ARCHITECTURE PAR RETROACTION POUR DES SYSTEMES MULTIMEDIA BASES SUR L'UTILISATION D'UN SERVEUR MAITRE

Claude DUVALLET & Bechir ALAYA

LITIS – Université du Havre
25 rue Philippe Lebon, BP 540
F-76058 LE HAVRE CEDEX
Prenom.Nom@litislab.fr

Résumé : Les applications multimédia gèrent des volumes importants 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 les architectures par rétroaction dans les SGBD temps réel afin de les appliquer aux systèmes multimédia distribués. Dans cet article, nous nous focalisons sur l'utilisation d'un serveur maître pour la mise en œuvre d'une architecture par rétroaction permettant de faire varier la qualité de service dans un système multimédia distribué.

Mots clés : systèmes multimédia distribués, qualité de service, architecture par rétroaction.

1 INTRODUCTION

Les progrès enregistrés dans le domaine des réseaux et l'amélioration constante des débits des réseaux 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) [Ram93] [DMS99] [RSD04].

Beaucoup d'applications multimédia distribuées doivent 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 grand nombre sur une période très courte (pic d'utilisation). Le besoin de bien gérer ce nombre important de données et d'offrir une bonne qualité de service aux utilisateurs déjà servis est primordial. L'étude de la gestion des données multimédia et de la qualité de service (QdS) particulièrement la QdS des flux vidéo permet de répondre à ces nouveaux besoins en adaptant les techniques existantes pour un transfert plus fiable et plus efficace des 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. L'objectif est d'assurer une gestion efficace des flux de données multimédia et une qualité de service acceptable en respectant les exigences multiples des différents flux vidéo.

Des travaux concernant la gestion des données volumineuses et la qualité de service dans les SGBDTR ont déjà été effectués [AHS06] [KSS02]. Tous ces travaux sont basés sur des architectures à retour d'expérience (ou *par rétroaction* ou *Feedback Control Scheduling Architecture (FCSA)*) qui contrôlent le comportement du système au moyen d'une boucle de rétroaction. Dans un premier temps, on mesure les performances du système afin de détecter les phases de surcharge. Puis, dans un second temps, on modifie les paramètres de fonctionnement du système afin de les adapter aux conditions réelles de charge observées. Ces conditions étant très changeantes, il faut répéter indéfiniment l'opération.

En raison de similarités existantes entre les SGBDTR et 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 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 ou que le réseau devient congestionné. Ces travaux s'appliquent particulièrement aux applications de vidéo à la demande (VoD), mais aussi à d'autres applications.

Dans cet article, nous présentons une architecture permettant de tenir compte de la congestion du réseau pour augmenter la qualité de service offerte aux utilisateurs. Dans la section 2, nous présentons brièvement le contexte et les travaux relatifs à la gestion de la qualité de service dans les SGBD temps réel. La section 3 décrit notre approche pour la gestion de la qualité de service dans les systèmes multimédia distribués. Dans la section 4, nous nous focalisons sur l'architecture du serveur maître qui est un des principaux composants de l'architecture par rétroaction que nous utilisons dans nos travaux. Enfin, nous concluons cet article par un bilan sur ce travail et les perspectives que nous comptons lui donner.

2 GESTION DE LA QUALITE DE SERVICE DANS LES SGBD TEMPS REEL

Dans une application reposant sur l'utilisation de SGBDTR, des transactions en provenance des utilisateurs arrivent à des fréquences variables. Lorsque la fréquence augmente de façon considérable, l'équilibre du SGBDTR est mis en péril à cause des surcharges. Durant ces périodes de surcharge, le SGBDTR va potentiellement manquer de ressources, et un grand nombre de transactions temps réel vont manquer leur échéance. Des travaux basés sur l'approche QdS tentent de rendre les SGBDTR plus robustes face à ces périodes d'instabilité (périodes de sous-utilisation et périodes de surcharge) [KSS02] [RSD04] [AHS06]. Ces travaux s'appuient sur des techniques de contrôle avec rétroaction [Lu01] et autorisent la manipulation de résultats imprécis [LSKJL94].

