Adaptation d'une application multimédia par un code mobile

Daniel Hagimont — Nabil Layaïda

Projet Sirac, Projet Opéra
INRIA Rhône-Alpes,
655 avenue de l'Europe
38330 Montbonnot Saint-Martin
{Daniel.Hagimont, Nabil.Layaida}@inrialpes.fr

RÉSUMÉ. *Dans un environnement réparti, la mobilité de code peut être utilisée pour adapter dynamiquement un serveur, afin de prendre en compte les caractéristiques de l'environnement sous-jacent ou bien d'étendre le service rendu par ce serveur. Nous présentons dans cet article une expérience consistant à adapter une application multimédia répartie en utilisant un agent mobile programmé en Java. Dans cette application répartie, un agent mobile est envoyé sur le site émettant une vidéo (le serveur) afin d'adapter les données vidéo transmises. L'expérimentation que nous avons conduite montre que l'adaptation par des agents mobiles permet notamment de réduire la charge de calcul sur la machine cliente (en la reportant sur le serveur) et de réduire sensiblement la charge sur le réseau, ou encore de personnaliser la prise en compte d'une panne.*

ABSTRACT. *In a distributed environment, mobile code can be used to dynamically adapt a server in order to take into account the characteristics of the underlying environment. It cal also extend the services offered by the server. In this paper, we present an experiment which consists in adapting a multimedia application using mobile agents programmed in Java. In this distributed application, an agent is sent a video server in order to adapt the transmitted video streams. Our experiment shows that such adaptations reduce significantly the CPU load on the client machine (by transferring it to the server), reduce the network traffic and provide customized failure handling.*

MOTS-CLÉS : *Agents mobiles, Java, vidéo, adaptation, performance.*

KEY WORDS : *Mobile agents, Java, video, adaptation, performance.*

1. Introduction

Les dernières années ont vu l'émergence d'une multitude d'équipements mobiles permettant de se connecter à l'internet (téléphones, assistants personnels …). Ces équipements sont très hétérogènes, en termes de capacités mémoire, de calcul ou d'affichage. Ils peuvent être connectés à l'internet à travers des réseaux très différents, comme GPRS, Ethernet, des modems ou des cartes infrarouge. De plus, les performances de l'internet sont très variables.

La conséquence de cette évolution est qu'il devient très difficile de faire des hypothèses sur l'environnement dans lequel une application répartie sera exécutée. Il devient alors nécessaire d'adapter dynamiquement une application répartie en fonction des caractéristiques de l'environnement d'exécution.

Dans un article précédent [HAG 00], nous avons montré (à travers différents scénarios d'application) comment des agents mobiles peuvent être utilisés dans une application client-serveur afin d'adapter les serveurs en fonction de besoins spécifiques aux clients ou de contraintes liées au médium de communication. Dans cet article, nous considérons l'utilisation d'agents mobiles pour adapter une application multimédia répartie. Un agent est envoyé à la machine distante qui fournit les données multimédia (le serveur ou un site intermédiaire souvent appelé *proxy*) afin d'adapter les données transmises en fonction des contraintes du client et/ou du réseau. Nous montrons qu'il est possible, avec un agent mobile programmé en Java [GOS 95], d'adapter à la volée la vidéo transmise par le serveur et que ces adaptations permettent de mieux gérer la qualité du service fourni par l'application.

La suite de l'article est structurée de la façon suivante. La section 2 présente les motivations de ce travail. La section 3 décrit les types d'adaptation que nous considérons. L'environnement d'expérimentation que nous avons utilisé est présenté en section 4. Des résultats préliminaires concernant les gains en efficacité sont rapportés en section 5. Après une comparaison aux travaux apparentés en section 6, nous concluons et présentons les perspectives dans la section 7.

2. Motivations

2.1. *Adaptation par code mobile*

Dans un article précédent [HAG 00], nous avons étudié l'intérêt des agents mobiles par rapport au modèle client-serveur, et montré que cet intérêt résidait dans la possibilité d'adapter les serveurs en fonction des besoins spécifiques des clients ou de l'environnement sous-jacent. Ces adaptations visent en général à déporter l'exécution d'un traitement d'un site (S1) à un autre (S2), afin de bénéficier de l'effet de localité. Cette localité a principalement deux effets. Le premier est de décharger le site S1 de la charge de ce traitement, en déportant ce calcul sur S2. Le deuxième

effet est de limiter les interactions entre les deux machines dans le cas où le traitement doit utiliser des ressources sur S2, les interactions à distance entre S1 et S2 devenant des interactions locales. Ce dernier effet est également valable lorsque S2 est un site bénéficiant d'un accès plus efficace à la ressource utilisée (par exemple un réseau plus performant).

