or missed, whereas a stream is defined with several other parameters, such as the size of *transmission time, deadline* and *period* as well as *dropping rate*. This unilateralism of analysis results in poor utilization of resource, and this poor utilization factor causes the considerable over-provisioning of the system. Consequently, this over-provisioning will cause the (*m,k*)-firm or similar systems (e.g. DWCS) to lose their practical interest (advantages, or benefits, …). Those facts have been shown in the previous chapter 3 [Li03], [Mok01], [Quan00].

2 Discussing R-(m,k)-firm in MPEG transmission

For example, consider a MPEG transmission stream, its transmission order differs from that for file storage or display. At the receiver side, some processing must be done after the reception of a group of picture (GOP) composing of I, P and B frames (i.e., decode) in order to display the video on screen or store to the disk. This is just the essence of R-(m,k)-firm constraint. In other words, per-packet deadline is not necessary for multimedia transmission; since the previous packet must wait for the following packets even it can arrive at time.

In addition, the packets in a MPEG GOP are related. The *I* frames are the most important ones in comparison with the *P* and *B* frames. The B frames can be dropped according to a fixed (m,k)-pattern while still maintaining a QoS acceptable degradation [Koubâa05]. This differentiates the packets as mandatory and optional ones. Mandatory

packets must be serviced (be sent in time) while optional packets could be dropped. On the other hand, all current multimedia transmissions have their individual attributes. For instance, MP3, the audio compression of MPEG, is no-consecutive relationship among the packets. Therefore, all packets (instances) will be treated equivalently, and the relationship among the packets in the application level could be ignored.

Therefore, in this section, we will propose a sufficient condition for a task set (stream set), which can tolerate some loss of the optional instances (packets). Without taking into account a special stream, we assume the mandatory and optional instances are defined according to the Sched_mkfirm pattern [Ramanathan99]. Moreover, the effective resource utilization will be shown.

3 Sufficient condition for R-(m,k)-firm system

It is obvious that R-(m,k)-firm constraint is more flexible than the (m,k)-firm one. Although there are some deadline misses, it can be acceptable for a real-time multimedia communication such as VoIP, VoD, as well as some networked control systems where over-sampled data are transmitted by a network [LiAINA06].

In this section, a sufficient condition will be proposed in order to show that this novel real-time system can effectively economize the resources. With the sufficient condition, the system resource can be pre-dimensioned, such that a task set with fixed priority under R-(m,k)-firm constraint could be deterministically guaranteed.

3.1 Dimensioning method in traditional real-time deadline constraint

In [Tindell94] [Tindell95], the worst-case response time was studied for a given general task set under non-preemptive fixed-priority scheduling. The task set can be judged as schedulable or not, according to whether or not the worst-case response time is inferior to the deadline. The worst-case response time can be calculated by the following theorem:

Theorem 15: [Tindell94]: For a non-preemption task set, $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$, the priority is assigned according to the index, that is, the smaller the task index is, the higher priority it has. Then the worst-case response time r_i of task τ_i can be obtained as follows:

$$r_i = \max_{q=0\ldots Q}\{w_{i,q} + c_i - q \cdot p_i\} \quad \text{where}$$

$$w_{i,q} = q \cdot c_i + \sum_{j \in hp(i)} \left\lceil \frac{w_{i,q} + \gamma_{res}}{p_j} \right\rceil c_j + l_i$$

$$l_i = \max_{u \in lp(i)}\{c_u\}$$

Where l_i denotes the block factor caused by the tasks with lower priority. Q is the smallest value such that $w_{i,q} + c_i \le (Q+1)p_i$, γ_{res} is the resolution with which time is measured, and $hp(i)$ is the subset of tasks with higher priority than task τ_i. p_i is the period length, and p_j is the period length of a tack $\tau_i \in hp(i)$.

However, as mentioned, we are interested in finding the necessary resource for a given general task set, such that the execution time of every packet is an unknown argument. Nonetheless, every packet's size is known and according to formula $c_i = B_i / R$, the execution time is transformed as the ratio of packet size and resource capacity. Hence, we will study the worst-case arrival workload in order to find the resource requirement. The following result is obtained.

Theorem 16: For a non-preemption task set, $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$, the priority is assigned according to the index, that is, the smaller the task index is, the higher priority it has. All deadlines could be met if the resource capacity satisfies the following conditions:
For any $q \in (0 \ldots Q)$,

$$\left(w_{i,q} + B_i\right) / R \le \left(q+1\right) p_i \quad \text{where}$$

$$w_{i,q} = qB_i + \sum_{j \in hp(i)} \left\lceil \frac{w_{i,q} + \gamma_{res}}{p_j} \right\rceil B_j + l'_i$$

$$l'_i = \max_{u \in lp(i)}\{B_u\}$$

where R denotes the resource capacity of the server. The other parameters have the same meaning as in the theorem of [Tindell95].

Clearly, the resource requirement can be measured by means of finding the smallest value of resource (R) which satisfies all:

$$(W_{i,q} + B_i) / (q+1)p_i = R \quad (q=0...Q). \qquad (31)$$

Practically, this method calculates the sum of workload loaded by tasks with higher priority (e.g. in terms of bit) in the interval between the arrival time and the instance's deadline, as well as block factor from tasks with lower priority. A big enough resource capacity will reduce every execution time, such that all instances could be serviced before their deadlines. Since the upper theorem calculates the workload in terms of the worst-case for a general task set, all concrete task sets will be schedulable under HRT fixed priority scheduling.

In the same way, this method can be extended to calculate the resource requirement for a task set under R-(m,k)-firm constraint.

In case of overload, we drop the instances according to the pattern proposed in [Ramanathan99], named as Sched_mkfirm [Ramanathan99], as the following formula:

$$j = \left\lfloor \left\lceil \frac{j \times m_i}{k_i} \right\rceil \times \frac{k_i}{m_i} \right\rfloor \qquad j = 1...n \qquad (32)$$

In this scheduling process, the j_{th} instance of task τ_i ($i \in (1,n)$) will be loaded to the system if the formula (32) is satisfied, otherwise it will be dropped.

This Sched_mkfirm pattern ensures exactly that m_i out of any k_i instances will be loaded into the system. Moreover, from the first instance of task τ_i, R-(m,k)-firm constraint requires that m_i instances must be finished by the end of every k_i periods.

3.2 Periodic characters of Sched_mkfirm pattern:

Lemma 1: counting from the beginning of window, the q_{th} (where $q \in (1, m_i)$) instance submitted to the system occurs at time $\left\lfloor (q-1)\frac{k_i}{m_i} \right\rfloor p_i$.

Lemma 2: counting from the end of window, q_{th} (where $q \in (1, m_i)$) instance submitted to the system occurs at time $\left\lceil q\frac{k_i}{m_i} \right\rceil p_i$.

Lemma 3: for a task τ_j, the most workload in a time interval of L is not higher than:

$\left\lceil \frac{L}{p_j}\frac{m_j}{k_j} \right\rceil c_j$.

The proofs of Lemma 1, Lemma 2 and Lemma 3 are trivial and straightforward. Moreover, they have been partly presented in [Ramanathan99]. It is thus omitted.

3.3 Dimensioning for R-(m,k)-firm constraint

From now on, we will propose a sufficient condition of R-(m,k)-firm, under the fixed priority scheduling among the task, and the instances are defined as mandatories and optionals according to Sched_mkfirm pattern. In addition, we assume that the delay factor $\Delta_i = p_i$.

In fact, the main aim is to dimension the least resource requirement for a given task set with tolerance of R-(m,k)-firm constraint. With this sufficient condition, the system will drop all optional instances of the tasks, such that the resource can be economised to serve other work. Obviously, this sufficient condition can also be used to deal with the overload situation which means that it is not possible to meet all deadlines.

Theorem 17: ***sufficient condition***

For any periodic task set under fixed priority non-preemptive scheduling, the instances are dropped according to the *Sched_mkfirm* pattern. Then, if the following formulas are satisfied, any concrete task set generated from it will be schedulable under R-(m,k)-firm constraint.

$$w_{i,q} \le \left\lceil q\frac{k_i}{m_i} \right\rceil p_i * R \qquad (q = 1 \ldots m_i)$$

$$w_{i,q} = qB_i + \sum_{j \in hp(i)} \left\lceil \frac{\left\lceil q \frac{k_i}{m_i} \right\rceil P_i}{P_j} \frac{m_j}{k_j} \right\rceil B_j + l_i$$

$$l_i = \max_{u \in lp(i)} \{B_u\}$$

All parameters have the same meaning as in the Theorem 16.

Proof: we prove this theorem by contradiction. We will trace the violation of R-(m,k)-firm constraint backwards from the first occurrence of violation instance. Since R-(m,k)-firm constaint permets the task instance to be finished after its deadline (which equals to the period), then the following instances submitted form the same task could be suspended by its previous ones. Then, the following trace of workload calculation will be base on these two sub-cases by iteration util the beginning of the fixed window.

Assume that time t_d is the first time when the system falls into failure state. The failure is caused by the task τ_i which violates R-(m_i,k_i)-firm constraint at time t_d. According to Lemma 2, the last submitted instance of τ_i occurs at time $t_s = t_d - \left\lceil \frac{k_i}{m_i} \right\rceil p_i$, for which we should calculate the worst interference. To do this in [t_s, t_d], three cases could be found:

Case 1: the previous submitted instance of task τ_i is completed before time point of t_s. l_i stands for the blocking factor due to non-preemption. That is, a lower priority task's instance has occupied the server before t_s, and it has the highest priority and must be finished at first.

The worst interference caused by the tasks with higher priority (*hp(i)*) could be calculated according to Lemma 3, such that the sum of total workload from higher priority task can be denoted by:

$$\sum_{j \in hp(i)} \left\lceil \frac{\left\lceil \frac{k_i}{m_i} \right\rceil p_i}{p_j} \frac{m_j}{k_j} \right\rceil B_j \qquad (33)$$

then, the highest workload in [t_s, t_d] can be described as:

$$w_{i,1} = B_i + \sum_{j \in hp(i)} \left\lceil \frac{\left\lceil \frac{k_i}{m_i} \right\rceil p_i}{p_j} \frac{m_j}{k_j} \right\rceil B_j + l_i \qquad (34)$$

Because task τ_i violates R-(m_i,k_i)-firm constraint, in this case, $w_{i,1}$ must be higher than the length of [t_s, t_d]. This contradicts the theorem when $q=1$.

Case 2: the previous loaded instance of task τ_i is completed just at time t_s. In this case, there is no longer the block factor caused by lower priority tasks. Moreover, the worst interference caused by higher priority task is the same as that in the first case. Clearly, the sum of total workload in time interval [t_s, t_d] is inferior than that in the case 1. So, if the case 1 can be satisfied, case 2 will be well satisfied too.

Case 3: the previous instance of task τ_i is completed after time t_s. In this case, we derive the workload in the time interval, when the last two instances are loaded into the system, such as $t_s = t_d - \left\lceil 2\frac{k_i}{m_i} \right\rceil P_i$.

For the calculation, the three same cases will be found as in the above analysis. In addition, the last two instances of τ_i suffer a block factor only once in the time interval [t_s, t_d]. If task τ_i is not suspended by itself, the sum of workload is described as:

$$w_{i,2} = 2B_i + \sum_{j \in hp(i)} \left\lceil \frac{\left\lceil 2\frac{k_i}{m_i} \right\rceil p_i}{p_j} \frac{m_j}{k_j} \right\rceil B_j + l_i \qquad .(35)$$

This iteration analysis can stop until $q = m_i$; in other words, until the last m_{ith} submitted instance. Because the last $(m_i+1)_{th}$ loaded instance must have been finished before:

$$t_d - \left\lceil m_i \frac{k_i}{m_i} \right\rceil P_i = t_d - k_i p_i. \qquad (36)$$

Otherwise, if it is finished after $t_d - k_i p_i$, the failure will have already occurred in the previous window. This contradicts the assumption that t_d is the first time of failure state.

End of proof.

4 Simulations to show R-(m,k)-firm advantage in terms of resource utilization

In this section, the sufficient condition will be used to provision the resource for a task set under R-(m,k)-firm constraint in comparison with (m,k)-firm constraint and HRT constraint under non-preemptive fixed priority scheduling. With the results, HRT over-provisioning problem will be shown and one can find that R-(m,k)-firm is beneficial and requires less resource (or more resource efficient).

Consider one task set with three tasks, $\Gamma_1 = \{\tau_1, \tau_2, \tau_3\}$ and their parameters are shown in Table 6:

Γ_1	B_i (kb)	$p_i = \Delta_i$ (ms)	(m, k)
τ_1	1	2	(4, 5)
τ_2	4	5	(7, 8)
τ_3	5	8	(7, 9)

Table 6: parameters of task set Γ_1

We will calculate the workload to be executed in order to guarantee HRT, (m,k)-firm constraint and R-(m,k)-firm constraint, respectively.

In Fig. 38, their workload at time points before which to be executed are shown, and the resource requirement can be represented by the slope of the beeline which just exactly bounds all the workload.

According to sufficient condition theorem, the greatest ratio of workload and its time length to be executed can be used to provision the system resource under Fixed Priority scheduling for HRT and R-(m,k)-firm constraint, as shown in Table 7.

Γ_1	Workload (Mps)	Resource (Mps)
HRT	1.95	3
(m,k)-firm	1.59	3
R-(m,k)-firm	1.59	1.75

Table 7: Provisioning results for real-time constraints

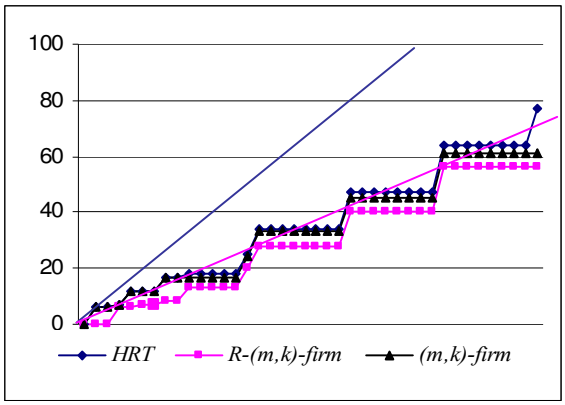


Fig. 38: The arrival workload to be executed

See from Table 7, it is obvious that task set Γ_1 has a workload of 1.95Mps, and in order to guarantee its HRT constraint, 3Mps bandwidth is required. (m,k)-firm constraint can tolerate some degradation, such that its workload reduces to 1.59Mps. Although (m,k)-firm constraint has a less workload, its resource requirement is the same as that in HRT. This fact verifies Property 3 in chapter 3. In contrast to (m,k)-firm, Γ_1 under R-(m,k)-firm has the same workload as that in (m,k)-firm constraint, note that it requires much less resource. Obviously, R-(m,k)-firm improves the performance.

