Une architecture d'ordonnancement par rétroaction pour les systèmes multimédia distribués

Bechir Alaya, Claude Duvallet, Bruno Sadeg

LITIS, UFR des Sciences et Techniques,
25 rue Philippe Lebon, BP 540,
F-76058 LE HAVRE CEDEX,
{Claude.Duvallet,Bruno.Sadeg}@univ-lehavre.fr

Résumé Les applications multimédia gèrent des quantités volumineuses de données. L'exploitation de ces données doit se faire en respectant les contraintes temporelles permettant de lire les paquets vidéo avec une certaine fluidité. Lorsque les contraintes temporelles ne sont pas respectées, la qualité de service fournie aux utilisateurs diminue. Partant du constat qu'il existe des similarités entre les applications multimédia et les SGBD temps réel, notre approche consiste à exploiter les travaux concernant la gestion de la qualité de service dans les SGBD temps réel afin de les appliquer au domaine des systèmes multimédia. Dans cet article, nous proposons une architecture permettant de garantir la qualité de service fournis aux clients lors des augmentations importantes de charge.

1 Introduction

Les progrès dans le domaine des réseaux et l'amélioration constante des débits permettent d'envisager l'exploitation de nouveaux services tels que ceux liés aux applications multimédia. Ces applications doivent traiter des volumes très importants de données et elles nécessitent que les traitements soient effectués avant des dates fixes pour garantir une bonne qualité de service dans les flux présentés aux utilisateurs. Les systèmes les mieux adaptés pour la gestion de données en quantités importantes tout en tenant compte des contraintes temporelles des applications sont les SGBD temps réel (SGBDTR) [1][2][3].

Beaucoup d'applications multimédia distribuées doivent très souvent faire face à des charges d'utilisation imprévisibles qui causent la surcharge du système. Par exemple, il peut s'agir de demandes utilisateur qui arrivent en grande nombre sur une période très courte (pic d'utilisation). Le besoin d'offrir une bonne qualité de service aux utilisateurs déjà servis est primordial. L'étude de la gestion de la QdS (Qualité de Service) particulièrement la QdS des paquets vidéo permet de répondre à ces nouveaux besoins en adaptant les techniques existantes pour un transfert plus fiable et plus efficace des nouveaux types de paquets vidéo sans changer l'infrastructure initiale. La problématique générale se situe dans l'adaptation des ressources disponibles (bande passante, tampon de stockage, serveurs vidéo, etc.) et la recherche de nouvelles techniques d'adaptation en cas de périodes d'instabilité (surcharge ou sous-utilisation) du système pour assurer une qualité de service acceptable en respectant les exigences multiples des différents flux vidéo.

Des travaux concernant la gestion de la qualité de service dans les SGBDTR ont déjà été effectués [4][5]. À cause des similarités avec les applications multimédia, nous cherchons à appliquer les résultats obtenus sur la gestion de la qualité de service dans les SGBDTR au domaine des applications multimédia. L'objectif à terme est de permettre de concevoir des applications multimédia qui seront en mesure de fournir des garanties de qualité de service et une certaine robustesse lorsque les demandes des utilisateurs croissent rapidement. Ces travaux s'appliquent particulièrement aux applications de *vidéo à la demande* (VoD).

Dans cet article, nous allons présenter l'approche que nous proposons pour la gestion de la qualité de service dans les systèmes multimédia. Dans la section 2, nous développons notre méthode pour la conception d'applications multimédia basées sur une approche "qualité de service". Ensuite, dans la section 3, nous présentons le simulateur que nous avons développé pour tester la validité de notre approche. Enfin, nous concluons cet article par un bilan sur ce travail et les perspectives que nous comptons lui donner.

2 Gestion de la qualité de service dans les systèmes multimédia distribués

La QdS d'une application multimédia est définie comme étant l'ensemble des exigences requises par cette application en termes de bande passante, qualité de visualisation, délai, gigue et taux de perte des paquets vidéo. Ces exigences sont intrinsèques à la nature de l'application. Notre approche consiste à reprendre les travaux effectués dans le domaine de la gestion de la QdS dans les SGBDTR [6][7] et de les adapter aux systèmes multimédia. Pour cela, nous proposons une amélioration de la méthode de contrôle par rétroaction pour les systèmes multimédia distribués.