Alors que dans l'étude précédente, nous avons étudié les bénéfices des agents mobiles sur des scénarios simples, nous nous intéressons dans cet article à l'utilisation des agents mobiles pour l'adaptation de serveur multimédia. La nature de l'application considérée est présentée dans la sous-section suivante.

Notons que dans le cadre de cette étude, nous ne faisons pas de différence entre les notions de *code mobile* et d'*agents mobiles*. Ceci vient du fait que nous avons utilisé une plate-forme à agents mobiles, même si un mécanisme d'exécution de code mobile à distance serait probablement suffisant pour réaliser les adaptations décrites ici. Le lecteur trouvera dans [FUG 98] une analyse des types de systèmes à code mobile (évaluation à distance, code à la demande et agents mobiles).

2.2. *Architecture de l'application*

L'architecture de l'application que nous considérons est basée sur le modèle client-serveur. Un client joue une vidéo sur un terminal, le contenu de la vidéo étant fourni par un serveur à travers une infrastructure de communication. Comme le montre la figure 1, l'opérateur du réseau de communication, qui est responsable du routage des messages émis, gère généralement des sites intermédiaires, appelés proxy, sur lesquels des traitements additionnels peuvent être effectués. Un aspect important de cette architecture est que nos adaptations (mises en place à l'aide d'agents mobiles) peuvent être déployées sur les sites proxy par lesquels passe le flux vidéo, sans qu'il ne soit nécessaire d'ouvrir les sites serveurs à ces adaptations.

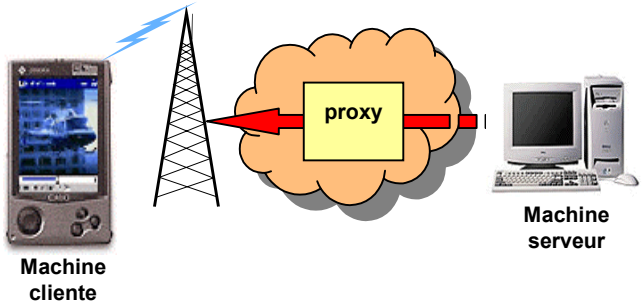

Figure 1. *Architecture de l'application*

La prise en compte des sites proxy permet de gérer deux niveaux de performance de réseau. Le premier niveau, constitué de l'épine dorsale du réseau, interconnecte les serveurs et les sites proxy avec une grande bande passante et une fiabilité relativement bonne. Le second niveau interconnecte les sites proxy et les terminaux, avec une bande passante plus limitée et variable (une liaison hertzienne par exemple) et une fiabilité plus relative.

Les sites proxy permettent également de distinguer les machines sur lesquelles sont gérés des logiciels standards (le client et le serveur) des machines sur lesquelles on réalise des adaptations. Dans notre application, les machines clientes sont pourvues d'un logiciel standard d'affichage vidéo que nous ne voulons pas modifier. Les serveurs fournissent du contenu vidéo et nous ne voulons pas modifier les logiciels du serveur ni gérer le contenu des vidéos dans différentes versions sur le serveur. Les sites proxy permettent de réaliser des adaptations déployées dynamiquement sans reconsidérer les logiciels installés sur les sites clients et serveurs.

Dans le reste de l'article, nous nous focalisons sur cette architecture à base de proxy et considérons que nos adaptations à base d'agents mobiles sont systématiquement déployées sur un site proxy par lequel passe la vidéo.

3. Les adaptations

3.1. *Amélioration de performance*

Dans notre système, une adaptation est instanciée sous la forme d'un agent mobile qui est déployé sur le site proxy afin d'y effectuer un traitement sur la vidéo envoyée au client. Comme mentionné auparavant, les deux motivations principales sont (1) de transférer de la charge de calcul du client vers le proxy et (2) de diminuer la quantité de données transférée et les interactions entre le client et le proxy. Dans les deux cas, le principe de cette adaptation est de modifier à la volée la vidéo transférée sur le proxy. Ces modifications peuvent avoir un impact non négligeable sur la charge de calcul sur le client qui présente cette vidéo, ou sur la consommation de la bande passante sur le réseau.

Considérons deux exemples illustrant ces deux motivations.