In addition, the over-provisioning problem is improved as well by R-(m,k)-firm constraint unlike HRT and (m,k)-firm.

4.1 Efficiency of R-(m,k)-firm in admission control

Furthermore, in this example, the largest resource requirement is occurred in the case where an instance τ_1 is blocked by an instance of τ_3 because of non-preemption.

Assume now that another task $\tau_4 = \{c_4, p_4, m_4, k_4\} = \{2, 5, 4, 6\}$ arrives to the system, and the available system resource currently is just 3Mps. Intuitively, τ_4 cannot be

admitted. However, if we recalculate the resource requirement according to the sufficient condition for R-(m,k)-firm constraint, the provisioning is shown in Table 8.

Γ_2	Workload(Mps)	Resource (Mps)
HRT	2.325	3
(m,k)-firm	1.85	3
R-(m,k)-firm	1.85	2.4

Table 8: Provisioning of resource utilization

Observe that with the new task τ_4, a new task set $\Gamma_2 = \{\tau_1, \tau_2, \tau_3, \tau_4\}$, requires the same resource as Γ_1 for HRT and (m,k)-firm constraint. Theoretically, the largest resource is still required in the case where an instance τ_1 is blocked by an instance of τ_3. This fact demonstrates the problem proposed in Property 1 and Property 2 in Chapter 3. However, our R-(m,k)-firm constraint still requires less resource than those in HRT and (m,k)-firm.

Actually, if task τ_2 finishes its transmission, the new task set $\Gamma_3 = \{\tau_1, \tau_2, \tau_4\}$, will still require the same resource for HRT and (m,k)-firm constraints, but Γ_3 under R-(m,k)-firm constraint will require less resource than Γ_2. Hence, we can demonstrate that the resource required by R-(m,k)-firm constraint is monotonic with respect to the workload. This is an expected result as R-(m,k)-firm constraint can ameliorate the problems in Property 1 and 2 in Chapter 3.

Until now, we have shown the advantage of R-(m,k)-firm in comparison with HRT and (m,k)-firm, and showed the achievement of the motivations.

4.2 Scheduling trajectory

In order to show the schedulablity of a task set with above provisioning method, we will show the scheduling trajectory as given in Fig. 39, where clock tick is 100μs.

In Table 9, the packet size of tasks in the set Γ_2 is transformed to be the transmission time, which is done simply by the pakcet sizes divided by the dimensioned resouce requirement shown in Table 8 (this relationship has been introduced in section 1 of chapter 3).

Γ_2	c_i (ms)	$p_i = \Delta_i$ (ms)	(m, k)
τ_1	0.4	2	(4, 5)
τ_2	1.7	5	(7, 8)
τ_3	2.1	8	(7, 9)
τ_4	0.8	5	(4, 6)

Table 9: Transform the instance workload to execution time

Fig. 39 shows a scheduling process in 300ms which is divided into three lines because of space limit. In each part, from up to down, scheduling traces are the process of τ_1, τ_2, τ_3 and τ_4. Obviously, τ_1, τ_2 and τ_3 are executed without miss of traditional deadline, while in τ_4 scheduling process, there are misses of traditional deadline in every window, such that any deadline-oriented scheme cannot be satisfied in this case. However, R-(4,6)-firm constraint of τ_4 is well guaranteed. This result shows the graceful degradation provided by R-(m,k)-firm constraint.

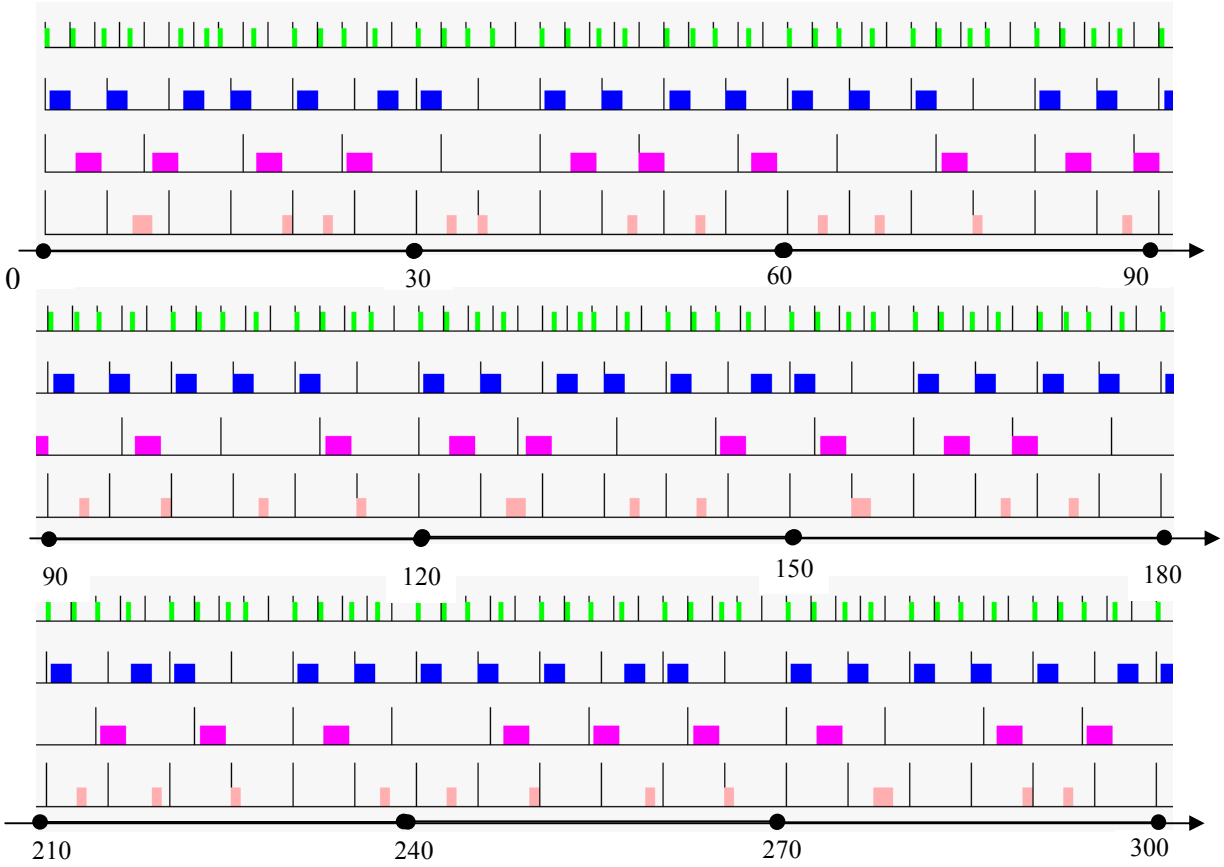


Fig. 39: Scheduling trajectory under non-preemptive fixed priority for R-(m,k)-firm

5 Conclusion

In order to ameliorate the low resource utilization for the network resource allocation, we proposed a relaxed version of (m,k)-firm constraint. R-(m,k)-firm constraint is first proposed in [LiAINA06], which aimed to give a new real-time scheme to provide graceful degradation. R-(m,k)-firm is a global and flexible scheme, while in this chapter, we focus on its advantage in aspect of scheduling for a periodic task set.

This R-(m,k)-firm real-time constraint enriches the quantification of QoS degradation for the real-time multimedia transmission, and it can adapt to different applications with fixed window version or sliding window version. Furthermore, an off-line sufficient condition has been proposed to evaluate the least resource requirement in the network for the deterministic R-(m,k)-firm guarantee. If the sufficient condition is satisfied, all generated concrete task sets will be schedulable by fixed priority allocation and the instances are dropped according to a Sched_mkfirm fixed pattern in condition of overload. The simulation and discussion have shown the effective resource provisioning in comparison with original (m,k)-firm constraint, and the low resource utilization problem is resolved. This scheme can also be applied in the next generation network QoS mechanism for the decision of admission control and resource allocation. The next chapter will discuss the application of R-(m,k)-firm and the implementation mechanism in the network.

Chapter 5

Guarantee R-(m,k)-firm Constraint of Network traffic by Queue Management

Together with the network scheme, it is always interesting to find the dimensioning method of bandwidth to cope with the congestion. Braden *et al.* distinguished between two classes of router algorithms related to congestion control, i.e., queue management and scheduling [Braden98]: "To a rough approximation queue management algorithms manage the length of packet queues by dropping packets when necessary or appropriate, while scheduling algorithms determine which packet to send next and are used primarily to manage the allocation of bandwidth among flows".

In real-time scheduling aspect, the analysis on R-(m,k)-firm constraint has been carried out in chapter 4. Therefore, in this chapter, we will focus on the use of R-(m,k)-firm constraint to control the network congestion by means of queue management.

Currently, the network congestion resolutions are not adaptive for multimedia transmissions since they randomly drop some packets without consideration of consecutive loss problem and the useless of retransmission. Conversely, we use R-(m,k)-firm constraint to deal with the congestion control by selectively dropping packets of (r,b)-bounded [LeBoutec02] multimedia flow using network calculus theorems. In addition, an active queue management mechanism is proposed, called Double Leaks Bucket (DLB), which performs dynamically a selective packet transmission and dropping mechanism according to the network load situation. The sufficient configuration condition of DLB is proposed to provide the deterministic guarantee for R-(m,k)-firm constraint.

1 Problem of real-time transmission

Nowadays, Internet supports more and more real-time and business-critical applications. We aim to ensure the real-time transmission of multimedia streams. The common characteristics of such kind of streams are the obligation of the timely packet delivery and the packet loss tolerance. Telephony and visio-conference over the Internet are typical examples. In this case, it is not necessary to retransmit lost packets since a retransmission may lead to unacceptable long delay. The fact that most of those applications may tolerate occasional packet losses until some extent (e.g. 1%-10% loss can be tolerated in packet audio applications [Bolot96]) is interesting, and it can be exploited for congestion management. However, a transmitted packet should be received within a bounded delay. During a congestion period, if a packet cannot be transmitted in time, it is often preferable to discard late packets rather than continue to transmit them. Moreover, discarding late packets is the most efficient and straightforward way to handle the congestion situation. In fact, the perceived quality for the user (e.g. a telephony conversation) depends not only on the loss rate but also on how the packet losses are distributed. Typically, consecutive packet losses must be avoided for a packetized voice flow [Bolot95].

Unfortunately, neither Intserv nor Diffserv proposed an efficient solution for providing them with transmission delay guarantees. In fact, as a flow often generates bursty traffic (case of most VBR applications and aggregated Diffserv class of flows), most of the existing solutions are based on the over-provisioning bandwidth reservation policy according to the peak rate of the flow in order to provide a bounded transmission delay to a flow (Intserv) or a class of flows (Diffserv). This results in a poor admission rate of real-time flows. Moreover, Diffserv does not provide any end-to-end real-time QoS guarantee [El-Gendy03]. Another problem we should deal with is how to drop the packets in providing real-time QoS in the network congestion in case of overload. This can occur when a path includes one or several routers which do not support Diffserv. Although occasional packet drops could be tolerated by many multimedia applications, long consecutive packet drops must be avoided since it can drastically decrease the QoS for applications such as audio/video diffusion.

2 Queue management introduction

In this section, we will introduce the main remarkable active queue management mechanisms. Queue management mechanisms can be divided into passive and active. Drop tail can be considered as a passive queue management which performs the best-effort for transmission service and drops the packet when the queue is filled. Active Queue Management is a technique of preventing congestion in packet-switched networks, and random early detection (RED) is currently a widely implemented active queue management.

2.1 Drop Tail

Drop tail can be considered as a passive queue management, which is the most widespread queue management policy (other solutions include dropping the first packet in the queue or dropping a random packet already stored in the queue). It drops the packets from the tail once the queue is filled. Hence, queue overflow will occur in case of network congestion. Additionally, this policy results in the problem of unacceptable consecutive packet losses, although it is simple to implement.

2.2 RED

Floyd and Jacobson [Floyd93] proposed Random Early Detection (RED) gateways back in 1993 to treat with the drawback of Drop Tail with an attempt to control the congestion level, limit queuing delays, and avoid buffer overflows. RED is an active queue management (AQM), which detects incipient congestion early and randomly drops the packets or marks them with an explicit congestion notification (ECN) bit, allowing the corresponding sources to reduce their transmission rates before queues in the network overflow and packets are dropped. RED can improve network performance in terms of delay, link utilization, packet loss rate and system fairness, and enhances routers to detect and notify end-systems of impending congestion earlier.

RED is a mechanism not designed to operate with any specific protocol. It needs five parameters: the maximum buffer size or queue limit (QL), the minimum (min_{th}) and maximum (max_{th}) thresholds of the "RED region", the maximum dropping probability (max_p), and the weight factor used to calculate the average queue size (w_q). RED uses the

average queue size (*avg*), instead of the current queue size, as a measure of incipient congestion, because it is rather intolerant to packet bursts. *avg* is calculated as an exponentially weighted moving average using the following formula:

$$avg_i = (1 - w_q) \times avg_{i-1} + w_q \times q \quad (37)$$

where the weight w_q is commonly set to 0.002 [Floyd01], and q is the instantaneous queue size. Two thresholds are set to manipulate the dropping process, if the average queue size does not exceed min_{th}, a RED router will not drop any packets, while all arrival packets will be dropped when the average queue size exceed the max_{th}. Early packet dropping starts when the *average* queue size exceeds min_{th} with the probability given as following formula and Fig. 40:

$$p_b = max_p \times (avg - min_{th})/(max_{th} - min_{th}) \quad (38)$$

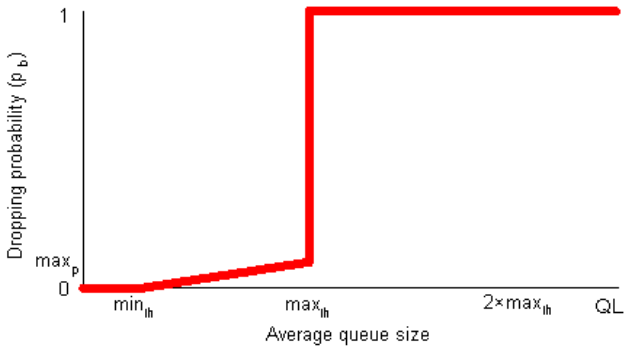


Fig. 40: RED dropping probability

Moreover, Floyd proposed a modification to the dropping algorithm (the "gentle option"), under which packets are dropped with a linearly increasing probability until *avg* exceeds $2 \times max_{th}$; after that all packets are dropped (Fig. 41).