La gestion de la qualité de service dans les SGBDTR repose souvent sur l'exploitation d'une boucle de rétroaction [Lu01] permettant d'adapter la qualité des données (temps réel) et des transactions (temps réel) [AHS06] en fonction de l'état de charge du système.

2.1 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 la qualité de service. Par contre, 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.

Nous allons donner une brève description de chacun des composants du modèle utilisé pour la gestion de la qualité de service dans les SGBDTR. La tâche du contrôleur d'admission est de contrôler les transactions "utilisateur" qui sont acceptées ou pas dans le système. Il effectue ce contrôle en fonction de la charge d'utilisation calculée et des paramètres de qualité de service spécifiés par le DBA[1] [AHS06]. Son fonctionnement est contrôlé par la boucle de rétroaction qui lui fournit ses paramètres de fonctionnement.

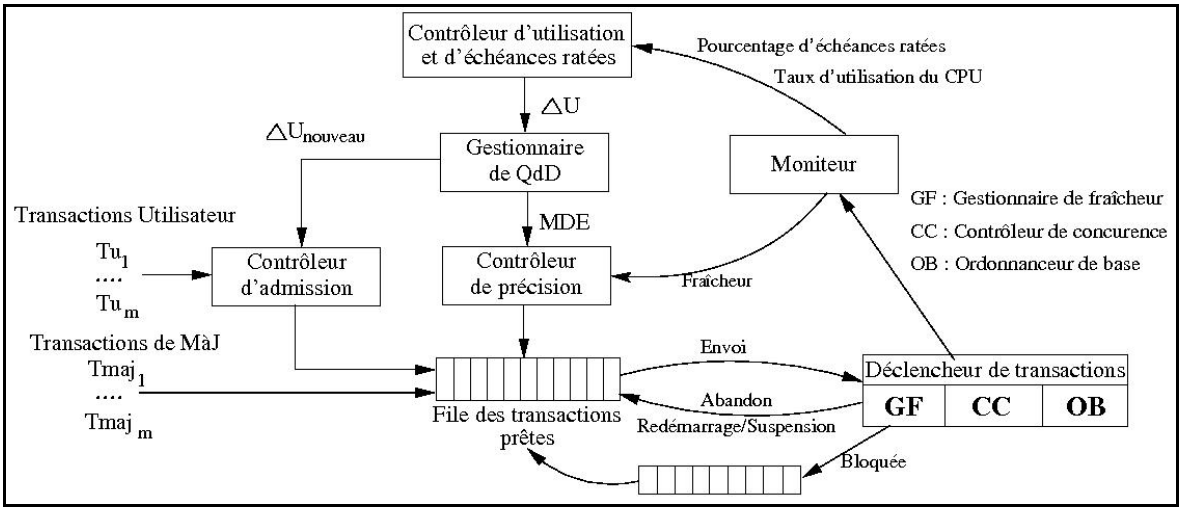

Figure 1 : Architecture de SGBDTR basée sur la rétroaction.

Les transactions qui sont admises dans le système sont placées dans une file d'attente avant d'être envoyées au déclencheur de transactions. Celui-ci a pour fonction de gérer l'exécution des transactions. Il dispose de plusieurs modules complémentaires :

- Un *gestionnaire de fraîcheur* : il vérifie la fraîcheur des données qui vont être accédées par une transaction. Si les données sont obsolètes (non fraîches) alors la transaction est mise en attente dans une file.
- Un *contrôleur de concurrence* : il est chargé de gérer les conflits d'accès aux données qui apparaissent entre les transactions. Dans la plupart des travaux [AHS06], il s'agit du protocole 2PL-HP (Two Phase Locking High Priority) [Ram93].
- Un *ordonnanceur de base* : il s'agit souvent du protocole EDF (Earliest Deadline First) [SLL92] qui ordonnance les transactions selon le principe que la transaction qui possède l'échéance la plus proche doit être exécutée en priorité.

Les différents modules du *déclencheur de transactions* peuvent être remplacés par des modules équivalents. C'est ainsi que l'on peut modifier la politique de *contrôle de concurrence* ou modifier la *politique d'ordonnancement*. Un *moniteur* permet de mesurer les performances du système en inspectant l'exécution des transactions (quantité de transactions terminées, abandonnées, qui ont ratées leur échéance,...). Les valeurs ainsi mesurées permettent d'alimenter le *contrôleur de qualité de service* et font parties de la boucle de contrôle par rétroaction qui va contribuer à stabiliser le système.