Une présentation multimédia peut être jouée sur différents types de terminaux, pouvant aller de la station de travail au PDA (*Personal Digital Assistant*). Ces terminaux diffèrent en termes de capacité de calcul et d'affichage (taille et qualité). Restituer sur un PDA une vidéo encodée pour être présentée sur un écran de station de travail est souvent inefficace parce qu'un PDA dispose d'un écran bien plus petit. Le PDA doit redimensionner la vidéo alors qu'il ne possède pas la capacité de calcul permettant de redimensionner la vidéo à la volée. Un scénario d'adaptation très

simple consiste à envoyer un agent mobile afin de redimensionner la vidéo sur le proxy, qui possède naturellement plus de capacité de calcul que le PDA. Notons au passage que certains formats d'encodage de la vidéo, généralement de moins bonne qualité, demandent moins de capacité de calcul au terminal. Certains terminaux peuvent également être optimisés pour le décodage de formats vidéo particuliers au moyen de protocoles réseau spécifiques. Il peut alors être intéressant d'envoyer sur le proxy un agent qui utilise ce protocole et convertit la vidéo dans le format supporté par le terminal.

Le second exemple illustre l'adaptation à la variation de bande passante du réseau. La vidéo peut être transférée sur des réseaux ayant des bandes passantes différentes, voire variables. Dans ce cas, l'adaptation consiste à envoyer sur le proxy un agent qui adapte la vidéo en fonction de la bande passante disponible. Cette adaptation peut modifier le facteur de qualité des images ou réduire la cadence des images, ce qui réduit significativement le volume d'information émis.

Notons que certaines adaptations peuvent cumuler ces deux avantages. Si on considère une vidéo restituée sur un PDA avec un écran noir et blanc à travers un réseau à faible bande passante, le fait de modifier l'encodage de la vidéo (et de réduire la qualité de la présentation) ou de supprimer les informations concernant la couleur (puisque le terminal n'est pas en mesure de les exploiter) peut à la fois réduire la charge de calcul nécessaire au décodage de la vidéo sur le PDA et réduire le volume de données transmises entre le proxy et le PDA. Un agent qui filtre les données transférées sur le proxy permet d'économiser des cycles de calcul sur le terminal du client et de réduire la bande passante consommée par l'application.

3.2. *Traitement de pannes temporaires*

Les terminaux mobiles utilisent souvent des réseaux hertziens qui se caractérisent par une bande passante réduite et variable, et par des pannes temporaires (interférences, déconnexions) plus fréquentes. Le traitement et la compensation de ces pannes sont un domaine d'application privilégié des adaptations à base d'agents mobiles.

Considérons le cas d'un usager qui visionne le journal télévisé sur son PDA. Le journal télévisé est rendu disponible par une chaîne nationale sur un serveur Web.

Les pannes temporaires que nous voulons prendre en compte ici sont les ruptures de connexion entre le terminal client et le proxy, par exemple lorsqu'il s'agit d'une liaison hertzienne. En cas de rupture de connexion entre le client et le proxy, l'usager n'a certainement pas envie de relancer la présentation depuis le début, mais plutôt de la reprendre là où il en était, même si la session a été interrompue pendant quelques instants. Pour fournir un tel service (sans recharger à nouveau toute la vidéo sur le PDA), il est nécessaire de disposer d'une interface adaptée sur le serveur, permettant de reprendre la session au point où elle a été interrompue. Un agent mobile envoyé

sur le proxy permet de fournir ce service à l'usager. Cet agent permet dans une première approche de recharger la vidéo sur le proxy et de ne retransmettre la vidéo du proxy vers le client qu'à partir du point d'interruption. Dans une seconde approche, le proxy peut servir de machine tampon. Le proxy peut, en cas de rupture de connexion entre le proxy et le client, continuer à acquérir les données vidéo à partir du serveur, le temps que la connexion soit rétablie.

Dans le cas où la panne temporaire que l'on veut prendre en compte est une rupture de connexion entre le proxy et le serveur, l'agent d'adaptation qui est envoyé sur le proxy peut implanter une politique de pré-chargement des données sur le proxy. Ainsi, en cas de rupture de connexion, le proxy pourra continuer à émettre les données vers le client, pendant que le proxy se reconnecte au serveur pour obtenir le reste des données. Une telle stratégie consommera des ressources disques sur le proxy, mais permettra en contrepartie d'économiser de la bande passante entre le proxy et le serveur (en évitant de recharger la vidéo pour reprendre la session).

