Gestion de la Qualité de Service par Reconfiguration Dynamique dans les Applications Interactives Multimédia

Philippe Roose, Marc Dalmau, Franck Luthon, Sophie Laplace

LIUPPA/IUT Informatique de Bayonne
Château Neuf, Place Paul Bert
64100 Bayonne, France
{roose|luthon@iutbayonne.univ-pau.fr}, dalmau@ieee.org

RÉSUMÉ. *Nos travaux concernent essentiellement le domaine des Applications Multimédia Distribuées (AMD). Nous travaillons plus précisément sur des applications multimédia conçues et réalisées à l'aide de composants logiciels et/ou matériels.*

Dans ce papier, nous nous penchons sur le problème de la qualité de service (QdS) dans les AMD. Nous ne nous limitons pas aux problèmes liés aux seuls aspects réseau Internet mais, nous nous intéressons également à la flexibilité de ce type d'applications. La solution que nous préconisons consiste à organiser les composants constituant l'application en groupes et sous-groupes de travail.

Les groupes de travail permettront d'ajuster les services offerts aux utilisateurs alors que les sous-groupes de travail représenteront plus spécifiquement la façon de réaliser ces services.

Afin de gérer les composants, les groupes, les sous-groupes, mais également les flux de données, nous utilisons une plate-forme spécifique adaptée utilisant des règles de type ÉCA (Événement-Condition-Action). L'implémentation est réalisée à l'aide de Java (Beans Java) et de JMF (Java Multimedia Framework).

Dans cet article, nous allons présenter un exemple de réalisation d'une application de visioconférence. Nous verrons comment à l'aide de règles nous pourrons gérer la dynamique des groupes et sous groupes de travail, mais également permettre la gestion de la QdS ainsi que traiter les problèmes d'interopérabilité entre les composants.

MOTS-CLÉS : *Qualité de Service, Composants, Applications Multimédia Distribuées, Interopérabilité.*

KEYWORDS: *Quality of Service, Components, Distributed Multimedia Applications, Interoperability.*

1. Introduction

La conception et la réalisation d'applications multimédia distribuées présente plusieurs points communs avec nos précédents travaux portant sur la ré-ingénierie d'applications pour la mise en œuvre de coopération [1]. Actuellement, les applications de ce type sont généralement implémentées en utilisant des composants distribués. Le problème est que la plupart ne permettent pas la gestion de la qualité de service nécessaire à ce type d'applications, à savoir la QdS au niveau utilisateur mais également au niveau environnement d'exécution.

L'un des points critiques des AMD est le fort besoin de flexibilité afin de fournir la meilleure qualité de service possible à chaque instant. L'interactivité inhérente à ces applications implique une adaptation rapide aux besoins des utilisateurs. De plus, étant distribuées, elles doivent gérer en permanence la QdS du réseau qui varie fréquemment. Afin de gérer ceci, nous proposons d'organiser ces applications autour de groupes de travail dynamiques (groupes de composants évoluant dans le temps). La constitution et la circulation des informations entre et dans ces groupes de travail doit se conformer à un certain nombre de contraintes : les utilisateurs, les services, les débits, les temps de réponse, etc.

Les composants formant une application multimédia distribuée seront organisés en groupes et sous-groupes de travail en respectant les caractéristiques suivantes [6] :

- *Composants*: Objets ou programmes réalisant une tâche spécifique et réalisés de manière à facilement inter opérer avec les autres composants et applications.
- *Groupes de travail* : Ils sont caractérisés par le service qu'ils fournissent à l'utilisateur. Par conséquent, un groupe de travail est attaché à chaque utilisateur. Son but est de fournir à l'utilisateur la vue désirée de l'application. C'est par l'inclusion/exclusion de sous-groupes dans un groupe que l'applications est rendue flexible puisque cela permet de répondre au mieux aux besoins de l'utilisateur.
- *Sous-groupes de travail* : Ils sont composés de composants coopérant dans le but de réaliser une tâche commune. Ils peuvent être considérés comme des "super-composants". Ils sont caractérisés par leur rôle ainsi que par la QdS qu'ils offrent. Afin de s'adapter aux contraintes externes, nous devons inclure dans ces sous-groupes des composants pertinents relativement à la situation présente. Leur dynamique permet à l'utilisateur d'obtenir, pour un même but, une QdS moins bonne mais plus rapide lorsque, par exemple, les performances du réseau ne permettent pas d'obtenir une qualité optimale.