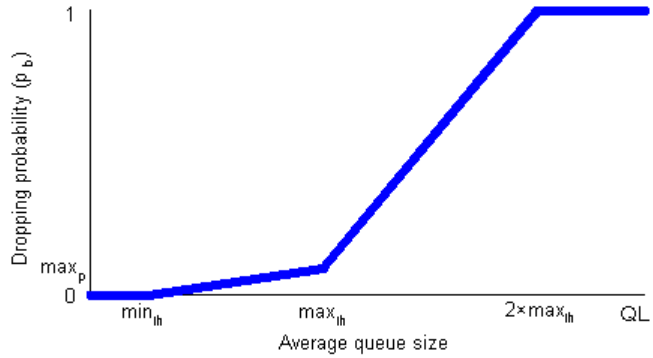


Fig. 41: RED with the "gentle option"

In [Jacobson], the authors pointed out that developed version such as: WRED (weighted RED), FRED (flow RED), RIO (in out bit), ARED (stabilized RED), etc, basing on the classical RED, all of which are basically drop-preference schemes which mean they are driven off the detailed sub-structure of a queue.

The RED and its developed versions have not the problem of the consecutive packet losses unlike the Drop Tail, since they drop the packets according to a dynamoic probability. However, they still haven't the guarantee on the non-consecutive drops. According to their probability based dropping policy, they can never provide the absolute guarantee of the flow.

2.3 BLUE mechanism

In contrast with RED, another active queue management BLUE is proposed to ameliorate the performance of RED in [WCFeng02]. The key idea behind BLUE is to perform queue management based directly on packet loss and link utilization rather than on the instantaneous or average queue lengths. This is in sharp contrast to all known active queue management schemes which use some form of queue occupancy in their congestion management. BLUE maintains a single probability, p_m, which it uses to mark (or drop) packets when they are enqueued. If the queue is continually dropping packets due to buffer overflow, BLUE increments p_m, thus increasing the rate at which it sends back congestion notification. Conversely, if the queue becomes empty or if the link is idle, BLUE decreases its marking probability. This effectively allows BLUE to "learn" the correct rate it needs to send back congestion notification.

- Upon packet loss (or $Q_{len} > L$) event:
  if ((now - last update) > *freeze time*)
    $p_m := p_m + \delta_1$
    last update := now
- Upon link idle event:
  if ((now - last update) > *freeze time*)
    $p_m := p_m - \delta_2$
    last update := now

Fig. 42: The BLUE algorithm

Fig. 42 shows the BLUE algorithm. Note that the figure also shows a variation to the algorithm in which the marking probability is updated when the queue length exceeds a certain value. This modification allows room to be left in the queue for transient bursts and allows the queue to control queueing delay when the size of the queue being used is large. Besides the marking probability, BLUE uses two other parameters which control how quickly the marking probability changes over time. The first is *freeze time*. This parameter determines the minimum time interval between two successive updates of p_m. This allows the changes in the marking probability to take effect before the value is updated again. While the experiments in this paper fix freeze time as a constant, this value should be randomized in order to avoid global synchronization [Floyed92]. The other parameters used, (δ_1 and δ_2), determine the amount by which pm is incremented when the queue overflows or is decremented when the link is idle. For the experiments in this paper, δ_1 is set significantly larger than δ_2. This is because link underutilization can occur when congestion management is either too conservative or too aggressive, but packet loss occurs only when congestion management is too conservative. By weighting heavily against packet loss, BLUE can quickly react to a substantial increase in traffic load. Note that there are myriad ways to manage p_m.

The first parameter, freeze time, should be set based on the effective round-trip times of connections multiplexed across the link in order to allow any changes in the marking probability to reflect back on to the sources before additional changes are made. For long-delay paths such as satellite links, freeze time should be increased to match the

longer round-trip times. The second set of parameters δ_1 and δ_2 are set to give the link the ability to effectively adapt to macroscopic changes in load across the link at the connection level. For links where extremely large changes in load occur only on the order of minutes, δ_1 and δ_2 should be set in conjunction with freeze time to allow pm to range from 0 to 1 on the order of minutes. This is in contrast to current queue length approaches where the marking and dropping probabilities range from 0 to 1 on the order of milliseconds even under constant load. Over typical links, using freeze time values between 10ms and 500ms and setting δ_1 and δ_2 so that they allow pm to range from 0 to 1 on the order of 5 to 30 seconds will allow the BLUE control algorithm to operate effectively. Note that while BLUE algorithm itself is extremely simple, it provides a significant performance improvement even when compared to a RED queue which has been reasonably configured.

2.4 Problem of current AQM mechanism for multimedia flows

In order to provide QoS guarantee, resource reservation or admission control should be done to the multimedia streams. Variable bit rate (VBR) and constant bit rate (CBR) are the main multimedia stream terms in networks. CBR stream can reduce its transmission rate after the reception of ECN.

While if some VBR streams all send with the highest rate (allocated for a more complex segment) as it happens, a busy period will occur. This busy period differs from transit congestion, since the busy period is much longer than transit congestion, and the busy period could happen more regularly than transit congestion.

As known, VBR's average bit rate is much smaller than the highest bit rate, such that this busy period could pass after a time interval, and the simultaneous occurrence of highest bit rate from some sources is a rare event. In this case, it is unreasonable to send ECN message to notify VBR streams, because the ECN will cause the retransmission of the whole segment, which will surely violate the real-time constraint such as TCP transmission.

In addition, if resource reservation or admission control is done according to the highest bit rate of all VBR streams, the link utilization will be very poor. So, it is better to

reserve less resource and drop some packets directly in the busy period within the stream's tolerance.

However, most of the Internet QoS mechanisms are designed to work with TCP/IP rather than RTP/UDP/IP. For instance, the well-known RED together with BLUE mechanism is implemented to run within a TCP session, such that the retransmission of the packets after the reception of ECN will violate the time delay requirement, and will be thought as useless. Moreover, as the network is already in congestion, the ECN packets together with the retransmission will make the congestion situation worse. Therefore, RTP will be employed and it is better to directly drop the packet in case of congestion under their tolerance extent rather than TCP retransmission and notification procedure.

In addition, almost the same loss rate can be achieved under the same available resource; in this case, packet loss distribution will be an important criterion for QoS of multimedia transmission because long consecutive dropping will cause the significant degradation of the multimedia stream. Therefore, a multimedia stream will be perceived with a better QoS if the loss packets are fairly distributed.

Therefore, the first contribution is that R-(m,k)-firm constraint can effectively control network congestion with a new active queue management mechanism, which selectively drops the packet within the loss tolerance extent, such that long consecutive packet loss can be deterministically avoided. The active queue management mechanism, called Double Leaks Bucket, selectively and fairly drop the packets in the stream for handling congestion during RTP/UDP/IP flow transmission Moreover, we will propose a sufficient configuration of the DLB which can deterministic R-(m,k)-firm constraint.

3 R-(m,k)-firm of sliding window constraint

Recall that R-(m,k)-firm constraint can be explained in terms of fixed window and sliding window. In MPEG multimedia stream, the packets are mutually dependent within a GOP (Group of Picture). Moreover, in GOP, the packets have consecutive relation and different importance, such that only optional packets can be dropped [Koubâa04]. Therefore, the fixed window version of R-(m,k)-firm constraint is suitable for compressed MPEG flows. The work in the previous chapter can resolve resource reservation or schedulability determination for the MPEG-alike streams.

On the other hand, all current multimedia transmissions have their individual attributes. For instance, MP3, the audio compression of MPEG, has no consecutive relationship among the packets. Therefore, all packets (instances) will be treated equally, and the relationship among the packets in the application level could be ignored. Intuitively, for this kind of stream, the packet transmission can be treated more freely and sliding window version R-(m,k)-firm constraint can be applied.

Even it could be a little redundant, we must recall the definition of R-(m,k)-firm constraint, which is digested for sliding window version.

Definition of the sliding window version of R-(m,k)-firm constraint:
In any time interval [*s, t*] (with $t\text{-}s \geq l \geq 0$), a task submits k units of workload (the amout of bits of packets) to a server. The server should finish the execution of at least m among them (*in order or not)* before time $t+\Delta$, where Δ is the maximum tolerable delay caused by the server for the group of k units.

The general definition of the R-(m,k)-firm constraint has been introduced in the previous chapter, which includes the fixed window version and the sliding window version. While, in this chapter, we will analyse the QoS guarantee for the network traffics in terms of sliding window and the following points should be minded:

1) In the time interval *[s, t]*, k quantity of workload arrives, and let $l = t\text{-}s$.
2) A task (or stream) τ_i focused in this chapter is a (r_i,b_i)-bounded source under *R-(m_i,k_i)-firm* constraint but with Δ as the deadline of any group of k consecutive instances starting at time s.
3) The real-time constraint includes both **(m,k) factor** and **delay factor**.
4) (m,k) factor of constraint: at least m among any k consecutive quantity of workload should be executed within the delay constraint Δ. In general case, (m,k) factor applies to the sliding window.
5) Delay factor Δ: this factor assures that (m,k) factor must be realised before a maximum delay after the release of the k_{th} workload starting from time s. For

the sliding window requirement, it requires that from no matter when one task generates k quantity of workload in time length l, the system assures the execution of at least m instances before $l+\Delta$.

6) Note that l is variable rather than a constant for a (r_i,b_i)-bounded stream since the workload arrives irregularly. As shown in Fig. 43 and Fig. 44, the length of l depends on how the k quantity of workload arrives, however, the delay factor Δ is counted from the end of $[s,t]$ with a fixed value.

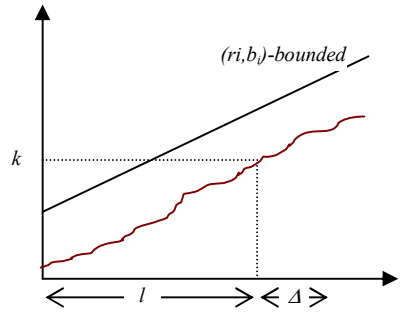


Fig. 43: *l* is variable for liquid workload

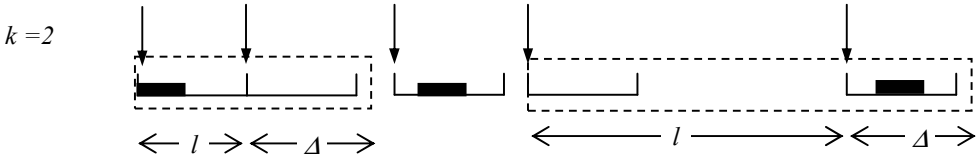


Fig. 44: *l* is variable for packet workload

7) The sliding manner of the window depends on the submission manner of the workload. In packet switching networks like Internet, if we stay at IP level, the information is transported in terms of packets. However, if we consider the physical layer the information is transmitted in terms of bits. Therefore, we will analyse the QoS for the workload in terms of the packet model and the liquid model. Another reason to interest in the liquid model (or fluid model) is that it gives a theoretic limit of the performance that one can hope to get. (This is quite similar to the relationship between GPS and it packet version WFQ). In the next section, we will also propose a mechanism in order to guarantee the sliding window version of R-(m,k)-firm constraint with the workload in terms of liquid and packet, separately.

8) s and t denote the 1_{st} and k_{th} workload arrival time, respectively. For the liquid model s and t can be anytime since the workload is in terms of liquid and arrives continuously, thus the window can *slide continuously*. However, for the packet model, it is assumed that one packet is submitted to the system without time cost. In other words, the difference between the packet workload and the liquid workload is that their submission times of the workload are discrete or continuous, respectively. Hence, s and t should be arrival time of packet for packet model.

4 DLB (Double-Leaks Bucket)

According to R-(m,k)-firm constraint of the sliding window, we developed a new mechanism double leaks bucket (DLB) [LiAINA06] from the traditional leaky bucket [Leboutec02], which can provide the deterministic guarantee of R-(m,k)-firm constraint and the fair distributed of dropped packets.

4.1 Liquid model of DLB mechanism

Firstly, to simplify the problem, we start the analysis with a liquid model, whose workload is in terms of 'water' that can be split infinitesimally, shown in Fig. 45. The network should guarantee the 'water' that travels through Serving Leak (SL), whilst the Discarding Leak (DL) controlled by one switch gives the capacity to throw out the water from the bucket. With the service guarantee for SL, R-(m,k)-firm constraint could be satisfied. The water going through the DL is discarded, and can be treated with whatever method never jeopardizing the network QoS.

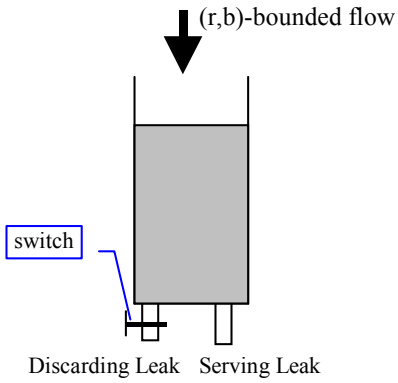


Fig. 45: Double-Leaks Bucket (Liquid Model)

Let C_1 and C_2 denote respectively the capacities of SL and DL. The control switch of DL works according to the quantity of the workload (represented by the height of the water in the bucket and denoted by q). This function is called as *double thresholds control function* (DTC function), as shown in Fig. 46, where 1 stands for the open state of the DL switch (or water gate) and 0 stands for the closed state.

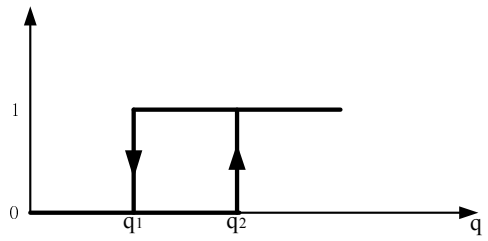


Fig. 46 : "Double Thresholds Control" function of the switch

During the increment of the water height, the switch is not open until the height grows to q_2, and once it is opened, it is not closed until the height goes down to q_1. When the switch is open, the discarding leak throws out the water from the bucket.