Ces politiques de gestion des pannes temporaires ne peuvent pas être implantées sur un équipement mobile. Elles peuvent être installées dynamiquement sur les sites proxy, à l'aide de code mobile, sans nécessiter de modification des logiciels utilisés sur les sites clients et serveurs.

3.3. *Personnalisation du service*

La troisième classe d'adaptation que nous considérons concerne l'extension du service fourni par l'application multimédia répartie. Nous visons ici des traitements sur la vidéo, effectués sur le site proxy, qui modifient son contenu.

Il peut s'agir d'intégrer une fonction supplémentaire dans la présentation de la vidéo, comme par exemple une notification en sous-titre lorsqu'un événement se produit comme l'arrivée d'un courrier électronique. Les terminaux très légers comme les PDA ne sont généralement pas multi-tâches et peuvent difficilement traiter l'arrivée d'un courrier électronique pendant que le PDA est en train de présenter une vidéo chargée à travers le réseau. Dans notre architecture, un agent peut être déployé sur le site proxy et chargé de réagir à un événement *arrivée de message*. La réaction de l'agent consiste à incruster dans la vidéo en sous-titre le nom de l'émetteur du message ainsi que le titre du message par exemple.

Une autre fonction, moins sympathique, est l'incrustation de messages publicitaires, qui peuvent par exemple être ajoutés par l'opérateur de télécommunication en fonction du profil de l'utilisateur, indépendamment du serveur et de la vidéo présentée.

4. Environnement d'expérimentation

Dans cette section, nous décrivons l'environnement d'expérimentation utilisé pour tester la viabilité de cette démarche. Nous décrivons également l'implantation que nous avons réalisée.

4.1. *Architecture de l'application*

Pour conduire notre expérimentation, nous avons développé une application dans laquelle un client joue une présentation vidéo chargée depuis un serveur Web. Le client se connecte à un site proxy en lui passant en paramètre l'URL du document multimédia à présenter. Le proxy accède au document multimédia avec le protocole HTTP. La communication entre le client et le proxy utilise le protocole RTP (Real Time Protocol) [IETF 00]. RTP est un protocole d'envoi de paquets implanté au dessus d'UDP, qui gère le cadencement des données multimédia à l'aide d'horloges logiques réparties. Cette architecture est représentée sur la figure 2.

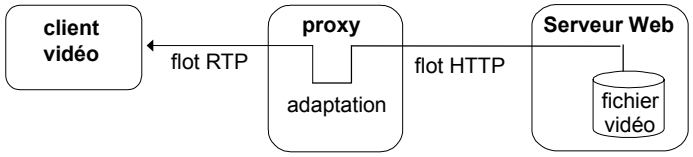

Figure 2. *Environnement d'expérimentation*

Dans notre environnement d'expérimentation, les trois machines, le client, le proxy et le serveur, sont des stations de travail, et le réseau d'interconnexion est un réseau Ethernet à 100 Mb/s.

Cet environnement matériel nous a permis de mettre au point nos scénarios de l'application multimédia répartie, et également d'évaluer les bénéfices de l'adaptation de l'application par des agents mobiles. Comme nous le verrons par la suite, nous avons pu notamment mesurer la réduction du volume de données transféré et la réduction de la charge de calcul sur la machine cliente, pour les adaptations que nous avons sélectionnées. Nous avons également pu évaluer la capacité d'adaptation de la vidéo à la volée sur le site proxy.

4.2. *Architecture logicielle*

Les logiciels sur le site serveur et sur le site client sont standards. Il s'agit respectivement d'un serveur Web Apache et d'un client de visualisation multimédia (nous avons utilisé JMStudio de l'environnement JMF décrit ci-dessous).

Notre objectif est de réaliser des adaptations de la vidéo sur le site proxy, en même temps que la vidéo est transférée vers le terminal client. Ces adaptations modifient le contenu du flot vidéo.

L'adaptation est effectuée par un agent mobile programmé en Java, qui est envoyé sur le site proxy préalablement à l'ouverture de la session multimédia. Nous appelons dans la suite **agent d'adaptation** l'agent qui est envoyé sur le proxy pour adapter la vidéo. Nous appelons **configuration du proxy** l'installation dynamique de l'agent d'adaptation sur le site proxy. Un service de configuration est disponible sur le proxy et permet la configuration du proxy. L'agent est envoyé au proxy depuis un *contrôleur de l'application* par un appel à distance (effectué au moyen de Java-RMI) au service de configuration sur le proxy. La figure 3 représente cette architecture.