Le *contrôleur d'utilisation et d'échéances ratées* (ou *contrôleur de qualité de service*) permet de réajuster les paramètres de QdS en fonction des valeurs déterminées par le moniteur et des paramètres de référence (fournis par le DBA). Les valeurs obtenues sont transmises au *contrôleur d'admission* et au *gestionnaire de qualité des données*. Le *gestionnaire de qualité des données* permet de réajuster la valeur du paramètre MDE[2] qui constitue le paramètre de

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

QdD. La valeur de MDE est calculée en fonction de l'utilisation du système. Le paramètre MDE est ensuite transmis au contrôleur de précision qui écarte les transactions de mise à jour lorsque les données à mettre à jour sont suffisamment représentatives du monde réel en considération de la valeur de MDE.

3 GESTION DE LA QUALITE DE SERVICE DANS LES SYSTEMES MULTIMEDIA DISTRIBUES

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 inhérentes 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 [KSSA02] [AHS06] et à les adapter aux systèmes multimédia. Pour cela, nous proposons une adaptation de la méthode de contrôle par rétroaction pour les systèmes multimédia distribués. Nous avons appelé cette architecture FCA-DMS (Feedback Control Architecture for Distributed Multimedia Systems). Nous allons appliquer un contrôle de la congestion du réseau et un partage équitable de la bande passante afin d'améliorer la qualité de service offerte aux utilisateurs.

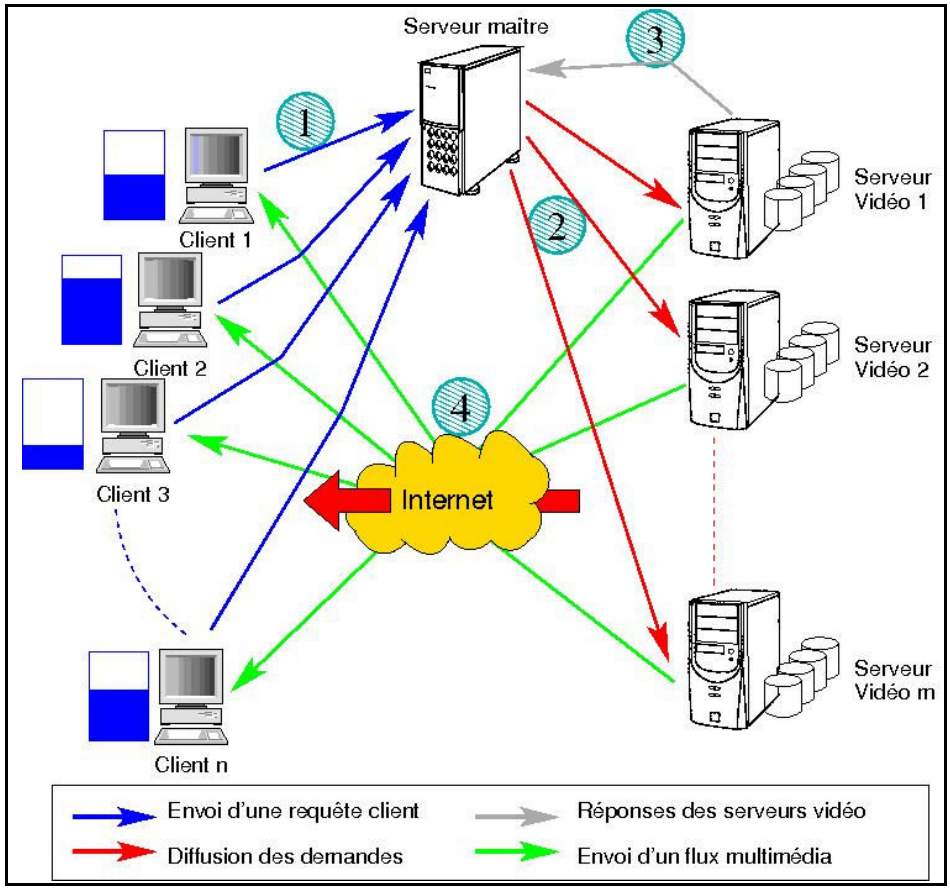

Figure 2 : Scénario de fonctionnement de l'architecture.