4.1.1 Service curve under (r,b)-bounded arrival stream.

We take the (r,b)-bounded arrival stream as a general source model. That is to say that the cumulative arrival curve is upper bounded by (r,b) [Leboutec02], where r is the average arrival rate, and b is the burst. Denoting by $F_i(t)$ the function of the arrival curve, the workload arrived at any interval $[s, t]$ is upper bounded by:

$$F_i(t) - F_i(s) \le b_i + r_i(t-s) \quad \forall 0 \le s \le t \qquad (39)$$

This (r,b)-bounded source generates k units of workload in a time length no smaller than $l = \left(\frac{k-b}{r}\right)^+$ [Leboutec02].

Assuming that the workload arrival bound is under the R-(m,k)-firm constraint. During the increment of the water height, the switch of DLB remains closed until the height increases to q_2. Once it is opened at q_2, it remains opened state until the height re-

duces to q_1. When the switch is opened (control function's value is 1), the Discarding Leak throws out the water from the bucket, so that the water height will be effectively reduced to assure delay factor. During this procedure, it is obvious that no more than C2/(C1+C2) quantity of water can pass through DL, such that this attribute will be used to guarantee the (m,k) factor.

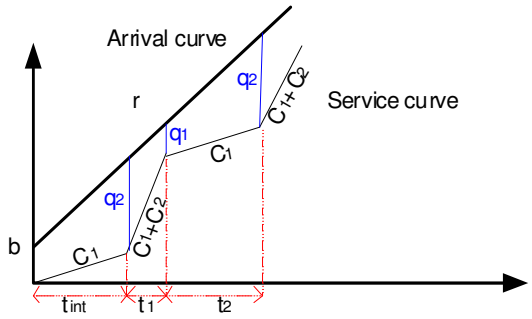


Fig. 47: Service curve evolution of DLB

Fig. 47 shows the evolution of the service curve under the critical cumulative arrival curve. Notice that SL is always on service unless the bucket is empty.

4.1.2 Sufficient condition for liquid model of DLB

After understanding the mechanism and its attribute to gurantee the (m,k) factor and the delay factor, we will propose the sufficient condition to configure DLB in order that deterministic guarantee can be achieved.

As shown in Fig. 47, R-(m,k)-firm constraint is proposed in condition of congestion or overload, such that it is obvious that $C1 < r$. So the height of water grows higher and higher in the interval t_{int}, as the DL switch is closed initially. Once the height reaches q_2, DL switch is open due to the double threshold control function (Fig. 46). In addition, we give $C1+C2>r$, so that the height of water goes lower and lower until the height decreases to q_1 (in the interval t_1). This procedure is iterated on until all arrival water has passed through either SL or DL.

Theorem 18: If the liquid model of DLB is configured under follow condition, then the R-(m,k)-firm constraint will be deterministically guaranteed.

Condition (1) : $C_1+C_2>r$; $\frac{C_1}{C_2} \geq \frac{m}{k-m} \Rightarrow C_1 \geq r\frac{m}{k}$;

Condition (2): if $b>q_2$, then $\max\left(\frac{b-q_1}{C_1+C_2}+\frac{q_1}{C_1}, \frac{q_2}{C_1}\right) < \Delta$

else $\max\left(\frac{q_2-q_1}{C_1+C_2}+\frac{q_1}{C_1}, \frac{q_2}{C_1}\right) < \Delta$

Proof:

Condition (1) ensures the (m,k) factor.

As the liquid model, water is an infinitesimal material (the unit could be atom, molecule, gram or ton, etc.). We configure C_1, and C_2 as $\frac{C_1}{C_2} \geq \frac{m}{k-m}$. Let $Q_{SL}(t)$ represent the throughput quantity through SL during]0,t], and $Q_{DL}(t)$ represent the throughput quantity through DL. It can be ensured that $\frac{Q_{SL}(t)}{Q_{DL}(t)} \geq \frac{C_1}{C_2} \geq \frac{m}{k-m}$, since SL is always on service unless the bucket is empty, while DL is only on service when the switch is open. Thus in any k units of water passed through SL and DL, there are at least *m* units passing through SL.

Condition (2) ensures the transmission delay factor of R-(m,k)-firm : when *k* units of water are affused into the DLB at time $t_a=\left(\frac{k-b}{r}\right)^+$. We calculate the maximum delay of a packet serviced by either SL or DL.

One case is when the switch *of* DL is open, and this case is further divided into two sub-cases: $b>q_2$ and $b<q_2$.

If $b>q_2$, the burst causes the *maximum* bucket load, then SL and DL service the workload together until the height decreases to q_1. Thereafter, SL service alone until the burst is finished. We know that results in the delay of the service for the burst is given by:

$$\frac{b-q_1}{C_1+C_2}+\frac{q_1}{C_1} \qquad (40)$$

In this case, if $b<q_2$, the height must decrease once the switch is open, so the maximum delay can be given by:

$$\frac{q_2-q_1}{C_1+C_2}+\frac{q_1}{C_1} \qquad (41)$$

Another case is that the height is so near to q_2, but the switch is still closed. SL will service alone the workload until empty the bucket. This case results in the delay of service, given by:

$$\frac{q_2}{C_1} \qquad (42)$$

It must assure that in any case the maximum delay is no more than △, so that we give the condition (2).

End of proof.

In a concrete system, the source has its attributes of arrival curve upper bounded by (*r,b*) and R-(m,k)-firm requirement, so in the analysis, these parameters are regarded as the known parameters. Moreover, we can get the available bandwidth, so C_1 should be configured under the available bandwidth. Based on these known parameters we should constitute one DLB (configure C_2, q_1, q_2) to cope with this flow for providing the deterministic R-(m,k)-firm guarantee.

4.1.3 Numeric application of liquid DLB

To show the numerical application, audio-CBR streams' parameters are considered. For example, given one flow, it generates 2Mbps of average arrival rate with 6kbit of burst. Such that the flow is bounded by (*r,b*)=(2Mbps, 6kbit). As an example, we assume that such a stream is under R-(3,5)-firm constraint and with △=20ms as the per-flow deadline.

Assuming that at admission control, only 1.6Mbps bandwidth is available. Obviously, it is not possible to guarantee the deadline of all the packets, as congestions are unavoidable. We set $C_1+C_2 = 1.25r = 2.5$Mbps, so that the queue length can be effectively reduced in case of congestion. Then, DLB is implemented and configured as C_1=1.5Mbit/s; and C_2=1Mbit/s according to $C_1 = \frac{m}{k-m} C_2$; moreover, q_1 and q_2 are configured as that q_1=6kbit, q_2=12kbit. With this configured DLB, the maximum delay can be

calculated by formula (41) as 8ms. So we can guarantee $t_{delay} \leq 8ms \leq \triangle = 20ms$. While, for guaranteeing all packets under delay of 20ms, it requires $r+b/t_{delay}$= 2.3Mbit/s of bandwidth [Leboutec02]. With DLB, it guarantees R-(m,k)-firm constraint in providing 8ms delay, but it requires only 1.5Mbit/s of bandwidth. Intuitively, in this example, DLB can still work under smaller bandwidth, so DLB is robust under smaller bandwidth.

4.2 Packet model of Double-Leaks Bucket

In the packet switching network, the information is encapsulated in packets, and the non-preemption is widely employed. So we now develop the DLB model according to the granularity of the packets and under the discrete time. Afterwards, the given R-(m,k)-firm is oriented the number of packets. Fig. 48 depicts the mechanism. Two new parts are added, named *temporary vessel buckets* (TVB) for DL and SL, respectively. They take an entire packet from the DLB after the service of the current packet in themselves. TVB of the DL can only get the next during the switch is still in open state.

The switch here is also controlled by the DTC (double thresholds control) function. Afterwards, for the packet model, the values of q_1 and q_2 are no longer the quantity of water, but the number of packets; to represent the quantity of workload we use q_1S or q_2S, where S denotes the size of the packet (in unit of bit or byte).

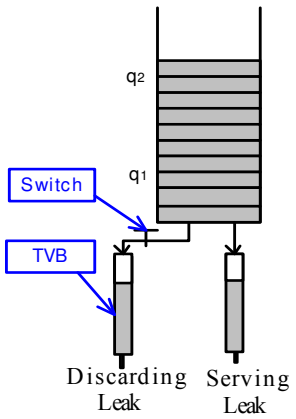


Fig. 48: Packet model of DLB

4.3 Sufficient condition of DLB in packet model

Due to the packet granularity, the sufficient condition will be more complex than that of the DLB liquid model, which is given in the following theorem.

Theorem 19: The sufficient condition for guaranteeing R-(m,k)-firm constraint of the packet model can be given as follows:

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

(2) If $b<q_2$

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

else

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

Proof:

Condition (1) ensures the (m,k) factor.

As mentioned in the liquid model, it should be provided that $\frac{C_1}{C_2} \geq \frac{m}{k-m}$ as in the liquid model. Furthermore, we must take into account the granularity of the packet. In case that TVB of DL has just taken one packet when the bucket height is q_1+1, then the switch will be closed. Therefore, the service time of TVB of DL will continue $\frac{S}{C_2}$, and SL will be on service at least $\frac{q_1}{C_1}S$. The service process should be that SL is always on service during DL does, then $\frac{q_1}{C_1} > \frac{1}{C_2}$. This deduces that $q_1 \geq \frac{C_1}{C_2} \geq \frac{m}{k-m}$. We set q_1 as the minimum value such as: $q_1 = \left\lceil \frac{m}{k-m} \right\rceil$, and it is clear that $q_2 > q_1$. So we can choose one suitable value for q_2, which is neither too much high nor causes the switch rotate too frequently.

Condition (2) is derived with the same strategy as that for liquid model to guarantee delay factor.

End of proof.

4.4 Numeric application of packet DLB model

Let's consider the same concrete example as in the liquid model, the packet size is S=6kbit, then we can configure this DLB as $C_1=1.44Mbit/s$, $C_2=0.96Mbit/s$, $q_1=2$, $q_2=5$. With this configured DLB, R-(m,k)-firm is guaranteed $t_{delay} \le \frac{q_2-1}{C_1} S = 10\text{ms} \le \Delta = 20\text{ms}$.

5 Performance comparison

In this section, DLB will be demonstrated its effective performance by comparison with other Active Queue Management (AQM) in terms of selective drop. In addition, DLB will also be shown the widely perspective implementation.

5.1 Selective drop of DLB

As shown above, DLB can provide deterministic *R-(m,k)-firm* guarantee to *(r,b)-bounded* tasks (or flows). To show the performance of DLB faced to more general flows, we simulate DLB as well as Drop Tail and RED in case of a Poisson flow with the rate $\lambda=1$ packet per time unit, with equal packet size. We assumed that the source generating this flow can tolerate up to 33% of loss. To simulate their behaviours during overload period, the server capacity is set to be less than the average arrival rate of the source. For equal comparison, the same queue size and server capacity are given to DLB, DROP TAIL and RED (parameters are shown in Fig. 49).

DLB pattern (q_1=3, q_2 = 6, queue size = 9, C_1 = 0.8, C_2 = 0.4)
011011011011011011011011011011011011011111111111111111110110110110110110
110110111111111011011011011011011011011011011111111111111111111111111111
011011011011011011011011011011011011011011
Drop Tail pattern (queue size = 9, C = 0.8)
101101111100001100011111111111111111111111111011101010010111111111101111
110100111110101001010000111110011111111111111111111111111111111111111101
110110011010001111111011111011111111110
RED pattern (w_q = 0.2, max_p = 0.34, min_{th} = 3, max_{th} = 6, queue size = 9, C = 0.8)
111010010000111111111111111111111111111111111010111111111100110011111101111
111111111111111111111111111111011001011101111101111111111111111111101000011
110001100010101111111111111111111101010

Fig. 49: Loss Packet patterns of DLB, Drop Tail and RED

Fig. 49 also shows a sample sequence of the serviced and dropped packets, respectively denoted by 1 and 0. As seen, DLB avoids consecutive losses, while Drop Tail and RED do not. Additionally, the performances of DLB, RED and Drop Tail are demonstrated in Table 10 in terms of queue length, queuing delay, loss rate, as well as the maximum and average consecutive loss.

	DLB	RED	DROP TAIL
Mean queue length	4.7	4.3	9.0
Mean delay (s)	4.8	4.5	8.7
Drop rate	22%	22%	20%
Maximum consecutive loss	1	4	4
Average consecutive loss	1	1.64	1.54

Table 10: Comparison of DLB, RED and Drop Tail.

See from the Table 10, since the server is set with the same capacity for DLB, RED and Drop Tail, almost the same amount of packets can be transmitted in the simulation time and the loss rates are also almost the same. However, DLB and RED perform better than Drop Tail in terms of mean queue length and mean delay. In addition, DLB performs much better than RED and Drop Tail in terms of maximum and average consecutive loss. This fact is the most important advantage of DLB in comparison with RED that it can deterministically avoid the long consecutive drops for the real-time multimedia streams.

5.2 Discussion among RED, BLUE and DLB

As seen, DLB is very simple without many parameters and can be easy configured, such that it is possible that per-flow can be treated separately to satisfy the individual requirement.

Moreover, DLB drops the packets with a deterministic rate (selective dropping) rather than probability as RED and BLUE. This selective drop can guarantee the avoidance of consecutive dropping. For multimedia flow, it is better to distribute the losses as fairly as possible, since long consecutive dropping could result in the information loss

and the loss is difficult to be compensated at the destination even under the small dropping rate.

However, RED and BLUE drop the packet randomly. For example, assume that even RED and BLUE drop the packets with the probability of 1/3, then the first three arrived packets may be dropped as 110, and the second three may be dropped as 011. Thus, the six packets are dropped as 110011, which is clearly worse than the treatment of DLB such as 110110. Moreover, no matter long consecutive dropping can occur for RED and BLUE according to the probable dropping.

Above all, DLB performs well in active queue management domain, but its main advantage is that it can provide a guarantee for the real-time multimedia transmissions, which is not concerned by either RED or BLUE. In fact, traditional queue management is unconcerned with real-time constraint. However, DLB is just proposed for multimedia real time transmission.

5.3 Implementation of DLB in Diffserv

Fig. 50 shows the Diffserv QoS management mechanism, which can provide a simple and coarse method for classifying services for various applications. In Diffserv, EF and AF (Assured Forwarding) are two standards of service levels for per hop behaviours (PHBs) [Davie02]. Traditionally, Real-time flows (called microflows in [Davie02]) will be classified to EF queue in order to get low delay and jitter. To give a delay guarantee to the aggregate, the node administrator has to reserve a sufficient portion of the whole bandwidth. Although a lot of work has been done on the mechanisms to share EF traffic with the others, however, how to assign this resource to the EF aggregate to guarantee the delay requirements of the individual microflows has not been addressed. Obviously, the effective resource provisioning method can enable the Diffserv to serve more real-time flows at the same time.