De par l'organisation en groupes et sous-groupes, il est nécessaire de disposer d'une plate-forme spécifique pouvant gérer les flots d'informations entre les entités. De même, la gestion des groupes de travail sera assurée par des règles intégrées à cette plate-forme [2]. Ces règles permettront la reconfiguration dynamique de l'application relativement à la QdS qu'elles évaluent. La plate-forme reçoit l'ensemble des données de l'application et, en fonction de la configuration actuelle,

elle les diffusera aux divers groupes/sous-groupes et, par conséquent, aux composants et aux règles concernés.

Les règles sont utilisées par la plate-forme pour évaluer et adapter les performances des composants relativement à l'environnement. Elles sont conformes au modèle ÉCA [7] - Événement-Condition-Action - et sont entièrement gérées par la plate-forme. Il existe deux type de règles :

- *Les règles d'interopérabilité* qui créent ou adaptent les flots de données entre les composants qui n'ont pas forcément été conçus pour travailler ensemble.
- *Les règles de supervision* qui réorganisent la composition des groupes et sous-groupes.

Ce papier est organisé comme suit : la première partie donne un exemple d'utilisation d'une règle d'interopérabilité simple. Ensuite, le même exemple est repris afin de montrer l'utilisation de règles de supervision utilisant la mesure de la QdS pour la gestion des groupes de travail.

2. Règles d'interopérabilité

Nous montrons ici (cf. figure 1) un exemple d'application de visioconférence via Internet. Dans un premier temps, nous partons de l'hypothèse que le débit est suffisant pour fournir à la fois la vidéo et le son. Si nous recevons un flux vidéo compressé avec RealVideo, dans une fenêtre de 160x120 à 10 images/seconde et avec un son de qualité téléphonique, un débit de 56Kbits est suffisant.

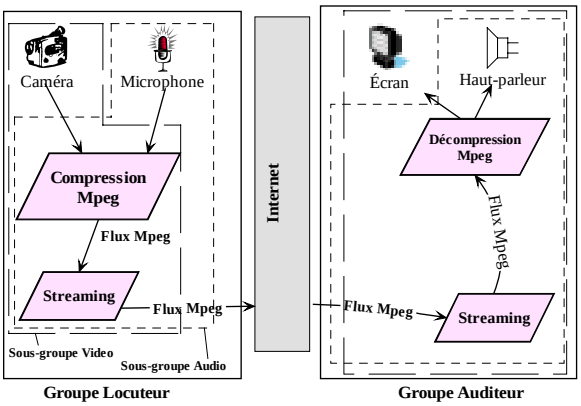

Figure 1 : débit élevé

Le problème est que si la bufferisation des données permet de compenser de brèves variations de débit, elle n'est pas suffisante pour supporter un décroissement durable du débit du réseau. Cela signifie que la diffusion/réception de la conférence peut momentanément être stoppée. Chacun sait que la vidéo est grosse consommatrice de bande passante, aussi, si nous voulons tout de même conserver le

support vidéo, nous pouvons par exemple nous contenter de remplacer ce flux vidéo par un ensemble de commandes pilotant un clone (ou ersatz) représentant les expressions faciales du locuteur (cf. figure 2). Ces changements suggèrent l'addition d'un composant d'analyse de visage qui identifie les points caractéristiques [3] (ou points anthropomorphiques) qui permettront ensuite de générer les commandes du clone. Bien sûr, l'utilisation d'un clone ne vaut pas l'image réelle à proprement parler, mais néanmoins, elle permettra de refléter les expressions du visage afin de suivre malgré tout visuellement la visioconférence et non pas uniquement par le biais du son.