2.1 Gestion de la qualité de service dans les SGBDTR

La gestion de la qualité de service dans les SGBDTR repose sur l'exploitation d'une boucle de rétroaction [8] permettant d'adapter la qualité des données (temps réel) et des transactions (temps réel) en fonction de l'état de charge du système. Par contre, dans les systèmes multimédia la gestion de la QdS repose sur l'adaptation de la qualité des paquets vidéo.

Dans les SGBDTR, les données temps réel représentent la capture de l'état du monde réel. Ces données doivent être remises à jour régulièrement afin de refléter au plus près le monde réel. Elles possèdent donc une durée de validité qui représente la période pendant laquelle elles peuvent être utilisées. Amirijoo et al. ont introduit la notion de qualité des données (QdD) [7] qui permet de considérer qu'une donnée stockée dans la base peut posséder un certain écart avec sa valeur dans le monde réel. On appelle cet écart "erreur sur la donnée" (noté DE[1]) et on le calcule en faisant la différence entre la donnée en base et la valeur du monde réel. Par exemple, on peut admettre que la donnée température lorsqu'elle vaut "37,2" en base est très proche de "37,3" qui est

[1] Data Error.

la donnée réelle et que, par conséquent, il n'est pas nécessaire de mettre à jour cette donnée. Cet écart possède un seuil (MDE[2]) qui permet de déterminer si la transaction qui souhaite mettre à jour une donnée temps réel peut être écartée (DE $<$ MDE) ou pas (DE $\geq$ MDE).

Les transactions temps réel utilisées dans les systèmes multimédia possèdent des échéances de type *firm* (échéances strictes non critiques [1]). C'est-à-dire que lorsqu'une transaction dépasse son échéance, elle devient inutile pour le système et est abandonnée. Dans les travaux sur la gestion de la QdS, deux types de transactions temps réel sont considérés :

- Les *transactions de mise à jour :* elles ont pour tâche de mettre à jour régulièrement les données acquises auprès de capteurs. Ces transactions sont exécutées périodiquement pour rafraîchir les valeurs des données temps réel.
- Les *transactions "utilisateur" :* elles effectuent des opérations de lecture/écriture de données non temps réel et/ou des lectures de données temps réel. La non prévisibilité de leurs arrivées dans le système et de la charge qu'elles suscitent, rend adéquate l'utilisation d'une architecture basée sur la rétroaction pour gérer les variations importantes de charge [8].

La conception des transactions temps réel est basée sur des techniques de calcul imprécis [9]. Chaque transaction est composée d'une partie obligatoire et de plusieurs parties optionnelles qui sont exécutées suivant le temps disponible pour l'exécution de la transaction. Plus le nombre de parties optionnelles exécutées avant échéance est grand, meilleure est la qualité des transactions (QdT).

2.2 Fonctionnement générale de l'architecture d'ordonnancement par rétroaction

L'architecture présentée dans la figure 1 permet d'adapter la qualité de service à la charge du système. Lorsque le système devient chargé, on diminue alors la qualité de service. En sens inverse, si le système est sous-utilisé alors on augmente la qualité de service. L'adaptation de la qualité de service est basée sur une boucle de rétroaction.

La boucle de rétroaction a pour tâche de stabiliser le système durant les phases de surcharge. Pour cela, elle s'appuie sur le principe d'observation puis d'auto-adaptation. L'auto-adaptation a lieu tout au long du fonctionnement du système car les demandes des utilisateurs sont imprévisibles et la charge doit être ajustée en permanence. L'observation consiste à prendre en compte l'état de fonctionnement du système et à déterminer s'il correspond aux paramètres de qualité de service initialement spécifiés. Cette observation se fait via le moniteur. À partir de l'observation effectuée, le système adapte ses paramètres, via le contrôleur de qualité de service, afin d'augmenter ou de diminuer le nombre de transactions acceptées dans le système. Le fonctionnement de la boucle de rétroaction doit faire tendre le système vers un état stable représenté par une valeur de référence fixée par le DBA[3] (par exemple, il peut s'agir d'un taux d'utilisation de 80%).