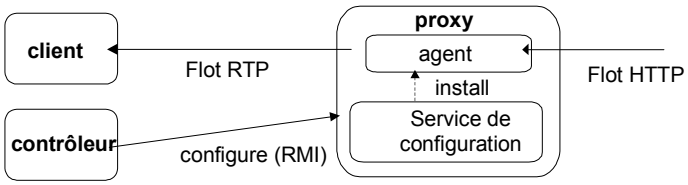

Figure 3. *Architecture logicielle d'adaptation*

Le code du proxy (le service de configuration) et le code des agents d'adaptation sont programmés en Java. Ils reposent sur Java Media Framework (JMF), un environnement de gestion de données multimédia (dans sa version 2.1.1 [SUN 99]). Nous avons utilisé l'environnement JMF pour différentes raisons. JMF est modulaire et facilite la combinaison de composants logiciels, permettant ainsi la configuration d'un logiciel sur le proxy. Il est assez complet, puisqu'il fournit la plupart des composants dont nous avons besoin. Des versions optimisées pour des plate-formes particulières sont également disponibles.

L'unité de traitement dans JMF est le *processeur*. Un processeur est un canal de traitement de flot multimédia qui contient 3 étages :

- un étage qui assemble les données (paquets) reçues en entrée et démultiplexe les flots média (audio et vidéo),

- un étage de traitement de chaque flot média en utilisant des codecs (librairies d'encodage et de décompression) ou des effets, qui sont des spécialisations de codecs que nous introduisons pour adapter le contenu des images de la vidéo,

- un étage pour le multiplexage des flots média obtenus et pour le découpage du flot multimédia résultant en paquets destinés à l'émission sur le réseau.

L'architecture à base de processeurs JMF est représentée sur la figure 4.

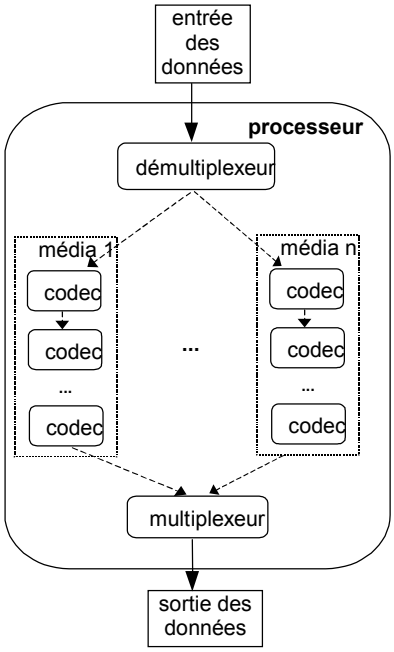

Figure 4. *Architecture d'un processeur JMF*

Le format de sortie de chaque flot media, incluant son format d'encodage et sa taille d'image, ainsi que le format de sortie du flot multimédia multiplexé (comme H.263 [ITU 96] ou MPEG [ISO 93]) peuvent être spécifiés à l'initialisation du processeur. JMF fournit des (dé)multiplexeurs et des codecs de (dé)codage des images.

Le code mobile d'adaptation peut inclure de nouveau codecs pouvant spécialiser les traitements effectués sur le site proxy, répondant à des besoins particuliers. L'agent d'adaptation initialise le processeur JMF qui effectue ces traitements. Le flot multimédia obtenu en sortie de processeur est envoyé en utilisant la pile protocolaire RTP de JMF.

Ainsi, le client reçoit une vidéo à travers une connexion RTP, que ce soit depuis un proxy adapté (avec notre architecture) ou non adapté (il reçoit la vidéo telle quelle). Nous n'avons donc pas besoin de modifier le logiciel (système ou application) sur le poste client.

4.3. *Adaptations du flot multimédia*

Un **flot multimédia** est composé de l'assemblage d'un ensemble de **flots média**, pouvant contenir de l'audio ou de la vidéo. Nous nous intéressons ici plus particulièrement à un flot vidéo composé d'une séquence d'images encodées (l'encodage couvre aussi la compression). Une image non encodée (que nous appelons **image brute**) est représentée dans un format linéaire, tel que RGB à 24 bits/point, permettant notamment de traiter (présenter à l'écran ou modifier) le contenu de l'image.

Cet empilement de couches est représenté sur la figure 5.