3.1 Présentation de l'architecture de rétroaction

L'architecture de systèmes multimédia distribués (cf. Figure 2), 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 fonctionnent sous le contrôle du serveur maître et ils envoient les signaux vidéo aux clients.
3. Les *clients* : ils effectuent des requêtes auprès du serveur maître et reçoivent les flux vidéo en provenance des serveurs vidéo. Périodiquement, ils envoient la mesure de la QdS reçue au serveur maître.

Le fonctionnement typique de l'architecture, basé sur une boucle de rétroaction, que nous proposons 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 envoie un rapport de rétroaction au serveur maître.
6. Si nécessaire, le serveur maître demande aux serveurs vidéo d'adapter leur qualité de service.

3.2 Dégradation de la qualité de service des flux MPEG

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 [ISO94]. 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.

Le standard MPEG 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), frames P (Predicted frames) et frames B (Bidirectional frames). Chaque image d'entrée est compressée selon un des trois types de frames mentionnés ci-dessus. Les frames I peuvent être considérées comme des frames de référence ; elles sont indépendantes des autres types de frames (qui les suivent ou qui les précédent). Les frames P et B ne sont pas indépendantes ; elles indiquent les différences relatives par rapport aux frames de référence. Plus précisément, une frame P spécifie les différences par rapport à la frame I précédente, alors qu'une frame B donne une interpolation entre les frames précédentes et suivantes de types I ou P.

Dans nos travaux, lorsqu'il est nécessaire de modifier la qualité de service d'une vidéo, nous commençons par supprimer des frames B car ce sont les moins importantes puis s'il faut encore réduire la qualité et que le nombre de frame B est nulle alors nous supprimons des frames P. Généralement, nous gardons les frames I car cela reviendrait à aboutir à une qualité de service trop faible.

4 ARCHITECTURE DU SERVEUR MAITRE

Le serveur maître permet de faire le lien entre les clients et les serveurs vidéo. Il reçoit les demandes de contenus de la part des clients puis les redirige en fonction de leur disponibilité.

Dans la figure 4, nous détaillons le principe de fonctionnement de l'architecture du serveur maître.

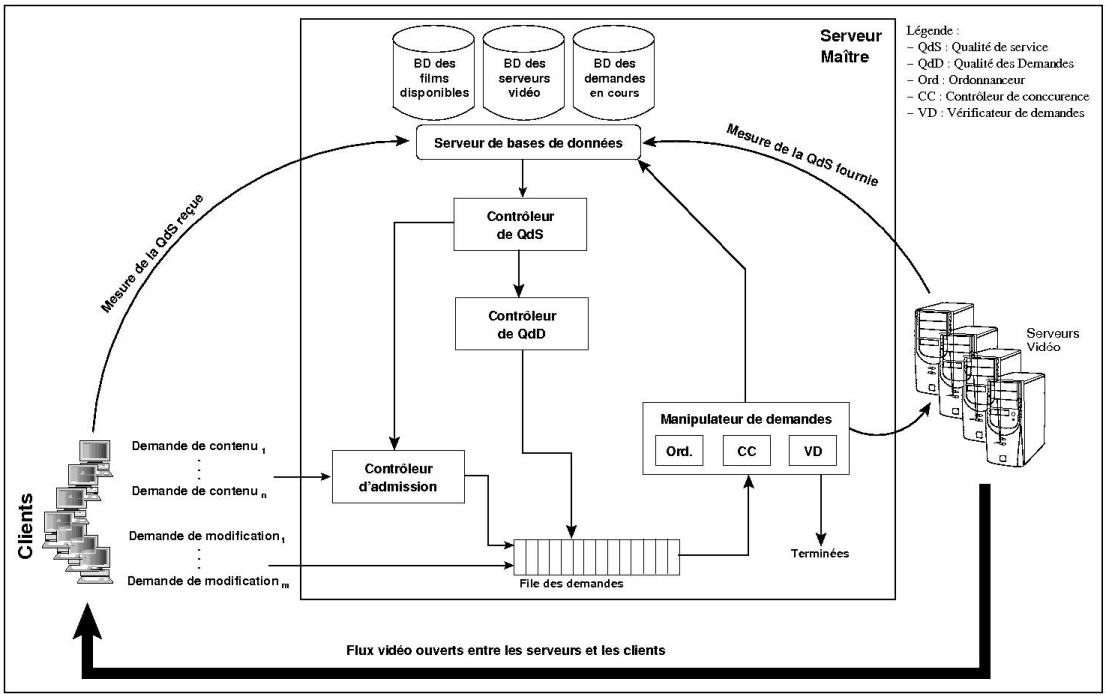

Figure 4 : Architecture du serveur maître.