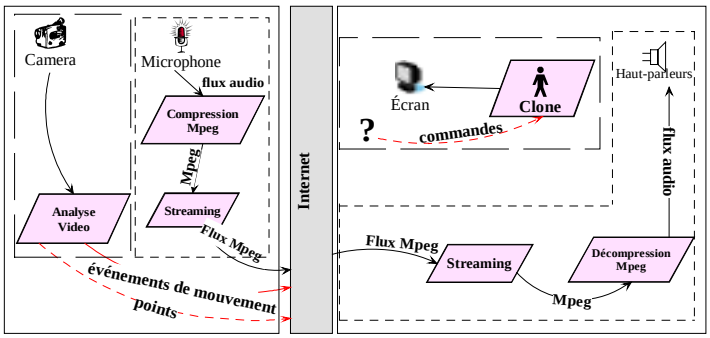

Figure 2 : faible débit

Comme nous pouvons le voir sur la figure 2, la vidéo n'est plus diffusée mais est remplacée par les points caractéristiques permettant au clone de s'animer [4] et de refléter l'expression du visage du locuteur. Ceci s'inspire des FAPs (*Facial Animation Parameters*) et des TDP (*Thermal Design Power*) de la norme MPEG 4 [8]. Avec cette solution, nous ne transmettons que le son (avec une qualité téléphonique nécessitant 13Kbps) et les points caractéristiques (20 points sont suffisants). Avec un débit de 10 images/seconde, pour commander le clone, nous utiliserons approximativement 2Kbps. Cette solution nécessite alors moins de trois fois le débit du premier cas.

Bien sûr, la qualité de la vidéo n'est pas comparable avec la première version mais cela permet à la vidéoconférence d'être poursuivie correctement et en temps réel. Néanmoins, la mise en œuvre de cette solution soulève quelques questions. Dans le premier cas, les flux vidéo et audio sont mélangés et synchronisés à la source. Dans le cas d'un faible débit, la vidéo n'est plus diffusée sur le réseau, seuls les points caractéristiques et les événements de mouvement le sont. Ainsi, il est impossible de produire un flux audio/vidéo synchronisé. De plus, à cause de la diffusion via Internet, les flots de données (les événements de mouvements, les points caractéristiques et le flux audio) ne sont plus synchrones. Nous devons re-synchroniser ces deux sources de données à leur réception. La solution que nous préconisons est d'utiliser une règle d'interopérabilité.

Elle recevra les événements concernant les mouvements, les points caractéristiques ainsi que le flux audio Mpeg (produit à la source) mais également ce

même flux une fois décompressé. En utilisant des algorithmes de traitement du signal, elle va détecter les événements caractéristiques à l'intérieur du son (silences, …) qu'elle va corréler avec les points caractéristiques et les événements de synchronisation issus du composant d'analyse de visage pour produire les commandes nécessaires à l'animation du clone de manière synchrone avec le son (cf. figure 3).

Si les deux flux sonores (la version Mpeg compressée et la version décompressée) sont nécessaires c'est que dans le premier nous détectons les événements de synchronisation alors que le second nous permet de connaître le moment où il faut produire la commande correspondante (il existe un délai non négligeable dû en grande partie au composant de bufferisation).

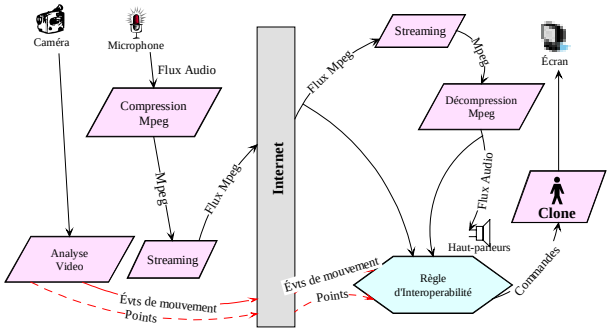

Figure 3 : Ajout d'une règle d'interopérabilité

Nous avons proposé une solution pour gérer le problème d'interopérabilité mais nous n'avons pas encore expliqué comment détecter le moment où nous devons utiliser l'un ou l'autre groupe de composants en fonction des deux contextes : le cas idéal et le cas dégradé.

3. Règles de supervision distribuée

Entre les figure 1 et 2, de nouveaux composants ont été ajoutés et les flux de données ont été réorganisés. Ceci ne peut être réalisé par les composants eux mêmes puisqu'ils n'ont pas été conçus pour cela et n'ont pas connaissance du contexte d'exécution de l'application. Ce rôle est alloué à la plate-forme qui doit réorganiser l'application et, qui plus est, au bon moment (par exemple entre deux phrases, et pas en plein milieu d'une). Ainsi, il est nécessaire d'avoir un mécanisme de mesure du flux qui détectera quand l'application doit être reconfigurée pour des raisons de baisse du débit disponible. De plus, il est nécessaire de savoir quand la bande passante sera suffisante afin de basculer dans l'autre configuration. Nous proposons pour cela d'utiliser des règles de supervision. Leur rôle sera de mesurer le débit du réseau afin de détecter un besoin de reconfiguration, et, le cas échéant, de dynamiquement modifier la composition des groupes pour proposer la meilleure

qualité possible. Nous pouvons voir ces différentes règles dans les figures suivantes (figures 4 et 5) :