[2] Maximum Data Error.
[3] Database Administrator.

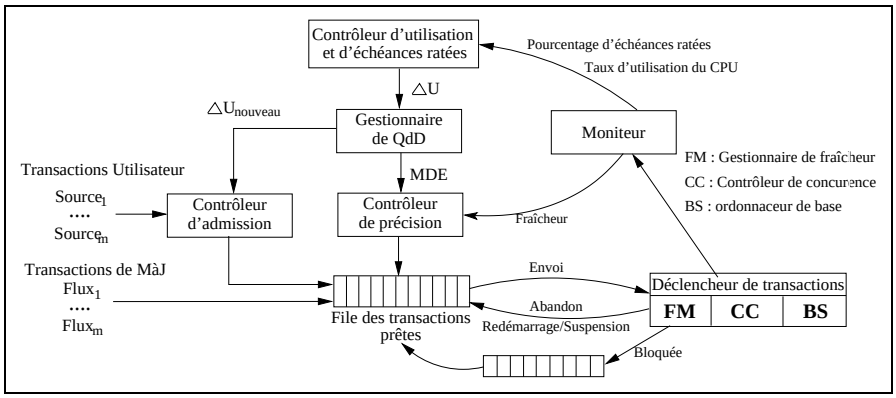

FIG.1. Architecture de contôle d'ordonnancement par rétroaction.

2.3 L'approche QdS dans les SGBD temps réel et les systèmes multimédia

Dans le tableau 1, nous avons effectué une comparaison entre la gestion de la qualité de service dans les SGBDTR et dans les systèmes multimédia. À partir du résultat de cette comparaison, nous pouvons appliquer les méthodes de gestion de QdS utilisées dans les SGBDTR aux systèmes multimédia.

	SGBD temps réel	Systèmes Multimédias
Mesures de performance	Nombre de transactions qui se terminent avant échéance	Nombre de paquets vidéo qui sont joués avant echéance
Qualité des transactions	Nombre de sous transactions optionnelles	Nombre de frames par seconde
Qualité des données	Différence entre la donnée réelle et la donnée stockée	Pas de qualité des données à gérer
Contrôleur d'admission	Limite le nombre de transactions « utilisateur » acceptées	Limite le nombre de demande

TAB.1. La gestion de la qualité de service dans les SGBDTR et les systèmes multimédias.

Dans les systèmes multimédia, que nous considérons, les données sont représentées par des paquets vidéo et ne sont pas mis à jour régulièrement. Par conséquent, aucune QdD ne peut être considérée au sens où cela est fait classiquement dans les SGBDTR. Par ailleurs, la transmission d'un groupe d'image (GoP) peut être comparée à l'exécution d'une transaction. Ainsi, dans un GoP, nous pouvons avoir recours à la diminution du nombre d'images transmises en supprimant celles qui sont optionelles. Lors des périodes de surcharge du système, cela nous permettra d'adapter la QdS, de la même façon que la QdT est adaptée dans les SGBDTR [4].

2.4 Architecture de contrôle par rétroaction appliquée aux systèmes multimédia distribués

Dans un précédent travail [10], nous nous appuyions sur les travaux de Natalia Dulgheru [11] et l'architecture QMPEGv2 [12] pour proposer une architecture de systèmes multimédia distribués (cf. figure 2). L'architecture de systèmes multimédia, que nous proposons, comporte trois entités principales :

1. Le *serveur maître :* il accepte les demandes des clients, désigne les serveurs vidéo devant servir la demande, surveille l'état du système et régule les flux vidéo de sorte que la QdS soit maintenue au meilleur niveau possible.
2. Les *serveurs vidéo :* ils envoient les flux vidéo aux clients et fonctionnent sous le contrôle du serveur maître.
3. Les *clients :* ils effectuent des requêtes auprès du serveur maître et reçoivent les flux vidéo en provenance des serveurs vidéo. Périodiquement, ils envoient un rapport de rétroaction au serveur maître (retour d'expérience sur la qualité de service reçue).