4.1 Les bases de données

Le serveur maître comporte un serveur de bases de données qui gèrent trois bases de données différentes :

- ***Base des films disponibles*** : Elle permet de maintenir à jour une liste de tous les films disponibles dans le système avec leur emplacement. Un film est éventuellement présent sur plusieurs serveurs vidéo. Le serveur maître collecte des informations à partir des serveurs vidéo.
- ***Base des serveurs vidéo disponibles*** : Elle permet de maintenir à jour une liste des serveurs vidéo disponibles. Lorsqu'un serveur vidéo se connecte, il en informe le serveur maître. Au cours du fonctionnement du système, le serveur maître vérifie la disponibilité des serveurs vidéo afin de mettre à jour la base de données. Cette base possède toutes les caractéristiques d'un serveur vidéo (capacité en bande passante, disque dure, etc.).
- ***Base des demandes en cours de traitement*** : Lorsqu'une demande de vidéo a été lancée alors on garde une trace dans cette base de données et on la met régulièrement à jour avec les données mesurées au niveau du serveur vidéo qui traitent la demande ainsi que celles mesurées au niveau du client qui reçoit le flux.

4.2 Les demandes

Dans l'architecture du serveur maître, on considère deux types de demandes qui vont être traitées :

- ***Demandes de films*** : Lorsqu'un client souhaite visualiser un nouveau film, il envoie une demande au serveur maître qui va décider s'il l'accepte ou non en fonction de la charge. Cette demande est transmise au contrôleur d'admission qui décide de son admission ou non dans la file des demandes à traiter.

- ***Demandes de modification de la qualité de service*** : Après l'ouverture d'un flux vidéo, de façon périodique des mesures, de la qualité de service coté client sont effectuées. Si cette qualité de service est différente de celle qu'attendait le client alors une demande est envoyée au serveur maître. Ces demandes sont envoyées de façon périodique dans la file des demandes à traiter.

4.3 Les composants

Nous allons décrire dans ce paragraphe l'ensemble des composants présents dans l'architecture du serveur maître (cf. Figure 4) :

- Le ***manipulateur de demandes*** permet de traiter toutes les demandes présentes dans la file des demandes. Pour choisir la prochaine demande qui devra être traitée, nous utilisons un ordonnanceur qui est basé sur la date de génération des demandes. Lorsque plusieurs demandes entrent en concurrence pour une même ressource qui n'est pas suffisamment disponible alors un contrôleur de concurrence permet de déterminer la demande qui sera traitée de façon prioritaire et celle qui sera abandonnée ou suspendue. Avant tout traitement d'une nouvelle demande, un vérificateur de demandes aura pour rôle de vérifier la disponibilité du contenu demandé avant de la traiter. Si le contenu n'est pas disponible alors la demande sera mise en attente et pourra être réinsérée plus tard dans la file des demandes à traiter.
- Le ***contrôleur d'admission*** permet d'écarter certaines demandes lorsque la charge globale du système ne permet pas de traiter toutes les demandes admises.
- Le ***contrôleur de qualité de service*** exploite les mesures effectuées du coté serveur et du coté client afin d'effectuer une mesure de la charge globale du système et du réseau. Connaissant le nombre de frames envoyées pour chaque flux vidéo et le nombre de frames reçues, nous pouvons estimer la capacité globale du réseau. À partir de cette capacité, nous pouvons déterminer s'il est possible d'augmenter la charge du système en acceptant plus de demandes ou si au contraire il faut réduire le nombre de demandes acceptées dans le système. Les demandes que nous considérons sont à la fois les demandes de contenu et les demandes de modification de la qualité de service des demandes en cours. Pour effectuer cela, le *contrôleur de qualité de service* transmet des informations au *contrôleur d'admission* et *au contrôleur de qualité des demandes de modification de la qualité de service*.
- Le ***contrôleur des demandes de modification de la qualité de service*** permet d'écarter ou non des demandes de ce type afin de diminuer la charge du système en évitant de traiter des demandes qui ne consiste qu'en une modification minime de la QdS. Pour déterminer le seuil au delà duquel la modification doit être prise en compte, nous utilisons une valeur transmise par le contrôleur de qualité de service.