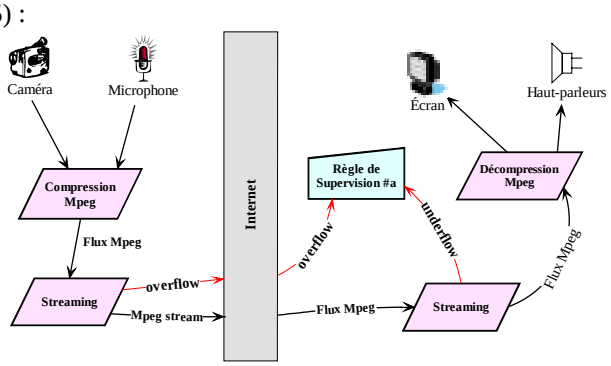

Figure 4 : Ajout d'une règle de supervision

Afin de détecter une baisse du débit utile, la règle de supervision (#a) reçoit deux événements du composant de bufferisation correspondant à un sous ou sur-remplissage du buffer (*underflow* ou *overflow*). Ils correspondent respectivement à 20% ou 80% d'occupation du buffer de streaming. Lorsqu'un événement d'underflow est reçu, il révèle un problème lié à la transmission (perte d'information, accroissement du temps de transmission). Au contraire, un événement d'overflow révèle que le débit actuel du réseau n'est pas suffisant (cf figure 4).
De même, lorsque nous sommes en configuration adaptée au bas débit, nous devons détecter quand l'application peut retourner à son état de configuration original.
Ceci ne peut être réalisé que lorsque le débit du réseau peut supporter l'ajout du flux vidéo. Afin d'éviter trop de changements dans la configuration, l'application ne peut basculer dans la configuration idéale que si le flot de donnée vidéo peut être supporté pendant un temps suffisant (en fonction de la taille du buffer). Ce rôle est dévolu à plusieurs règles de supervision (cf. figure 5).

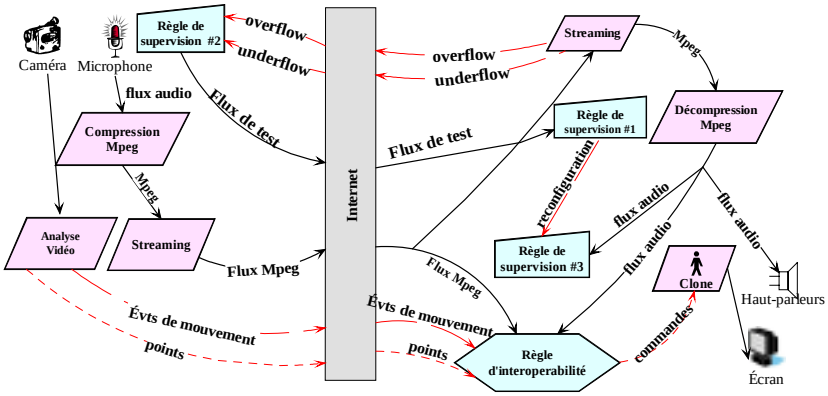

Figure 5 : Ajout de règles de supervision et d'interopérabilité dans le cas de bas débit

La règle #2 produit un flux de données de test via le réseau vers la règle #1. Ce flux de données correspond en volume à un flux vidéo, il est contrôlé par la règle #2 en utilisant les événements d'overflow et d'underflow produits par les composants de streaming. Tant qu'aucun événement d'overflow n'est apparu, le flux de test continue. Une fois stoppé, il ne peut être redémarré que lors d'un événement d'underflow. La règle #1 reçoit ce flux et produit un événement de reconfiguration lorsqu'il a été correctement reçu pendant un laps de temps significatif.
Cet événement permet de savoir que l'application est capable de revenir dans sa configuration initiale (idéale). Il sera reçu par une autre règle de supervision (#3) qui gère la reconfiguration des sous groupes de l'application. Elle ne doit pas uniquement tenir compte des événements de reconfiguration mais également de l'état de l'application. Par exemple, l'application ne doit pas être reconfigurée automatiquement alors que le locuteur est en train de parler. Ainsi, la règle de supervision va aussi analyser le flux audio décompressé pour détecter les silences. Ainsi elle pourra reconfigurer l'application au bon moment en changeant la composition des sous-groupes [5].