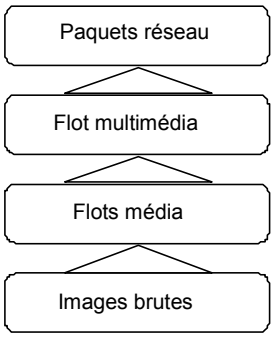

Figure 5. *Hiérarchie d'encapsulation multimédia*

A chaque étage de l'encapsulation, des adaptations peuvent être effectuées :

- Au niveau des paquets réseaux (objet *entrée des données* sur la figure 4). C'est à ce niveau que sont généralement implantés les gestionnaires de trafic (*traffic shapers*) qui peuvent modifier la forme du trafic sur le réseau, pour mieux gérer les surcharges du réseau.

- Au niveau du flot multimédia (objet *démultiplexeur* sur la figure 4). C'est à ce niveau que l'on peut ignorer les flots média inutiles, ou optimiser la synchronisation entre les flots média (audio et vidéo).

- Au niveau des flots media (objets *codecs* sur la figure 4). Certains de ces codecs sont des encodeurs dont on peut modifier la nature ou les paramètres qui contrôlent l'encodage des images.

- Au niveau du contenu des images en réduisant la taille ou en retirant des informations jugées inutiles comme la couleur, ou encore en modifiant le contenu (sémantique) des images.

5. Evaluation de performance

Cette section propose une évaluation de performance de notre architecture permettant l'adaptation d'applications multimédia par des agents mobiles. Nous précisons tout d'abord les conditions d'évaluation, puis nous présentons les coûts de base des principaux mécanismes d'adaptation. Nous proposons ensuite une évaluation d'un scénario d'adaptation. Cette adaptation consiste à convertir une vidéo du format MPEG au format H.263 et à redimensionner les images. Cette évaluation étudie les performances de l'adaptation sur la machine proxy, les performances de la présentation sur la machine cliente et le volume d'information transféré entre ces deux machines.

Toutes les mesures ont été effectuées dans des conditions réelles, à savoir avec un déploiement complet de notre architecture logicielle.

5.1. *Environnement de mesures*

L'environnement de mesure implante l'architecture présentée en section 4. Il s'agit de trois machines interconnectées par un réseau Ethernet à 100 Mb/s.

La première machine exécute un serveur Web Apache et met à disposition une vidéo au format MPEG. La seconde machine exécute notre logiciel de proxy adaptable. Il s'agit d'un PC AMD Athlon à 1,2 GHz avec 256 Mo de mémoire. La machine cliente est un PC Pentium III à 1GHz avec 256 Mo de mémoire.

Toutes les mesures ont été effectuées avec le même fichier vidéo. Ce fichier contient un flot vidéo MPEG avec une taille de 352X288 points et une cadence de 30 images/sec.

Notre évaluation a consisté à comparer deux configurations du proxy. La première transmet le flot vidéo MPEG sans le modifier. La seconde modifie le format d'encodage de la vidéo en la convertissant au format H.263 et réduit la taille de la vidéo. Dans la seconde configuration, nous avons utilisé trois tailles d'image standards, qui sont celles que l'on peut générer en H.263 (table 1) :

Format	**Largeur**	**Hauteur**
CIF	352	288
QCIF	176	144
SQCIF	128	96

Table 1. *Tailles des images pour notre expérimentation*

Ces mesures permettent de montrer que l'adaptation à la volée de la vidéo transmise dans une application multimédia répartie est possible à des coûts

acceptables. D'autres adaptations peuvent également viser le traitement des déficiences du réseau (section 3.2) ou la modification du service accompli par l'application multimédia (section 3.3).

Nous avons comparé l'envoi de la vidéo en MPEG dans sa taille originelle, avec l'adaptation à la volée de la vidéo en la convertissant en format H.263, respectivement dans les tailles CIF, QCIF et SQCIF. Nous avons mesuré le coût de mise en place de l'adaptation (section 5.2), la capacité du proxy à effectuer l'adaptation à la volée (section 5.3), l'impact de l'adaptation sur le volume d'information envoyé sur le réseau (section 5.4), et enfin l'impact de l'adaptation sur la charge de calcul nécessaire au décodage de la vidéo sur la machine cliente (section 5.5).

5.2. *Coûts des mécanismes d'adaptation*

La configuration du proxy à l'aide d'un agent d'adaptation se compose de l'envoi de l'agent sur le site proxy (en utilisant Java-RMI) et de l'initialisation du processeur JMF servant à appliquer les adaptations embarqués dans l'agent.