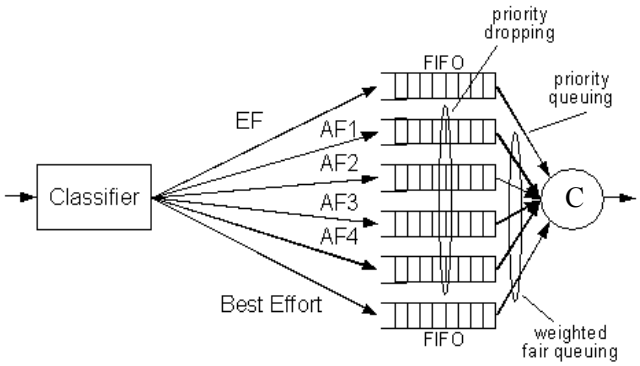


Fig. 50: An example of DiffServ node [Davie02]

DLB can provide efficient resource allocation with selective dropping of the packets such that we can implement DLB between the classifier and the EF queue, as shown in Fig. 51.

If we have *n* real-time microflows forming an EF aggregate, each with their different time constraints as described in Section 3.1, we have the classic scheduling problem of a set of n flows $\Gamma=(\tau_1, \ldots \tau_n)$ *with* $\tau_i = (c_i, p_i, d_i, m_i, k_i)$, as shown in Fig. 51

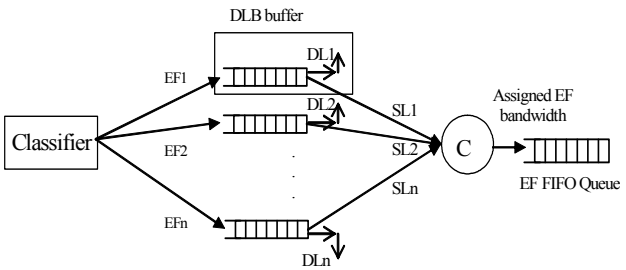


Fig. 51: Implementation of DLB model

With an assigned EF bandwidth, our aim is to accept the maximum micro-flows at the connexion admission control. Using DLB on each microflow can then achieve more efficient bandwidth utilization than using hard real-time scheduling. The bandwidth assigned to EF is redistributed to each microflow using WFQ (Weighted Fair Queuing). This approach can find many applications in the current networks to provide the adaptive controllable QoS according to the R-(m,k)-firm constraint.

6 Comparison between R-(m,k)-firm and other real-time constraint relaxation strategies

The initial motivation of (m,k)-firm constraint is that the deadlines of at least *m* out of any *k* consecutive packets must be guaranteed. Similar to (m,k)-firm constraint, Dynamic Window-Constraint Scheduling (DWCS) [West04] can accept a certain deadline misses such that *x* is the maximum acceptable packet loss in a fixed given window *y* packets.

(*m,k*)-firm and DWCS constraints just take into account the quantity of the deadline met or miss, whereas a stream is defined with several other parameters, such as *transmission time, period and dropping rate*. This unilateralism of analysis results in bad utilization of resource, and this poor utilization factor causes the considerable over-provisioning of the system. Again, this over-provisioning causes the (*m,k*)-firm or its similar systems (e.g. DWCS) to loss their practical interest.

6.1 Comparison by simulation

In this subsection, we give a scenario to show that R-(m,k)-firm constraint can significantly gain the utilization factor compared with (m,k)-firm constraint and DWCS constraint.

The scenario consists in four tasks, each of which has R-(m_i,k_i)-firm constraint. We will simulate for a strict delay factor (i.e., $\Delta_i = p_i$) to show the high gain of utilization.

The task set is scheduled under non pre-emptive fixed priority scheduling, and the priority is indicated by the index. Then, the schedule trace is shown in Fig. 52.

Notice that R-(m,k)-firm constraints are well guaranteed for either sliding window or no-overlapping window with delay $\Delta_i = p_i$. Such that, for any task, in any window of size $k_i p_i$, there are at least m_i instances executed. Dashed window is marked in Fig. 52, which can help the readers to verify this.

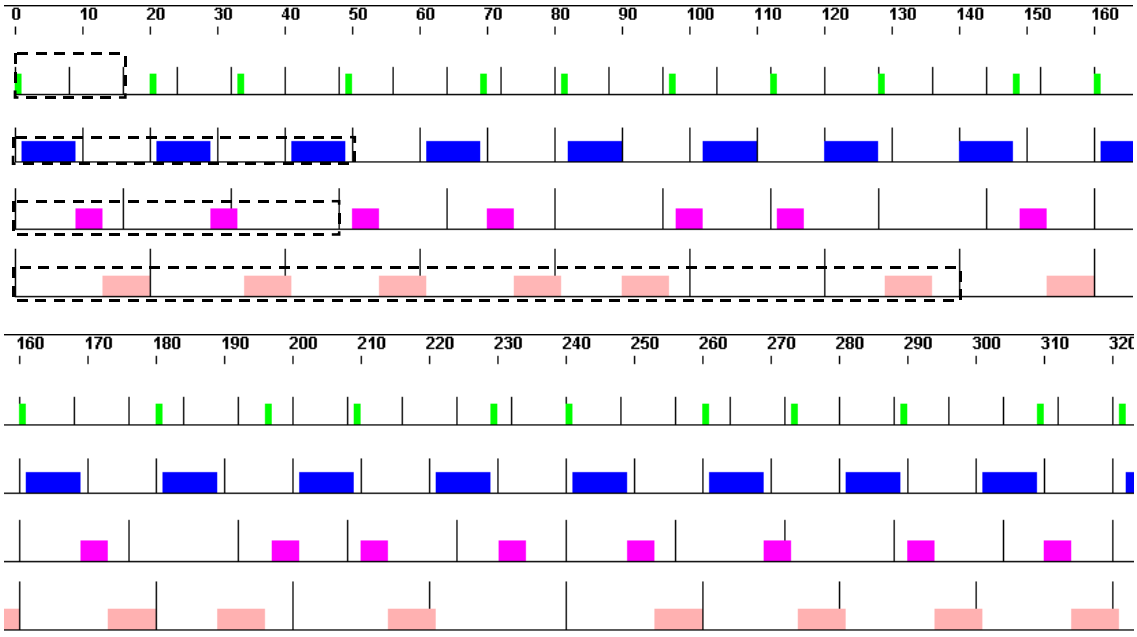
It is also easy to discover that a lot of missed deadlines exist during the scheduling trace for τ_3 and τ_4; caused by this, all deadline schemes (DWCS and (m,k)-firm constraint) will be violated. However, R-(m,k)-firm constraint replaced the per-instance

deadline by the delay of group of instances, such that a high utilization is gained (88%). Note that more gain of utilization can be achieved with the no-zero delay factor.

	Period = Δ_i (ms)	Execution time (ms)	R-(m,k)-firm
τ_1	8	1	(1,2)
τ_2	10	8	(2,4)
τ_3	16	4	(2,3)
τ_4	20	7	(5,7)
Utilization		88%	

Table 11: R-(m,k)-firm simulation scenario

Moreover, this scenario dealt with the periodic task set, which makes it different from the numerical applications given in section 4. The numerical applications given in section 4 are applications of DLB for the tasks upper bounded by (r,b), while periodic task has less burst. Again, it should be noticed that R-(m,k)-firm constraint abandons the per-deadline restriction in order to achieve the high resource utilization, which demonstrates the relaxation and flexibility of R-(m,k)-firm constraint.



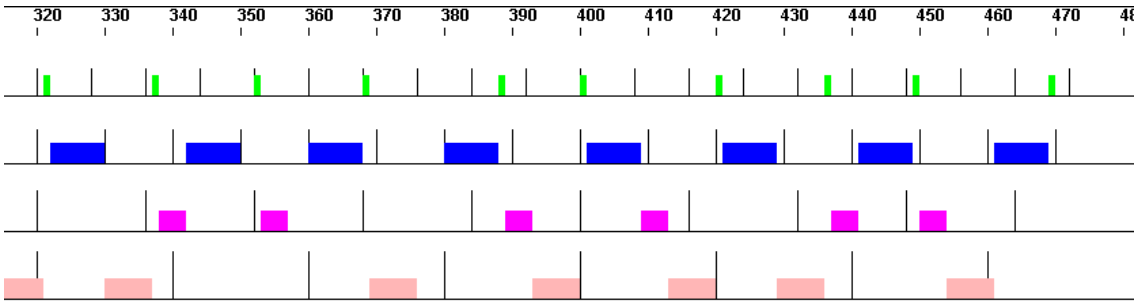


Fig. 52: Scheduling trace of task set in Table 11

7 Compare R-(m,k)-firm with other relaxed constraint models

Recall that R-(m,k)-firm scheme is a strategy to relax the too tight (and unnecessary) hard real-time constraint for the applications which can tolerate both some deadline misses and some packet losses. R-(m,k)-firm constraint, consisting of fixed-window version and sliding-window version, can be employed to describe and evaluate the requirements of various real-time application.

Similarly, Pinwheel model [Hotel89], frame-based model [Liberto99] and Virtual window constrained model [Zhang04] can also be considered as other relaxation strategies. In what follows we will discuss on those models and show that R-(m,k)-firm is a general model which can include the other ones.

First, we will recall the definitions of Pinwheel model, frame-based model and Virtual window constrained mode, even they have been introduced in chapter 3. During the digest of the definitions, the advantage, reasonability and the adaptability of R-(m,k)-firm will be naturally demonstrated.

The frame-based model defines a set of tasks, which is to execute within each frame (time window) and is to complete before the end of frame. The problem is to schedule the task set in a single frame with deadline D.

Frame-based model in [Liberto99] assumes that all task instances are stored in a single queue at the beginning of the frame (i.e. released at the same time at the beginning of the time window). Note that frame-based model removes the deadlines of instances to provide a scheduling in the same frame size for all tasks. However, for current diverse applications, tasks demand different frame size, which makes the frame-based model little interest for practical use in QoS control networks.

- **Fixed window version of R-(m,k)-firm includes frame base model.**

Note that frame-based model can be considered as one special case of R-(m,k)-firm, and fixed window version of R-(m,k)-firm can be transformed to frame-based model with some specification conditions. Given a task set under R-(m,k)-firm constraint, the transformation conditions are: all instances in a fixed window are released at the beginning of the window, the delay factors Δ of every task equals to there period ($\Delta_i = p_i$) and $(k_i-1)p_i + \Delta_i = k_i p_i$ of each task is set with the same value as the frame size D. With these special conditions, the *fixed window version of the R-(m,k)-firm constraint* is transformed to frame-base model.

- **Sliding window version of R-(m,k)-firm includes Pinwheel model.**

The Pinwheel model reserves different quantity of time in different frame size for each task rather than frame-based model. The generalized pinwheel scheduling problem is an offline scheduling for satellite-based communication as follows: Given a multiset $\{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$ of ordered pairs of positive integers, determine an infinite sequence over the symbols $\{1, 2, 3, \ldots, n\}$ such that, for each i, $1 \le i \le n$, the Pinwheel constraint requires that any subsequence of b_i consecutive integers contains at least a_i times i . Differing from the frame-based model, Pinwheel guarantees the execution time in a sliding frame.

By consequence, Pinwheel model can also be transformed from the *sliding window version of R-(m,k)-firm constraint*. Given a task set under R-(m,k)-firm constraint, the following specification conditions can transform R-(m,k)-firm to Pinwheel: $k_i p_i = b_i$, $c_i = 1$, $m_i = a_i$, $\Delta_i = p_i$.

As mentioned, frame-based model constrains all tasks in a unique frame; on the other hand, Pinwheel is designed for satellite-based communication model which only concerns unit size execution time. Therefore, they cannot service current diverse multimedia applications, while R-(m,k)-firm are more general and flexible to serve the multimedia applications (already discussed) including frame-based and Pinwheel application areas.

- **Fixed window constraint of R-(m,k)-firm constraint includes virtual deadline constraint.**

Similar to R-(m,k)-firm constraint, Zhang and West [Zhang04] proposed a relaxed window constraint to gain utilization, virtual deadline constraint. It allows task instances to be served after their deadlines, as long as it can guarantee a minimum fraction of service to a task in a fixed window. Obviously, this model can be regarded as the *fixed window version of R-(m,k)-firm constraint* when delay factor $\Delta_i=p_i$. Its scheduling algorithm, virtual deadline scheduling, applies the P-faire scope strategy to modify instances' deadline according to the execution time. Moreover, virtual deadline scheduling mechanism requires that all tasks have the unit size execution time and the periods must be the multiple of the execution time. In converse, our proposition of dynamic scheduling mechanism, DLB for R-(m,k)-firm constraint, has no special requirement on the task set model, and R-(m,k)-firm constraint can be deterministically guaranteed by DLB with high utilization as well (under the sufficient conditions in Theorem 18 and Theorem 19).

In one word, R-(m,k)-firm can also be considered as a modified real-time constraint like above models to define a more general scheme for general task set.

8 Conclusion

Current networks often fall into congestion caused by the overload and the overprovisioning is not always possible, such that graceful degradation of Quality of Service in networks is necessary for efficiently supporting the loss tolerant real-time applications such as VoIP, VoD, etc. Selective discarding of packets according to the (m,k)-firm model during overload periods is the key issue of our approach.

In this chapter, two main contributions can be found. First, the sliding window version of Relaxed (m,k)-firm constraint is employed in this chapter. Under this R-(m,k)-firm constraint, long consecutive loss of packets can be avoided, so that it is suitable for divers multimedia applications. This novel real-time constraint orients to the more general (b,r)-bounded streams, including periodic and sporadic ones. Another contribution is that one dynamic mechanism, called Double-Leaks Bucket, has been proposed to deterministically guarantee the R-(m,k)-firm constraint.

The comparison with other schemes showed the generality of R-(m,k)-firm constraint in contrast with DWCS, Frame-based Model, Pinwheel Model, and Virtual Deadline Scheduling model. Furthermore, Simulation scenarios showed how R-(m,k)-firm

constraint increases the resource utilization by replacing per packet deadline with the delay on the group of packets. Its implementation in Diffserv context is straightforward.