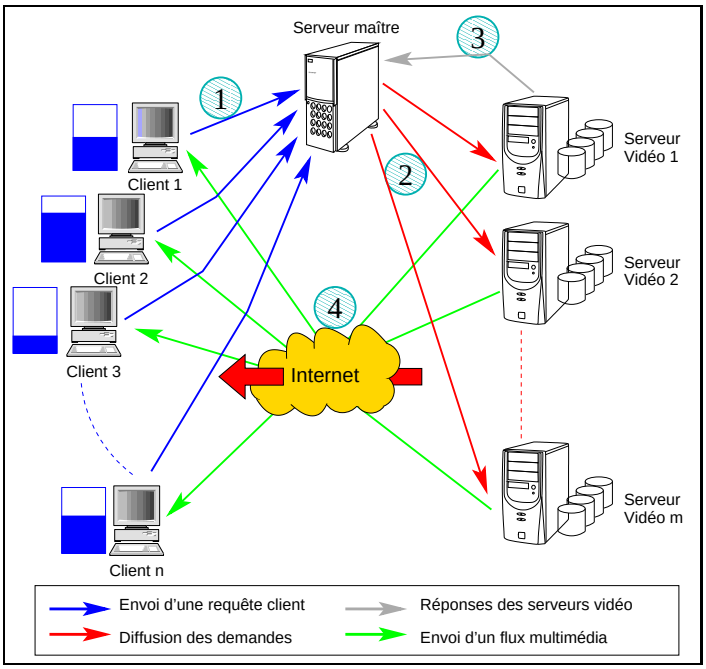

FIG.2. Scénario de fonctionnement de l'architecture.

Le fonctionnement typique de l'architecture, que nous proposons, basé sur une boucle de rétroaction, est le suivant :

1. Le client initie la demande d'une vidéo et envoie cette requête au serveur maître.

2. Le serveur maître diffuse la requête ”client” aux serveurs vidéo disponibles dans le système.
3. Les serveurs vidéo renvoient leur réponse au serveur maître qui en choisit un.
4. Une connexion est alors établie entre le client et le serveur vidéo, qui permet la diffusion du flux vidéo.
5. Périodiquement, le client envoit un rapport de rétroaction au serveur maître.
6. Si nécessaire, le serveur maître demande aux serveurs vidéo d'adapter leur qualité de service.

La boucle de rétroaction consiste ici à adapter la qualité de service en fonction des conditions de charge du système (les serveurs et la congestion du réseau). On observe la qualité de service perçue par le client et si nécessaire, on augmente ou on diminue la qualité de service coté serveur. Pour augmenter ou améliorer la qualité de service, on va augmenter ou diminuer le nombre de frames transmises. Pour cela nous nous appuyons sur les caractéristiques du standard MPEG [13] qui définit un mécanisme pour coder, lors de la compression, la vidéo. Il prend en entrée une séquence vidéo et la compresse selon trois types de frames : frames I (Intra), frames P (Predicted) et des frames B (Bidirectional). Les frames I sont des frames de références. Les frames P permettent de reconstruire une image à partir d'une frame I et les frames B utilisent des frames I et P pour reconstruire une séquence. Les frames I sont donc les plus critiques. Pour diminuer d'éventuelles congestions du réseau, il est nécessaire de supprimer des frames dans une séquence de frames mais ces suppressions doivent être effectuées de manière contrôlée. C'est pourquoi, nous proposons dans le paragraphe suivant une méthode de contrôle de la congestion du réseau.

2.5 Contrôle de la congestion du réseau

Les fluctuations du réseau en cas de surcharge pourraient dégrader sévèrement le niveau de service offert aux clients. En effet, le nombre de serveurs vidéo envoyant les paquets vidéo est inconnu. Par conséquence, le nombre de paquets transmis est également inconnu et celui-ci peut être important.