L'initialisation d'un processeur JMF est réalisée en deux phases : la spécification et la création. La spécification est la définition des contraintes utilisées pour la construction du processeur. Les contraintes pouvant être spécifiées sont :

- au niveau du flot multimédia : le format d'encodage en sortie,

- au niveau de chaque flot média : le format de sortie et la liste des codecs qui doivent être présents dans le canal associé à ce flot média,

- au niveau de chaque codec : les formats d'entrée et de sortie.

La création est la phase pendant laquelle les canaux du processeur sont construits et optimisés, tout en respectant les contraintes spécifiées. Si besoin est, par exemple lorsque le format de sortie d'un codec est différent de celui d'entrée du prochain codec dans le canal, des composants additionnels peuvent être automatiquement insérés dans le canal lors de la création.

L'initialisation d'un processeur JMF est une opération complexe qui est coûteuse en temps. Nous avons mesuré le temps d'initialisation du processeur JMF dans les deux cas principaux qui nous concernent dans ces mesures :

- MPEG. Le processeur JMF acquiert ses données depuis le serveur Web et les transmets directement sur une connexion RTP avec la machine cliente.

- H.263. Le processeur JMF acquiert ses données depuis le serveur Web, redimensionne les images et convertit la vidéo au format d'encodage H.263, puis transmets le flot vidéo sur une connexion RTP avec la machine cliente.

La table 2 donne les coûts d'initialisation dans ces deux cas.

Cas	Coût d'initialisation
MPEG	1905 ms
H.263	2227 ms

Table 2. *Coût d'initialisation d'un processeur JMF*

L'initialisation d'un processeur JMF est chère dans les deux cas, mais reste acceptable par rapport à la durée d'une présentation multimédia.

Nous avons également observé que les coûts d'installation du code mobile (implantant un comportement particulier), et d'ajout dans le processeur JMF d'un codec supplémentaire sont négligeables par rapport au temps d'initialisation du processeur JMF.

5.3. *Performances de l'adaptation à la volée sur le proxy*

L'adaptation de la vidéo à la volée à l'aide d'un code mobile installé dynamiquement sur la machine proxy n'est possible que si le site proxy est capable de traiter ces données en temps réel. Afin d'évaluer la capacité de traitement en temps réel sur le site proxy, nous avons mesuré la cadence des images à la sortie du proxy lorsque celui-ci réalise une conversion en format d'encodage H.263 en redimensionnant la vidéo. Ces mesures ont été prélevées au niveau de la connexion RTP sortant du proxy.

La cadence d'image obtenue est fortement dépendante de la taille ciblée, car le travail d'encodage est réduit lorsque les images à traiter sont plus petites. La table 3 donne les résultats obtenus.

Taille	Conversion MPEG -> H.263
CIF	6 images/sec
QCIF	19 images/sec
SQCIF	30 images/sec

Table 3. *Cadence des images après conversion sur le proxy*

On observe que la machine proxy peut difficilement générer à la volée une vidéo en H.263 et au format CIF, car on obtient une cadence de sortie de 6 images par seconde. Rappelons que la cadence théorique de la vidéo est de 30 images par secondes. Cependant, lorsque l'on diminue la taille de la vidéo ciblée, on voit que le proxy est capable de générer une vidéo à une cadence nettement supérieure. En particulier, pour le format SQCIF qui est probablement le format approprié pour un

terminal portable comme un PDA, le proxy est capable d'effectuer la conversion en temps réel à la cadence théorique (30 images par seconde) de la vidéo.

De nombreux effets peuvent être intégrés sur le proxy afin de modifier la vidéo comme dans les exemples décrits en section 3.1. Afin d'évaluer l'influence du rajout de codecs dans le canal de traitement de la vidéo sur le proxy, nous avons mesuré l'évolution de la cadence d'image à la sortie du proxy lorsque l'on fait varier le nombre de codecs. Les codecs que nous avons ajoutés sont des codecs *passants* qui transmettent simplement la vidéo au codec suivant.