Chapter 6

Conclusion and Future Work

1 Overall conclusion

This thesis addressed mechanisms, policies and schemes of providing graceful degradation of real-time quality of service in distributed, adaptive real-time system and network. Since adaptive real-time system and real-time multimedia streams are loss-tolerant or deadline misses tolerant applications, we used (m,k)-firm scheme to evaluate and measure the graceful degradation of quality of service, and to selectively drop a certain amount of packets in case of overload. This scheme can fairly distribute the lost instances in the task execution sequence and avoid long consecutive losses. The goal of our work is to seek the effective resource provisioning method to deterministically guarantee the (m,k)-firm constraint for a real-time task set accessing a system with a limited resource. This work meet the challenges of admission control mechanism for QoS guarantee in networks and the processor capacity provisioning in networked control system.

To begin with, we have introduced the general definitions of (m,k)-firm constraint and the analytical model, as well as the state of the art of the (m,k)-firm scheduling methods with their sufficient conditions of schedulability.

In this thesis, the research contributions have been illustrated by several steps and we have deduced some theoretical results. The chapter 2 focused on the dynamic (m,k)-firm scheduling algorithm, DBP, because it is flexible and suitable for adaptive real-time applications. A sufficient condition of off-line schedulability determination and an on-

line schedulability verification time interval are given for NP-DBP-EDF scheduling method. And then the efficiency of this sufficient condition is analysis in an in-vehicle embedded control system; however, the resource utilization maybe very low in case of deterministic guarantee. Even be worse, the resource requirement for a task set under (m,k)-firm constraint cannot be deterministically economised in comparison with HRT [LiTII06].

In consequence, we analyzed the reasons of low utilization in real-time scheduling area including but not limited in (m,k)-firm constrained task set. After the deep analysis and result summarization, we found that the real-time schedulability is mainly affected by the inter-distribution of tasks' instances for concrete task set (release time for periodic task set [Jeffay91]). Moreover, the general scheduling problem of a real-time task set under HRT and (m,k)-firm constraint has been proven as NP-hard in strong sense and resource utilization can be arbitrary low [Garey77] [George00] [Jeffay91] [Mok01] [Quan00] [LiDEA03].

According to the various past works [West04] [Zhang04] [Quan00], we have deduced three perspective research ways to improve the resource utilization for real-time system: (1) heuristic method for sub-optimal result (2) specifying task set parameters (3) relaxation of real-time constraint. The first way losses the practical interests since it costs a large computation time. The second way must have extreme restrictions on task model (e.g., DWCS model [West04]) and makes it not suited for the practical networks scheduling. Therefore, according to the third research perspective way, we gave our uppermost contribution of this thesis, called relaxed (m,k)-firm (R-(m,k)-firm) constraint.

R-(m,k)-firm has been proven as an effective and efficient graceful QoS degradation scheme in terms of resource utilization, and is very flexible and suitable for multimedia audio/video streams.

Because the scheduling and queue management are the two classes of router algorithms to provide QoS in case of network overload (congestion), we focused on these two areas for providing R-(m,k)-firm guarantee.

In real-time scheduling aspect, we proposed an offline sufficient condition to deterministically guarantee R-(m,k)-firm constraint for a task set under fixed priority scheduling method [LiETFA06].

In queue management aspect, we proposed an active queue management mechanism to dynamically drop the packets in case of overload (network congestion), while still deterministically guarantee R-(m,k)-firm constraint to (*r,b*) upper bounded flows. In addition, this mechanism can be implemented in Diffserv to provide the real-time transmissions QoS guarantee [LiAINA06].

2 Future work

We have deeply researched in real-time schedulability for performance degradation with loss-tolerance applications. The future work can be carried out in following points:

- One of the challenges is how to specify the loss-tolerant degree (or degradation degree of quality of service) for a concrete task or multimedia stream. This degree should be specified according to the human perception and the real-time stream's characteristics. Fortunately, m and k of (m,k)-firm and R-(m,k)-firm constraints can be set as any natural number with the only condition $m<k$. Therefore, the work in this thesis can provide the fundamental theorems for QoS guarantee for any specified degree.
- We plan to implement DLB mechanism in software router, Click modular, developed by MIT LCS's Parallel and Distributed Operating Systems group (now part of MIT CSAIL). Click routers are flexible, configurable, in which we can observe and analyze the stability of QoS management systems.
- We should also research in the various scheduling algorithms for R-(m,k)-firm constraint, synthesizing the QoS requirement of applications and network mechanisms.

Appendix A

Comments on "Dynamic Window-Constraint Scheduling of Real-Time Streams in Media Servers"

In this appendix, we will point out errors in the paper [West04] of R. West, Y.T. Zhang, K. Schwan and C. Poellabauer, published in IEEE transaction on computer and give a correct proof of theorem 3.

1 Error Indications

In the above paper, theorem 3 and Corollary 1 make up a principal contribution; however, we find some incorrectness in the proofs. The theorem 3 is given as:

Theorem 3 of [West04]: In each nonoverlapping window of size q in the hyperperiod, H, there cannot be more than q streams out of n with current window-constraint $0/y'_i$ at any time, when $U = \sum_{i=1}^{n} \frac{(y_i - x_i)}{q y_i} \le 1.0$.

This theorem leads to the major interesting contribution that utilization factor could arrive to *100%*, which is given in corollary 1, such as:

Corollary 1 of [West04]: Using DWCS, the least upper bound on the utilization factor is 1.0, for the set Γ, in which each stream $S_i \in \Gamma$ is characterized by the 3-tuple $(C_i = K;\ T_i = qK;\ W_i = x_i/y_i)$.

Firstly, we will show the errors in brief. Note that the proof of theorem 3 is based on Lemma 1 (see lines 8 and 14, column 2 page 757), and the errors exist in the proof of Lemma 1 as well.

Lemma 1 of [West04]: Consider a set of n streams, $\Gamma = \{S_1, \ldots, S_n\}$, where $S_i \in \Gamma$ is defined by the 3-tuple with the same windows constraint factors ($C_i = K$; $T_i = qK$; $w_i = x_i/y_i$). If the utilization factor, $U = \sum_{i=1}^{n} \frac{(y_i - x_i)}{qy_i} \leq 1.0$, then $x_i = y_i - 1$ maximizes n.

In the proof of Lemma 1, (on line 5 column 2 page 751), it says that "if all window-constraints are equal for each and every stream", which is not a precondition anywhere. It is clear that this condition can not be used in proof since it is not a characteristic of stream set. Again, theorem 3 is neither correct since its proof is based on Lemma 1.

The problem of Lemma 1 is that "if all window-constraints are equal for each and every stream" is an added condition in the proof. Clearly, due to the integrality of the proof, the partner of "if", "else" case must be thought of, but it is lacked. We also traced the trajectory of the paper, theorem 3 were presented in authors' other papers [West99] [West00] with different proofs, but none gave the satisfying result. Furthermore, we must indicate that Lemma 1 was given in an ambiguous way, and it is not correct at all.

Until now, the bare-bones of the problem have now been indicated. Secondly, we will start our deeper analysis of Lemma 1. This lemma is given in an ambiguous way. From the text, it is clear that C_i and T_i are constant values, and the resource is 1. However, x_i and y_i cause the ambiguity. One comprehension is that y_i is already fixed, and in this condition, it is clear that a greater x_i value leads to a smaller utilization factor, such that more streams can be schedulable (maximize n). This case is evidently true, but with the proof of theorem 3, we found that the proof is based on another case in which x_i and y_i are variables.

The other comprehension is that x_i and y_i are arguments. The problem is how to maximize the schedulable stream set number n, as in the proof of theorem 3, under the

constraint of $\sum_{i=1}^{n} \frac{(y_i - x_i)}{q y_i} \le 1.0$. Note that Lemma 1 is not tenable with this comprehension, since we can easily demonstrate one counterexample. Such as: stream set Γ_1 with two streams ($x_1=x_2=1$, $y_1=y_2=2$) is schedulable, however, Γ_2 with three streams ($x_1=x_2=x_3=4$, $y_1=y_2=y_3=6$) can schedulable. Obviously, Γ_1 corresponds to Lemma 1 (as $x_1=y_1-1$, $x_2=y_2-1$), but it does not maximize stream numbers in comparison with Γ_2.

In a word, we should say that the greater the value of each x_i/y_i is (instead of $x_i=y_i-1$), the more streams can be scheduled. This is because that greater x_i/y_i leads to a smaller utilization factor $U = \sum_{i=1}^{n} \frac{(y_i - x_i)}{q y_i} \le 1.0$. As the above counterexample, $1/2<4/6$, this causes the Γ_2 with 3 streams can be schedulable rather than Γ_1 with 2 streams. After all, it is obvious that the case of $y_i \to \infty$ and $x_i=y_i-1$ maximizes $n \to \infty$, and in this case the server could do nothing no matter when. What is the meaning of this system?

2 Re-proof of the DWCS-2 sufficient condition

As stated in previous section, bandwidth utilization section (*section 4.2 in [West04]*) has collapsed, as *Lemma 1, theorem 3 and Corollary 1* in the paper are not correct or not correctly proven. In fact, except Lemma 1, the idea of utilization factor is correct in our point of view, so that we will prove them in what follows.

According to the ambiguity of Lemma 1, we will prove the case of fixed y_i, that is to say all $y_i = y$. In this special case, theorem A is very simple to be proven, and is stated:

Theorem A: Consider a set of n streams, $\Gamma = \{S_1, \ldots, S_n\}$, where $S_i \in \Gamma$ is defined by the 3-tuple with the same windows constraint factors ($C_i = K$; $T_i = qK$; $w_i = x_i/y$). If the utilization factor, $U = \sum_{i=1}^{n} \frac{(y - x_i)}{q y} \le 1.0$, then no more than q streams out of n have 0 window-constraint at any time. Such that the stream set is schedulable under DWCS scheduling.

Proof: We can say that if no violation of window constraints occurs in a window size, all Window-Constraint factors will be reset as the initial value as $w_i = x_i/y$.

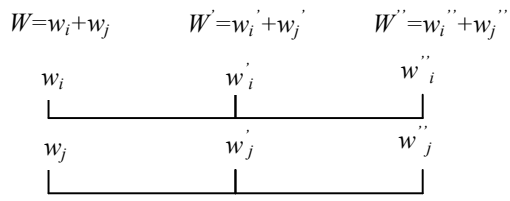
Lemma A: In every window, the sum of Window-Constraint ($W=\sum_{i=1}^{n} w_i$) at the period beginnings is a monotone increasing function.

Proof: We think of the condition n>q, because otherwise deadline misses never occur.

In every period, q streams can be serviced and n-q streams' packet will be lost. Such that the stream set can be split into two subsets:

1). q streams are serviced in the period and WC are modified as $\frac{x_i}{y-1}$ after the current period (denoted as α).

2). n-q streams are not serviced and their WC turn as the $\frac{x_j-1}{y-1}$ after the current period (denoted as β).



then, the change of sum of window-constraint values can be illustrated as follows:

$$W'\text{-}W=\sum_{\alpha}(w_i'-w_i)+\sum_{\beta}(w_i'-w_i)=\sum_{\alpha}(\frac{x_i}{y-1}-\frac{x_i}{y})+\sum_{\beta}(\frac{x_j-1}{y-1}-\frac{x_j}{y})=\sum_{\alpha}(\frac{w_i}{y-1})+\sum_{\beta}(\frac{w_j-1}{y-1})$$

$$=\sum_{\alpha+\beta}(\frac{w_i}{y-1})-\sum_{\beta}(\frac{1}{y-1})\geq\frac{1}{y-1}(\sum_{\alpha+\beta}w_i-(n-q))$$

with the condition of $\sum_{i=1}^{n}\frac{(y-x_i)}{qy}\leq 1.0$, it is easy to get $\sum_{i=1}^{n}w_i-(n-q)\geq 0$. Such that $W'\text{-}W\geq 0$.

In the same way, $W\leq W'\leq W''$ and so forth till the beginning of the last period in the window. At the end of window, all window-constraints are reset as initial value.

End of proof

According to Lemma A, at each and every end of period, the sum of window-constraint value is never smaller than (n-q), and because each window-constraint is no higher than 1. Therefore, no more than q zero window-constraints can exist in any period. Thus, all window-constraints can be satisfied.

End of proof in condition of the same window size.

The more general case, when y_i and x_i are variables, is more complicated to prove. As stated in [West04], window-constraint and (m,k)-firm constraint can be transformed each other. Moreover, Window-constraint has been proven as NP-hard in strong sense [Mok01]. Its similar constraint (m,k)-firm has also been proven as NP-hard in strong sense [Quan00]. This means that no optimal scheduling can be found for them. Furthermore, in [Mok01], under certain special conditions, there are the efficient on-line polynomial-time optimal algorithms.

One special condition is: streams are synchronized at time 0, have unit service time and their periods are multiple of service time. Under these conditions, one optimal P-fair window-constraint stream set is schedulable in condition of $\sum_{i=1}^{n} \frac{(y_i - x_i)}{q y_i} \le 1.0$ [Mok01]. We should explain P-fair window-constraint at first.

We prove it with the concept of P-fair window-constraint introduced in [Mok01]. Unit size assumption [Mok01] states: if the execution time of all the tasks in a task set is 1, then the task set is a unit-size task set, and its scheduling problem is a unit-size scheduling problem.

A schedule is P-fair if and only if the schedule satisfies not only the constraints defined in the definition of unit-size schedule, but also scope constraint as: the lth P-fair scope of stream i is defined as: $\left[r_i + \left\lfloor \frac{l \cdot k_i}{m_i} \right\rfloor \cdot p_i, r_i + \left\lceil \frac{(l+1) \cdot k_i}{m_i} \right\rceil \cdot p_i \right]$, where l denotes a time interval length, r_i denotes the release time, p_i is the period, m_i and k_i denotes that m_i deadlines should be met among k_i consective ones.

Lemma B: if a stream set satisfies P-fair window-constraint, then, it has already satisfied window-constraint in [West04].

Proof: Obviously, the beginning of $((a-1)m)_{th}$ P-fair scopes and the end of $(am)_{th}$ P-fair scopes (where a is natural number) equal the beginning and the end the a_{th} window size of Window-constraint, respectively. Therefore, if P-fair window-constraint is satisfied from $((a-1)m)_{th}$ instance to $(am)_{th}$ instance, then window-constraint is surely satisfied in a_{th} window for window-constraint.

End of proof.

Intuitively, window-constraint is less restricted than P-fair window-constraint, since it doesn't require the distribution of executions in windows.