Une première solution, que nous proposons, consiste à effectuer un partage de la bande passante en fonction de la priorité des flux multimédia (exemple : journaliste / simple internaute). Une file d'attente est attribuée à chaque niveau de priorité. Mais, cette solution présente plusieurs inconvénients dans la garantie de la QdS pour tous les flux vidéo. En effet, un flux malveillant de priorité élevée peut perturber les autres flux dans la même file d'attente et ajouter un délai supplémentaire. Pour remédier à ces problèmes, nous proposons un contrôle du partage de la bande passante. Cette technique vise à gérer plus efficacement les ressources entre les différents flux vidéo. L'objectif est de s'assurer que les accès aux ressources du réseau se font d'une manière équitable et d'empêcher qu'un flux en rafale puisse consommer plus de ressources que la capacité maximale qui lui est alloué. Nous ajoutons aussi que, lorsqu'un grand nombre de paquets vidéo veulent accéder aux ressources du réseau, il nous faut garder un niveau de priorité pour les paquets vidéos qui restent plus prioritaires notamment en fonction du type de paquets (frame I, B et P).

Pour gérer la congestion du réseau et le choix des frames, nous nous appuyons sur le modèle (m,k)-firm qui est caractérisé par deux paramètres m et k. Plus formellement, une application est dite sous contrainte temps réel (m,k)-firm [14] si au moins m opérations parmi k opérations consécutives doivent respecter leurs échéances, c'est-à-dire terminer leur exécution avant expiration de leurs échéances. Dans le contexte des applications multimédia, les opérations sont présentés par les paquets vidéo.

Nous proposons une technique de gestion des paquets vidéo traversant le réseau. Celle-ci permet de décomposer en plusieurs classes les flux vidéo suivant leurs caractéristiques de tolérance aux pertes ou suivant leurs contraintes (m,k)-firm. Autrement dit, chaque classe contient les paquets vidéo de contraintes (m,k)-firm similaires. L'avantage de cette technique est de réaliser un compromis entre les ressources partagées en bande passante et la granularité de la QdS au sein d'une même classe de flux vidéo. En revanche, l'inconvénient de cette technique est que le respect de la contrainte (m,k)-firm d'une classe n'implique pas forcément le respect de cette contrainte pour chacun des flux vidéo appartenant à la même classe.

2.6 Stratégies de répartition des contenus en fonction de la charge des serveurs vidéo

Dans certains cas, un grand nombre de clients veulent accéder au même film. Un serveur vidéo ne peut diffuser que les films stockés sur ses disques. Si ce film n'est disponible que sur un seul serveur vidéo alors la probabilité que cette ressource devienne saturée augmente. Pour trouver une solution à ce type de problème, nous employons la stratégie de répartition suivante :

1. Lorsqu'un serveur vidéo est saturée, il envoie une requête à l'ensemble de ses confrères (les autres serveurs vidéo).
2. Trois réponses sont alors possibles :
   (a) Au moins un serveur vidéo répond qu'il possède la vidéo et qu'il est en mesure de traiter la requête (il n'est pas saturé).
   (b) Certains serveurs vidéo répondent qu'ils ne possèdent pas la vidéo et qu'ils ne sont pas en mesure de traiter la requête car ils sont saturés.
   (c) Certains serveurs vidéo répondent qu'ils ne possèdent pas la vidéo mais qu'ils seraient éventuellement en mesure de traiter la requête car ils ne sont pas saturés.
3. Dans les cas a et b, la stratégie de réplication n'entre pas en jeu.
4. Dans le cas c, on va chercher à répliquer la vidéo sur un serveur qui n'est pas saturé. On élit donc un serveur vidéo parmi ceux qui ont répondu qu'ils n'étaient pas saturés. Le choix du serveur vidéo est effectué en fonction de la QdS maximum qui peut être garantie par l'un des différents serveurs vidéo non saturés.

3 Simulations

Pour tester la faisabilité de notre architecture, nous avons eu recours à la conception d'un simulateur afin de vérifier le comportement du système et son adaptation par rapport aux variations de charge.

3.1 Présentation du simulateur

Ce simulateur (cf. figure 3) reprend les différents composants de l'architecture présentée dans la section 2. Un serveur maître est mis à disposition des serveurs vidéo participants à la diffusion des vidéos. Ainsi, ce service permet d'organiser l'ajout de serveurs en leur attribuant un numéro et en référençant les objets accessibles sur ceux-ci.