4. Conclusion

Le développement des architectures distribuées et des réseaux, la modélisation à base de composants (UML) et les outils de réalité virtuelle couplés à la maturité des algorithmes de traitement de sons et d'images nous permettent d'imaginer de nouvelles approches et fonctionnalités pour les applications distribuées multimédia (apprentissage à distance, vidéoconférence, télésurveillance, TV interactive, etc.). Néanmoins, pour être utilisées commercialement, ces applications doivent pouvoir offrir une certaine qualité de service afin de répondre du mieux possible aux besoins des utilisateurs.

Pour l'instant, les aspects de QdS sont essentiellement abordés du point de vue réseau (comme exemple la notion de circuits virtuels dans ATM). A ce niveau de réseau, nous pouvons uniquement fournir une information diffusée du mieux possible (ce qui est plus que compromis dans le cas d'un réseau Internet). Lorsque les limites du réseau sont atteintes, l'application ne fonctionne plus (exemple des Web-TVs). Notre projet (une implémentation est en cours) consiste à traiter ces problèmes avant qu'ils n'apparaissent. Ceci permettra à la plate-forme d'anticiper la configuration de l'application en conséquence. Elle pourra décider de supprimer purement et simplement certaines données ayant été qualifiées de "moins importantes" ou de remplacer une fonctionnalité par une autre moins gourmande en terme de flux d'information. Les informations permettant de prendre de telles décisions sont puisées dans des documents XML produits au cours de l'analyse [2].

La solution que nous proposons est basée sur une plate-forme distribuée supportant les applications multimédia. Elle gère l'ensemble des problèmes énumérés précédemment. De plus, cette plate-forme, par son implémentation en JAVA, se veut indépendante à la fois du matériel et du système d'exploitation.

Enfin, cette plate-forme s'exécute en utilisant des données (documents XML) issus d'une analyse préliminaire de l'application en terme de QdS [9] puis mises à

jour en temps réel. L'utilisation du standard XML facilite les modifications des documents afin de les adapter (et par conséquent d'adapter l'application) au nouvel environnement. Elle facilite également la diffusion des informations par utilisation d'une approche adaptée : SOAP (Simple Object Access Protocol).

12. Bibliographie

[1] Philippe Roose - ELKAR : *A Component Based Re-engineering Methodology to Provide Cooperation* - 25th Annual International Computer Software and Application Conference (COMPSAC 2001) - IEEE Computer Society Press - PR01372, pp. 65-70 - ISBN 0-7695-1372-7, ISSN 0730-3157 - Chicago - USA - October 2001

[2] Marc Dalmau, Philippe Roose, Franck Luthon - *A Software Architecture for Component Based Multimedia Applications* – 6th International Conference on Integrated Design & Process Technology (IDPT 2002) - 23-28 June 2002 - Pasadena (California) - USA *(to be published).*

[3] M. Liévin, Franck Luthon, 2000, *A hierarchical Segmentation Algorithm for Face Analysis – Application to Lipreading*, IEEE Int'l Conference on Multimedia & Expo, ICME 2000, Vol. 2, pp. 1085-1088, New-York.

[4] Le Goff, B., Guiard-Marigny, T., Cohen, M., & Benoît, C., 1994, *Real-time analysis-synthesis and intelligibility of talking faces*, Proceedings of the 2nd ESCA/IEEE Workshop on Speech Synthesis, New Paltz, New York.

[5] João Costa Seco, Luís Caires, 2000, *A basic Model of Typed Components*, ECOOP 2000, LNCS 1850, pp 108-128, Sophia Antipolis, Nice, France.

[6] Clemens Szyperski - *Component Software - Beyond Object-Oriented Programming* - Addison-Wesley / ACM Press, ISBN 0-201-17888-5, 1998.

[7] Dayal U., Blaustein B., Buchmann A. – « *The HiPAC Project* : *Combining Active Database and Timing Constraints* » - Sigmod Record - Vol 17 - N°1 - March 1988 - pp 51-70.

[8] Rob Koenen - *Overview of the MPEG-4 Standard* – International Organisation for Standardisation - ISO/IEC JTC1/SC29/WG11 – Coding of Moving Pictures and Audio - ISO/IEC JTC1/SC29/WG11 N4030 – March 2001 - (http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm

[9] Philippe Roose, Marc Dalmau, Franck Luthon - A distributed Artchirtecture for Cooperative and Adaptative Multimedia Applications - COMPSAC 2002 - IEEE Computer Society Press - 26-29 August 2002 - Oxford - England (to be published)