Theorem B: there is a set of n streams, $\Gamma = \{S_1, \ldots, S_n\}$, where $S_i \in \Gamma$ is defined by the 3-tuple ($c_i = K;\ p_i = qK;\ w_i = x_i/y_i$). If the utilization factor satisfies, $U = \sum_{i=1}^{n} \frac{(y_i - x_i)}{qy_i} \leq 1.0$, then all windows constraints can be satisfied.

Proof:

We insert one virtual stream with a window constraint $\mu/\mu=1$, where μ is one very small positive number (for example 0.001), the window-constraint value is reset at every period. The virtual stream is defined as ($c_v = K;\ p_v = K;\ w_v = \mu/\mu$). Clearly, this virtual stream has the highest priority among value 1 window-constraints. The service of this virtual stream can never cause the failure state of the system, because it is serviced only when all streams have the value 1 window constraints. Therefore, for every stream, it can be serviced exactly (y_i-x_i) times in each window under feasible situation, otherwise, system fails into failure state.

In P-fair schedule, the server could be idle in condition that all P-fair constraints have been satisfied. The idle time can be mapped to Virtual Stream service in window constraint, such that the problem of schedule of window constrained stream set (except virtual stream) can be constructed with a one-to-one mapping with P-fair scheduling.

In fact, DWCS only needs to re-sort the P-fair schedule pattern according to its rules shown in Table 2 in [West04], and the process is shown in Algorithm 1. Since all streams have been allocated exactly (y_i-x_i) execution time, this re-sort program must be able to produce correct DWCS pattern in polynomial time. After this mapping, the second step is

to distribute occupations of the Virtual Stream to other normal streams in set, as shown in Algorithm 2. The distribution is done according to the lowest window-numerator first order according to Table 2 in [West04] as well. So far, the stream set (without virtual stream) schedule has been constructed.

So due to the construction, windows constraint schedule exists.

End of Proof

Notice that DWCS can give a high utilization schedule even the proof has problem. While based on the results in relative paper [Mok01], we gave the proof as above.

3 Pseudo-code:

```
produce the P-fair window-constraint pattern;
t=0;
initialize window-constraint;
While (t<Hyperperiod)
 {
   a = number of streams which have window-constraint value inferior to 1;
  if (a<=q)
   {
     move forwards a determined streams to this period;
     allocate (q-a) to virtual streams;
     move backwards other streams in current period;
   }
  else
   {
     move forwards q streams with the highest priority streams according to Table 2 in [West04];
     move backwards other streams in current period;
   }
  modify window-constraint values;
  t=t+qK;
 }
```

Algorithm 1: Re-sort process from P-fair window-constraint

```
t=0 ;
while (t<Hyperpeirod)
 {
   if (virtual stream is serviced at t)
     {
      allocate t time slot to stream S_i, according to Table 2 in [West04];
      t=t+K;
```

```
    }
    else
      t=t+K;
}
```

Algorithm 2: Distribution of Virtual Stream

Since the existence of P-fair window constrant pattern has been proven with the added virtual stream, Algorithm 1 does the mutation of executed instance of P-fair window-constrant to produce a successful execution pattern corresponding to Dynamic Window-Constraint Scheduling. Algorithm 2 distributes the execution time to other streams, which are allocated to the virtual stream in Algorithm 1. Hence, the existence of DWCS pattern can be proven.

Appendix B

Fatality of (m,k)-firm constraint

As know, the workload of a periodic task set under HRT and (m,k)-firm constraint are $\sum_{i=1}^{n} \frac{c_i}{p_i}$ and $\sum_{i=1}^{n} \frac{m_i}{k_i} \frac{c_i}{p_i}$, respectively. If a task set can tolerate some miss deadlines according to (m,k)-firm constraint, the workload will be much less.

In this partition, we will analysis whether a periodic task set (with the same period and execution time) can require less resource (server capacity) under (m,k)-firm constraint than under hard real-time constraint. In other words, we will demonstrate how much the gain of utilization (m,k)-firm constraint can achieve. As known, the resource utilization is very low for a non-preemptive system which serves a task set under HRT or (m,k)-firm constraint. Let R_1 denote the resource requirement which can provide the HRT constraint for a periodic task set. Actually, the practical advantage of (m,k)-firm constraint is to decrease the resource requirement as less as possible to $R \cdot \left(\frac{\sum_{i=1}^{n} \frac{m_i}{k_i} \frac{c_i}{p_i}}{\sum_{i=1}^{n} \frac{c_i}{p_i}} \right)$. Unfortunately, the result is shown very pessimistic. That is to say a periodic task set could require the same resource as HRT for whatever (m,k)-firm scheduling, such that (m,k)-firm constraint losses its motivations.

The pessimistic result is given in the following theorem, which is established by seeking the special (m,k)-firm for each task with which the resource requirement for the task set equals that of HRT.

Theorem B-1: For any given periodic task set $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ $(n \geq 2)$, where $\tau_i = (c_i, p_i)$. There must be a (m_i,k_i)-firm constraint for each task τ_i, such that it forces the server to service all instances of a task τ_i instead of the (m,k)-firm constraint, whereas (m_i, k_i) is given as $m_i<k_i$.

Proof: assume the system is overloaded, because otherwise (m,k)-firm is not necessary for this system. Therefore, HRT can not be guaranteed and some instances must be rejected without execution due to (m,k)-firm scheduling.

It has been given in [George00] that a periodic task is schedulable if there is no missed deadline in [0, $r+2LCM$], where r stands for the last release time of all tasks and LCM is the *least common multiple of* $\{p_1,...,p_n\}$). Hence, there must be some missed deadlines (be dropped) in the interval [0, $r+2LCM$], because otherwise HRT will be satisfied. Let τ_i be a task with at least one missed deadline, and let t_i be the invocated time (beginning time of period) of the missed instance.

It can been mentioned in formula (23) (in section 2.3 of chapter 2) that any scheduling of a periodic task set under (m,k)-firm constraint after the time $L_{max} = r+ \left(\prod_{i=1}^{n} (k_i - m_i + 1) + 1 \right) \mathrm{LCM}$ must be the iteration in the time interval [0, L_{max}]. Therefore, the task τ_i has no less than one missed deadline in [0, L_{max}], and another instance must be missed as well with the period beginning at t_i+L_{max}.

Based on these two given results, we can determine a (m_i,k_i)-firm constraint for task τ_i , which can not be satisfied. Let $L_{max} - r = n_i p_i$, then we configure task τ_i with $(m_i,k_i)=(n_i,n_i+1)$.

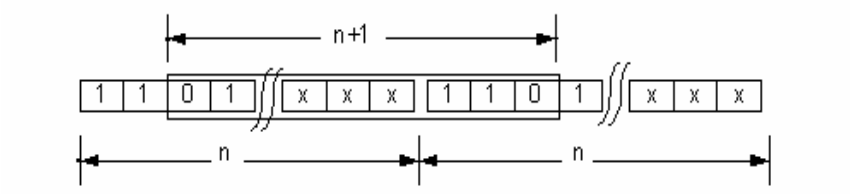


Fig. A- 1: Reverse orders of two scheduling method

Note that the instances of task τ_i invoked at time t_i and $t_i+L_{max}-r$ can not be scheduled and the deadlines are missed according to the same scheduling algorithm. In the other word, if any one deadline of the two instances could be met, the other will be surely met. Hence, if we configure the task τ_i with (n_i,n_i+1)-firm constraint, the scheduling can only service all instances of task τ_i without missed deadlines to satisfy (n_i,n_i+1)-firm constraint.

All the other tasks can be configured according to the above method, such that the non deadline can be missed even every task has a (m,k)-firm constraint (m<k).

End of Proof

It has been proven that a special (m,k)-firm set can be found for a given task set such that the server is obliged to service all instances. Hence, this task set requires no less resource for satisfying the found (m,k)-firm constraint (m<k) than for satisfying HRT constraint. In addition, we will show that this fact has many possibilities.

Lemma B-1: For any given task set, every task has infinite probabilities aggregate of (m,k)-firm configuration to impose the processor execute all of the instances of any task**.**

Proof : From the upper result, at least we can extend the *(m,k)*-firm from (n_i,n_i+1) to $(2n_i,2n_i+2)$ and so on until infinite. Although this infinite set is the subset of which can impose the processor execute the task with HRT constraint, the set is already proven as infinite. So we can prove this lemma.

End of Proof

Discussion

After the demonstration of the pessimistic result, we should give two points.

First, note that the special (m,k)-firm set is configured according to a large multiple of *LCM* of the period, such that the value of m_i and k_i may be very large. In fact, it could be easily achieved in some cases. For example, a task set $\Gamma = \{\tau_1, \tau_2\}$, and $(c_1, p_1, m_1, k_1)=(c_2, p_2, m_2, k_2) =(6, 10, 2, 3)$. Obviously, the system is overload for HRT since $\sum_{i=1}^{2} \frac{c_i}{p_i} > 1$. However, we have $\sum_{i=1}^{2} \frac{m_i}{k_i}\frac{c_i}{p_i} < 1$, such that (m,k)-firm constraint is wished to be schedulable with some loss tolerance. Unfortunately, the following Fig. A-2 shows that it can not be schedulable for any scheduling method.

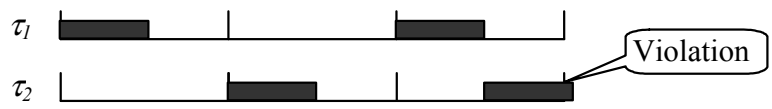


Fig. A-2: Violation example of (m,k)-firm

In Fig. A-2, two tasks restricted by (2,3)-firm constraints always oblige the server to finish one instance of two tasks in one period. If this fact can be satisfied, HRT is already realized.

Second, there is an infinite set of (m_i,k_i)-firm values for any task which imposes the server to execute all instances as HRT, and a task can not avoid the (m,k)-firm configuration to be included in it. The infinite set is configured by (n_i,n_i+1) and its multiples, where $n_i = (L_{max}-r)/p_i$, $L_{max} = r+ \left(\prod_{i=1}^{n} (k_i - m_i + 1) + 1 \right) \mathrm{LCM}$. Observe that n_i is configured for task τ_i based on the parameters of other tasks; in other words, the special set of (m_i,k_i)-firm constraint is affected by the other tasks. However, a task invokes the instances without the awareness of other task parameters, and a (m,k)-firm constraint is the private real-time constraint for a task which is given according to the task's characteristics. One task can not know the parameters of the other tasks, and in real networks, the tasks start and terminate at anytime. Moreover, the set of pointe to be tested is much bigger than our given

infinite special set of (m,k)-firm constraint. So the (m,k)-firm configuration of a task can not be configured deterministically outside of the given special (m,k)-firm constraint set.

Reference:

[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.

[Baruah96] S. Baruah, N. Cohen, G. Plaxton, and D. Varvel, "Proportionate Progress: A Notion of Fairness in Resource Allocation," Algorithmica, vol. 15, no. 6, pp. 600-625, June 1996. Extended Abstract in Proc. 1993 ACM Ann. Symp. Theory of Computing.

[Baruah98] Baruah, S. K. and Lin, S. 1998. "Pfair Scheduling of Generalized Pinwheel Task Systems". IEEE Trans. Comput. Vol 47, no. 7, pp 812-816, Jul. 1998.

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

[Bernat03] G. Bernat "Response Time Analysis of Asynchronous Real-Time Systems"; Real-Time Systems, 25, 131-156, 2003

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

[Blazewicz76] Blawewicz J. "Scheduling dependent tasks with different arrival times to meet deadlines," in H. Beilnerand E. Gelenbe (eds.) Modelling and Performance Evaluation of Computer Science, North Holland, Amsterdam, 57-65. 1976

[Bolot95] Bolot J. C., Crépin H., Garcia A. V., "Analysis of Audio Packet Loss in the Internet", Proceedings of Networks and Operating System Support for Digital Audio and Video, p. 163-174, avril 1995.

[Bolot96] J. Bolot and A. Vega-Garcia. "Control Mechanisms for Packet Audio in the Internet", Proceedings of IEEE Infocom, 1996.

[Braden98] Braden, B., D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, Recommendations on Queue Management and Congestion Avoidance, RFC 2309, April 1998.

[Carlier78] Carlier J "Problem a une machine," Report No. 78.05, Institut de Programmation, Universite de Pierre et Marie Curie, Paris, France.1978

[Chang00] Cheng-Shang Chang, "Performance Guarantees in Communication Networks" by Springer Verlag, 2000

[Cruz91] R. L. Cruz, "A calculus for network delay, Part I". *IEEE Transactions on Information Theory*, 37(1):114-131, Jan. 1991.

[Davie02] B. Davie et al., "An expedited forwarding PHB", IETF RFC 3246, March 2002.

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

[Flexray04] FlexRay Consortium, http://www.flexray.com, 2004.

[Floyd01] Floyd, S., RED: Discussions of Setting Parameters, November 1997, http://www.aciri.org/floyd/REDparameters.txt (June 2001).

[Floyd93] Floyd, S., and Jacobson, V., "Random Early Detection Gateways for Congestion Avoidance". In ACM/IEEE Transactions on Networkingg, 3(1), August 1993.

[Floyed92] S. Floyd and V. Jacobson. "On Traffic Phase Effects in Packet-Switched Gateways." Internetworking: Research and Experience, 3(3):115–156, September 1992.

[Ford62] L. Ford and D. Fulkerson, Flows in Networks. Princeton, N.J.:Princeton Univ. Press, 1962.

[FWang01] F. Wang and P. Mohapatra, “Using differentiated services to support Internet telephony”, Computer communications, Vol.24, Issue18, pp1846-1854, Dec. 2001.

[Garey77] Garey, M. R., D. S. Johnson, “Two-processor scheduling with start-time and deadlines”, SIAM J. Comput. 6, 416-426.

[Garey78] Garey, M. R., D. S. Johnson, B. B. Simons, R. E. Tarjan, “ Scheduling unit time tasks with arbitrary release times and deadlines,” unpublished results, 1978

[George00] L. Georges, P. Mühlethaler, N. Rivierre. “A Results on Non-Preemptive Real time Scheduling”, INRIA research report N° 3926, may 2000.

[George96] L. GEORGE, N. RIVIERRE, M. SPURI, “Preemptive and Non-Preemptive Real-Time UniProcessor Scheduling”, INRIA Research Report, No. 2966, September 1996

[Hamdaoui95] M. Hamdaoui 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] Moncef Hamdaoui, Parameswaran Ramanathan, "Evaluating Dynamic Failure Probability for Streams with (m, k)-Firm Deadlines," IEEE Transactions on Computers, vol. 46, no. 12, pp. 1325-1337, Dec., 1997.