4.4 Bilan

Le serveur maître utilise donc un manipulateur de demandes pour traiter les demandes de films ainsi que les demandes de modification de la qualité de services. La régulation des demandes est effectuée au moyen d'un contrôleur de qualité de qualité de service. Lorsque la charge du système augmente alors le nombre de nouvelles demandes admises va être réduit afin de garantir une certaine qualité aux demandes déjà admises. Durant ces mêmes périodes, les exigences sur l'amélioration de la qualité de service seront amoindries.

5 CONCLUSION ET PERSPECTIVES

Dans cet article, nous avons proposé une amélioration de l'architecture de contrôle par rétroaction pour les systèmes multimédia distribués (FCA-DMS). Notre objectif est de fournir

une garantie de la qualité de service fournie aux clients. Notre principale contribution, pour cet article, réside dans la présentation détaillée de l'architecture du serveur maître avec les différents éléments qui le composent. Des simulations présentées dans [ADS09] ont montré l'apport de notre architecture.

Une des extensions possibles à nos travaux 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. Une perspective à plus long terme consiste à construire un véritable serveur de vidéos à la demande basé sur l'architecture que nous proposons.

6 BIBLIOGRAPHIE

[ADS09] B. Alaya, C. Duvallet, B. Sadeg. « A New Approach to Manage QoS in Distributed Multimedia Systems ». J. of Computer Science and Information Security, 2(1): 1-10, 2009.

[AHS06] M. Amirijoo, J. Hansson, and S. H. Son. « Specification and management of QoS in real-time databases supporting imprecise computations », IEEE Transactions on Computers, 55(3):304-319, 2006.

[DMS99] C. Duvallet, Z. Mammeri, and B. Sadeg. « Les SGBD temps réel », Technique et Sciences Informatiques, 18(5): 479-517, 1999.

[ISO94] ISO/IEC 13818-2. « Information Technology-Generic Coding of Moving Pictures and Associated Audio, Part 2: Video », Recommendation ITU H.262, International Standardization Organization, November 1994.

[KSS02] K-D. Kang, S.H. Son, and J.A. Stankovic. « Service Differentiation in Real-Time Main Memory Databases », In 5th IEEE International Symposium on Object-Oriented Real-time Distributed Computing (IEEE ISORC'02), Washington D.C, 2002.

[KSSA02] K. Kang, S.H. Son, J.A. Stankovic and T.F. Abdelzaher. « A QoS-Sensitive Approach For Timeliness and Freshness Guarantees in Real-Time Databases », In Proceedings of the Euromicro Conference on Real-Time System, pages 203-212, 2002.

[LSKJL94] J.W.S. Liu, W.-K. Shih, and J.-Y. Chung K.-J. Lin, R. Bettati. « Imprecise Computations », In Proceedings of the IEEE, volume 82, pages 83-94, January 1994.

[Lu01] C. Lu. Feedback « Control Real-Time Scheduling », PhD thesis, University of Virginia, 2001.

[NLW00] J. Ng, K. Leung, and W. Wong. « Quality of Service for MPEG Video in Human Perspective », Technical report, Hong Kong Baptist University., 2000.

[Ram93] K. Ramamritham. « Real-Time Databases », Journal of Distributed and Parallel Databases, 1(2):199-226, 1993.

[RSD04] K. Ramamritham, S.H. Son, and L.C. DiPippo. « Real-Time Databases and Data Services », Journal of Real-Time Systems, 28: 179-215, 2004.

[SLL92] S.H. Son, J. Lee, and Y. Lin. « Hybrid protocols using dynamic adjustment of serialization order for real-time concurrency control », Journal of Real-Time System, 4(3): 269-276, 1992.