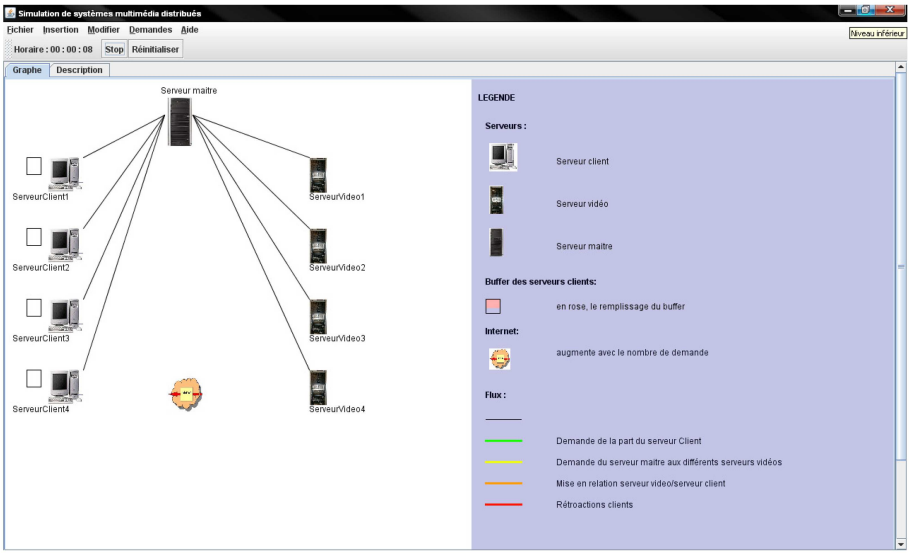

FIG.3. Le simulateur.

Après le démarrage du serveur maître, les serveurs vidéo qui souhaitent participer à la distribution de vidéos se référencent auprès de celui-ci et obtiennent ainsi un numéro.

Pour effectuer une requête, le programme client s'adresse au serveur maître, qui distribue la requête aux serveurs vidéo disponibles. Lorsque le serveur maître a désigné un serveur vidéo capable de répondre à la requête client — c'est à dire qu'il dispose du contenu demandé et qu'il est capable de fournir la QdS exigée — il envoie la référence du serveur vidéo au client. Ensuite, la diffusion en provenance du serveur vidéo et à destination du client peut commencer. Après un certain temps de diffusion, un retour sur la qualité de service reçue par le client est envoyé au serveur maître.

Ce simulateur a été réalisé en JAVA et une modélisation objet a permis de concevoir les trois grandes parties de l'architecture.

3.2 Objectifs des simulations

Les simulations ont pour objectif de montrer comment notre simulateur permet d'adapter la QdS des clients efficacement selon la charge courante dans le système. Notamment, le système doit s'adapter lorsque le nombre de clients qui effectuent des

requêtes varie dans le temps, faisant ainsi varier la charge du réseau. La congestion du réseau peut provenir de deux sources différentes :

- interne : un grand nombre de clients effectuant des requêtes dans notre système. On peut limiter ce nombre de clients par le biais du contrôleur d'admission localisé au niveau du serveur maître.
- externe : le réseau est utilisé par d'autres applications qui peuvent aussi provoquer la congestion. Notre architecture doit s'adapter en réduisant la qualité de service fournie aux différents clients tout en la maximisant par rapport à leurs exigences initiales.

3.3 Résultats

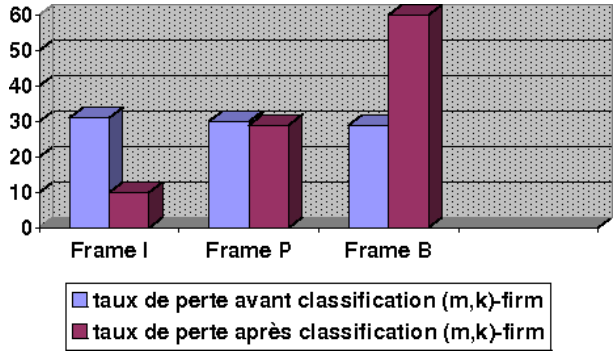

FIG.4. Le taux de perte de chaque type de paquet du flux MPEG avec et sans classification (m,k)-firm.