[Holte89] R. Holte, A. Mok, L. Rosier, I. Tulchinsky, and D. Varvel. “The pinwheel: A real-time scheduling problem”. In Proc. of the 22nd Hawaii International Conference on System Science, pages 693–702, January 1989.

[Horn74] W. A. Horn. “Some Simple Scheduling Algorithms.” Naval Research Logistics Quarterly, 21:177–185, 1974.

[Jacobson] Van Jacobson. “Notes on using RED for Queue Management and Congestion Avoidance” ftp://ftp.ee.lbl.gov/talks/vjnanog-red.ps.gz

[Jeffaty91] K. Jeffay, D.F. Stanat, and C.U. Martel, “On Non-Preemptive Scheduling of Periodic and Sporadic Task”, IEEE real-time systems symposium, pp129-139, San Antonio (USA), Dec. 4-6, 1991.

[Jia05] N. Jia, E. Hyon, Y. Song, “Ordonnancement sous contraintes (m,k)-firm et combinatoire des mots”, RTS’2005, Paris (France), April 2005.

[Kleinrock76] L. Kleinrock. "Queueing Systems, Volume 2: Computer Applications. Wiley, 1976.

[Koren95] Gilad Koren and Dennis Shasha. Skip-over: Algorithms and complexity for overloaded systems that allow skips. In Real-Time Systems Symposium, 1995.

[Koubaa04a] A. Koubâa, Y.Q. Song, "Evaluation and improvement of response time bounds for real-time applications under non pre-emptive fixed priority scheduling", International Journal of Production Research, Vol.42, No.14, pp2899-2913, Taylor & Francis group, July 2004.

[Koubaa04b] Koubâa, A., Song, Y.Q., “Loss-Tolerant QoS using Firm Constraints in Guaranteed Rate Networks”, 10th IEEE Real-Time and Embedded Technology and Applications (RTAS'2004), Toronto (Canada) 25-28 May 2004.

[Koubâa05] A. Koubâa, Y.Q. Song, "Graceful Degradation of Loss-Tolerant QoS using (m,k)-Firm Constraints in Guaranteed Rate Networks", Computer Communications, Vol.28, pp1393-1409, 2005.

[Lawler73] Lawler E. L., “Optimal sequencing of a single machine subject to precedence constraint,” Management Sci. 19, 544-546. 1973

[LeBoudec02] J.Y. Le Boudec , P Thiran, “Network Calculus : A Theory of Deterministic Queueing Systems for the Internet”, Springer Verlag July 2002

[Li06TII] Li, J.; Song, Y.; Simonot-Lion, F. “Providing Real-Time Applications with Graceful Degradation of QoS and Fault Tolerance According to (m, k)-Firm Model” IEEE Transactions on Industrial Informatics, May 2006 Volume: 2, Issue: 2 P.112- 119 ISSN: 1551-3203.

[LiAINA06] Jian Li, YeQiong Song, "DLB: A Novel Real-time QoS Control Mechanism for Multimedia Transmission," aina, pp. 185-190, 20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06), 2006.

[Liberato99] Frank Liberato, Sylvain Lauzac, Rami Melhem, Daniel Mosse. “Fault Tolerant Real-Time Global Scheduling on Multiprocessors”, p.0252, 11th Euromicro Conference on Real-Time Systems, 1999.

[LiDEA03] J. LI. 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, Jun, 2003.

[LiJDIR05] Li Jian, Song YeQiong “DLB: A novel real-time QoS control mechanism for multimedia transmission”, 7eme Journées Doctorales Informatic et Reseau, JDIR’06, Dec 13-15, 2005.

[LiRTNS06] Jian Li, YeQiong Song, “R-(m,k) firm: A novel QoS scheme for real-time flow guarantee in Networks” 14TH International Conference on Real-Time and Network Systems, RTNS’06, Poitiers, France May 30-31, 2006

[Liu and Hu06] Liu, D.; Hu, X.S.; Lemmon, M.D.; Ling, Q. “Firm real-time system scheduling based on a novel QoS constraint”, IEEE Transactions on Computers, Volume 55, Issue 3, March 2006 Page(s):320 - 333

[Liu69] C. Liu, “Scheduling Algorithms for Multiprocessors in a Hard Real-Time Environment,” JPL Space Programs Summary 37-60, vol. II, pp. 28-37, 1969

[LiWFCS04] Li, Jian and Song, YeQiong and Simonot-Lion, Françoise Schedulability analysis for system under (m,k)-firm constraints. In 5th IEEE International Workshop on Factory Communication System - WFCS'2004, Vienne, Autriche, Sep 2004.

[LM80] J. Y.T.Leung, M.L.Merril, “A note on preemptive scheduling of periodic, Real Time Tasks”, Information processing Letters, Vol. 11, n o 3, Nov 1980.

[Mok01] Mok, A.K. and W. R. Wang, “Window-Constrained Real-Time Periodic Task Scheduling”, 22nd IEEE Real-Time Systems Symposium (RTSS'01), pp15-24, London, England, December 03 - 06, 2001.

[Navet00] N. Navet, Y.Q. Song, F. Simonot, "Worst-case deadline failure probability in real-time applications distributed over CAN (controller area network)", Journal of systems architecture - the EUROMICRO Journal, 46 (2000) pp607-617.

[OSEK01] OSEK, OSEK/VDX operating system, version 2.2, 2001. http://www.osek-vdx.org

[Parekh92] A. Parekh. "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks. PhD dissertation, Massachusetts Institute of Technology, February 1992.

[Poggi03] E.-M. Poggi, 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] G. Quan and X. Hu, “Enhanced Fixed-priority Scheduling with (m, k)-firm Guarantee”, Proc. Of 21st IEEE Real-Time Systems Symposium, pp.79-88, Orlando, Florida, (USA), November 27-30, 2000.

[Ramanathan99] P. Ramanathan, "Overload management in real-time control applications using (m; k)-firm guarantee," IEEE Transactions on Parallel and Distributed Systems (Special issue on Dependable Real-time Systems), pp. 549-559, June 1999.

[Roadmap03] Project IST-2002-34820, Roadmap, “Adaptive Real-Time Systems for Quality of Service Management”, http://www.systemes-critiques.org/ARTIST/Roadmaps, May 6th 2003.

[RV94] ISO, Road vehicles – Interchange of digital information – Controller area network for high-speed communication, ISO 11898, International Organization for Standardization (ISO), 1994.

[Simons78] Simon B., "A fast algorithm for single processor scheduling," Proc. 19th Ann. Symp. On Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 246-252., 1978

[Sorenson75a] Sorenson, P.G. "A Methodology for Real-Time System Development", Ph.D. Thesis, University of Toronto, June 1974.

[Sorenson75b] Sorenson, P.G., Hamacher, V.C. (1975). A Real-Time Design Methodology, INFOR, Vol. 13, No., (February 1975), pp. 1-18.

[Striegel03] Striegel A., G. Manimaran, "Dynamic Class-Based Queue management for scalable media servers", Journal of systems and software, vol.66, No.2, pp.119-128, May 2003.

[Tindell94] K. Tindell Fixed-Priority Scheduling of Hard Real-Time Systems. PhD thesis, University of York, UK, 1994.

[Tindell95] K. Tindell, A. Burns and A.J. Wellings "Analysis of Hard Real-Time Communications" Real-Time Systems, Vol. 9(2), pp. 147-171, Kluwer Academic Publishers (September 1995).

[WCFeng02] Wu-chang Feng, Kang G. Shin, Dilip D. Kandlur, and Debanjan Saha. The BLUE active queue management algorithms. IEEE/ACM Transaction on Networking, 10(4):513--528, August 2002.

[West00] R. West and C. Poellabauer, "Analysis of a Window-Constrained Scheduler for Real-Time and Best-Effort Packet Streams", Proc. of 21st IEEE Real-Time Systems Symposium, Orlando, Florida, (USA), November 27-30, 2000.

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

[Wilwert05] C. Wilwert, N. Navet, Y.Q. Song, F. Simonot-Lion, "Design of Automotive X-by-Wire Systems", to appear in The Industrial Communication Technology Handbook (edited by R. Zurawski), CRC Press, 2005.

[Zhang04] Yuting Zhang, Richard West and Xin Qi, "A Virtual Deadline Scheduler for Window-Constrained Service Guarantees", in Proceedings of the 25th IEEE Real-Time Systems Symposium (RTSS), December 2004.

[ZWang02] Z. Wang, Y.Q. Song, E.M. POGGI and Y.X. Sun, "Survey of Weakly-Hard Real Time Schedule Theory and Its Application", International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES2002), Wuxi (china), Dec. 16-20, 2002.

PERSONAL PUBLICATIONS

Journal and book chapter:

[1] Li, J.; Song, Y.; Simonot-Lion, F. “Providing Real-Time Applications with Graceful Degradation of QoS and Fault Tolerance According to (m, k)-Firm Model” IEEE Transactions on Industrial Informatics, Publication Date: May 2006 Volume: 2, Issue: 2 P.112- 119.

[2] Ye-Qiong Song, A. Koubaa, J. Li. “Qualité de service temps réel selon le modèle (m,k)-firm”, Systèmes temps réel 2: Ordonnancement, réseaux et qualité de service (Traité IC2, série Informatique et systèmes d'information), ISBN : 2-7462-1304-4, June 2006.

[3] Jian Li, YeQiong Song, "DLB: A Novel Real-time QoS Control Mechanism for Multimedia Transmission", Special Issue on Performance Evaluation of Web and Grid Based Computing of IJHPCN , accepted for 2nd round revision review.

International Conference:

[4] Li, Jian and Song, YeQiong and Simonot-Lion, Françoise “Schedulability analysis for system under (m,k)-firm constraints”. In 5th IEEE International Workshop on Factory Communication System - WFCS'2004, Vienna (Austria), Sep. 2004.

[5] Jian Li, YeQiong Song, "DLB: A Novel Real-time QoS Control Mechanism for Multimedia Transmission", 20th International Conference on Advanced Information Networking and Applications (AINA'06) - Volume 1, pp. 185-190, 18-20 April 2006, Vienna, 2006.

[6] Jian Li, YeQiong Song, “R-(m,k) firm: A novel QoS scheme for real-time flow guarantee in Networks” 14TH International Conference on Real-Time and Network Systems, RTNS’06, Poitiers, France May 30-31, 2006. Also selected for the special issue in International Journal of Embedded Systems.

[7] Jian Li, YeQiong Song, “Relaxed (m,k)-firm Constraint to Improve Real-time Streams Admission Rate under Non Pre-emptive Fixed Priority Scheduling” 11th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA’06, , Prague, Czech Republic, September 20-22, 2006.

Colloquia:

[8] LI Jian, Song YeQiong “DLB: A novel real-time QoS control mechanism for multimedia transmission”, 7eme Journées Doctorales en Informatique et Reseaux, JDIR’06, Dec. 13-15, 2005.

[9] Li Jian, Song YeQiong, “A Graceful QoS Degradation Scheme for Loss Tolerant Real-time Applications”, Third Taiwanese-French Conference on Information Technology,Nancy, France, March 28-30, 2006

Report:

[10] Li, Jian and Song , YeQiong and Simonot-Lion, Françoise. “Providing real-time applications with graceful degradation of QoS and fault tolerance according to (m,k)-firm model. INRIA Technical Rapport, A05-R-406. 2005.

Memory:

[11] Jian. Li. “Sufficient Condition for Guaranteeing (m,k)-firm Real-Time Requirement Under NP-DBP-EDF Scheduling.” Technical report No. A03-R-452, MSc. thesis, LORIA, June, 2003.

Résumé

Cette thèse se focalise sur le développement des algorithmes d'ordonnancement sous contrainte (m, k)-firm, ainsi que leurs applications pour la gestion de la qualité de service (QdS) dans les réseaux et systèmes temps réel distribués. L'objectif recherché est la garantie déterministe de la QdS tout en maintenant un fort taux d'utilisation des ressources.

Les contributions sont (1) l'établissement d'une condition suffisante d'ordonnançabilité d'un ensemble de tâches sous l'algorithme « distance based priority »; (2) la définition de R-(m, k)-firm, un nouveau modèle qui relâche la contrainte (m, k)-firm et qui permet de modéliser de façon plus juste des exigences du temps réel souple; (3) le développement d'un algorithme efficace de dimensionnement de ressources sous contrainte (m, k)-firm relâchée; (4) la proposition de « Double Leaks Bucket » pour la gestion active de files d'attente permettant de maintenir une QdS en cas de surcharge des réseaux.

Mots clés : Temps réel, (m,k)-firm, Gestion active de file d'attente, Ordonnancement, Qualité de service.

Abstract

This work focuses on the scheduling algorithms under (m,k)-firm constraint, as well as the applications for QoS (quality of service) management in the networks and distributed real-time system. The research aim is to achieve the deterministic guarantee of QoS with high resource utilization.

The contributions in this thesis include (1) proposing a sufficient condition for determining the schedulability of a real-time task set under Distance Base Priority scheduling algorithm; (2) defining a novel real-time constraint which relaxes the (m,k)-firm constraint and provides a more suitable modelling of soft real-time; (3) developing an effective resource provisioning algorithm under this relaxed (m,k)-firm constraint; (4) proposing an active queue management mechanism, called Double Leaks Bucket, which can guarantee the QoS with dynamic dropping of the packets during the networks overload period.

Key words: Real-time, (m,k)-firm, Active Queue Management, Scheduling, Quality of Service.

INSTITUT NATIONAL
POLYTECHNIQUE
DE LORRAINE

AUTORISATION DE SOUTENANCE DE THESE
DU DOCTORAT DE L'INSTITUT NATIONAL
POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :
Madame Pascale MINET, Chargée de Recherche, INRIA Rocquencourt, Le Chesnay
Monsieur Pascal LORENZ, Professeur, Université de Haute Alsace, Colmar

Le Président de l'Institut National Polytechnique de Lorraine, autorise :

Monsieur LI Jian

à soutenir devant un jury de l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :

"Garantir la qualité de service temps réel selon l'approche (m,k)-firm"

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : « **Informatique** »

Fait à Vandoeuvre, le 05 février 2007
Le Président de l'I.N.P.L.,
F. LAURENT

NANCY BRABOIS
2, AVENUE DE LA
FORET-DE-HAYE
BOITE POSTALE 3
F - 54501
VANDŒUVRE CEDEX

TEL. 33/03.83.59.59.59
FAX. 33/03.83.59.59.55