Nous remarquons sur la figure 4 que le taux de perte est moins important pour les frames I, considérés comme critiques. Nous distinguons ainsi, qu'avec la classification suivant les contraintes (m,k)-firm, il existe une différentiation claire entre les trois types de frames. Par conséquent, le rejet sélectif des frames est très efficace pour maintenir une QdS acceptable au client en minimisant la perte des frames critiques, c'est-à-dire les frames I.

4 Conclusion et perspectives

Dans ce travail, nous avons proposé une amélioration de l'architecture de contrôle par rétroaction pour les systèmes multimédia distribués. Notre objectif est de fournir une garantie temporelle déterministe selon les contraintes temporelles aux flux vidéo temps réel. Nos principales améliorations ont concerné l'utilisation d'une classification (m,k)-firm pour les paquets vidéo et la mise en place d'une stratégie de répartition des vidéos.

Une des extensions possibles de ce travail consiste à modifier l'architecture que nous avons présentée afin notamment d'apporter une certaine tolérance aux pannes car

le point faible de celle-ci réside dans la présence d'un seul serveur maître. Il s'agit également de présenter l'importance du partage équitable de la bande passante et de donner une certaine priorité aux paquets vidéo au niveau des ressources du réseau, de façon à augmenter sa fiabilité et sa robustesse et à converger vers la QdS désirée par les clients. La conception d'un simulateur nous a déjà permis de valider la faisabilité de notre approche et devrait nous permettre à terme de fournir des résultats démontrant l'apport réel de cette nouvelle approche. Une perspective à plus long terme consisterait à construire un véritable serveur de vidéos à la demande basé sur l'architecture que nous proposons.

Références

1. Duvallet, C., Mammeri, Z., Sadeg, B. : Les SGBD temps réel. Technique et Sciences Informatiques **18**(5) (1999) 479–517
2. Ramamritham, K. : Real-time databases. Journal of Distributed and Parallel Databases **1**(2) (1993) 199–226
3. Ramamritham, K., Son, S., DiPippo, L. : Real-Time Databases and Data Services. Real-Time Systems **28** (2004) 179–215
4. Amirijoo, M., Hansson, J., Son, S.H. : Specification and management of qos in real-time databases supporting imprecise computations. *IEEE Transactions* on *Computers* **55**(3) (2006) 304–319
5. Kang, K., Son, S., Stankovic, J. : Service Differentiation in Real-Time Main Memory Databases. In : 5^{th} IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (IEEE ISORC'02), Washington D.C (2002) 119–128
6. Kang, K., Son, S., Stankovic, J., Abdelzaher, T. : A QoS-Sensitive Approach For Timeliness and Freshness Guarantees in Real-Time Databases. In : Proceedings of the Euromicro Conference on Real-Time System. (2002) 203–212
7. Amirijoo, M., Hansson, J., Son, S. : Algorithms for Managing QoS for Real-Time Data Services Using Imprecise Computation. In : Proceedings of the International Conference on Real-Time and Embedded Computing Systems and Applications (RTCSA'2003), Taiwan (2003) 136–157
8. Lu, C. : Feedback Control Real-Time Scheduling. PhD thesis, University of Virginia (2001)
9. Liu, J., Shih, W.K., K.-J. Lin, R. Bettati, J.Y.C. : Imprecise Computations. In : Proceedings of the IEEE. Volume 82. (1994) 83–94
10. Zeddini, B., Duvallet, C., Sadeg, B. : Une approche qualité de service dans les systèmes multimédias distribués. Revue électronique des technologies de l'information - Special Issue of The 9^{th} MCSEAI'06 (4) (2007) 949–960
11. Dulgheru, N. : Management of QoS in Distributed MPEG Video. Master's thesis, University of Linköping (2004)
12. Ng, J., Leung, K., Wong, W. : Quality of Service for MPEG Video in Human Perspective. Technical report, Hong Kong Baptist University. (2000)
13. ISO/IEC 13818-2 : Information Technology-Generic Coding of Moving Pictures and Associated Audio, Part 2 : Video. Recommendation ITU H.262, International Standardization Organization (1994)
14. Hamdaoui, M., Ramanathan, P. : A Dynamic Priority Assignment Technique for Streams with (m,k)-$Firm$ Deadlines. IEEE Tansactions on Computers **44**(4) (1995) 1325–1337