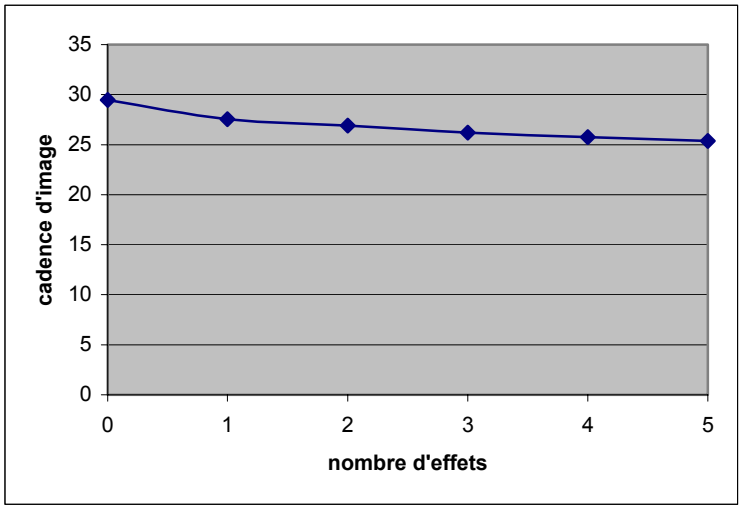

Figure 6. *Hiérarchie d'encapsulation multimédia*

On observe sur la figure 6 que lorsqu'on ajoute 5 codecs, la cadence d'image passe de 30 à 25 images secondes. Ce surcoût nous paraît acceptable, étant donné que le nombre de codecs combinés ne doit probablement pas devenir très grand. Il faut noter ici que ces mesures portent sur le coût de l'infrastructure permettant de rajouter des codecs d'adaptation. Le coût effectif de l'adaptation dépend naturellement de l'effet réalisé.

Notons enfin qu'une autre adaptation possible (il y en a beaucoup d'autres) est de réduire la cadence théorique de la vidéo afin de réduire la charge sur le proxy, le réseau et la machine cliente.

5.4. *Impacts sur le volume de données transmises*

Lorsque les adaptations sont déployées pour améliorer les performances de l'application multimédia répartie, le bénéfice visé peut concerner le volume de données transféré sur le réseau.

Dans le scénario que nous utilisons pour nos mesures, nous encodons la vidéo en H.263 et nous réduisons la taille des images de la vidéo. L'encodage en H.263 réduit fortement le volume de données de la vidéo encodée, sachant que la vidéo obtenue est de moins bonne qualité. Cette perte de qualité n'est pas forcément préjudiciable pour l'usager, lorsque celui-ci utilise l'application depuis un terminal dont les capacités d'affichage sont limitées. La réduction de la taille des images réduit naturellement le volume de données de la vidéo.

Afin d'évaluer l'impact de nos adaptations sur le volume d'information transféré, nous avons observé l'émission de paquets RTP sur le proxy. La figure 7 montre l'évolution dans le temps de la quantité d'octets émis sur la connexion RTP.

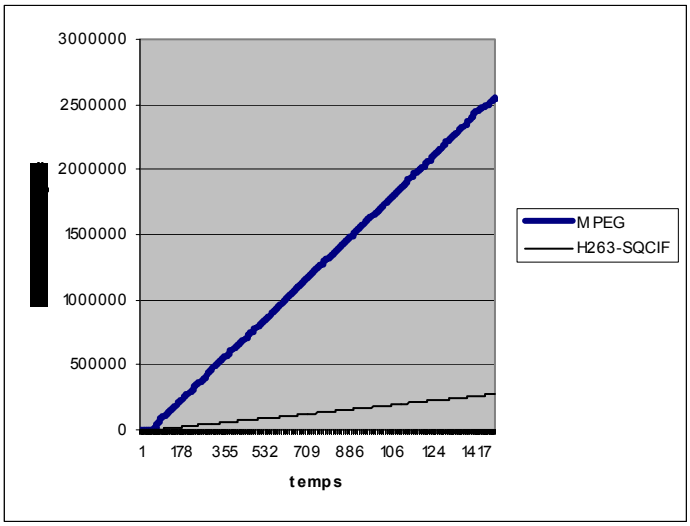

Figure 7. *Volume de données émises sur le réseau*

Dans la figure 7, nous ne présentons la courbe d'émission pour le format d'encodage H.263 que pour la taille d'image SQCIF. En effet, comme on l'a vu en section 5.3, c'est pour cette taille que le proxy est capable de convertir la vidéo de MPEG à H.263 en temps réel. Pour les autres tailles, le proxy ne peut pas convertir la vidéo en temps réel et la quantité d'octets émise diminue forcément.

Format	Cadence d'émission
MPEG	173 K octet/sec
H.263-SQCIF	18 K octet/sec

Table 4. *Cadence d'émission de données sur le réseau*

La table 4 donne les pentes des courbes de la figure 7, c'est à dire les cadences mesurées d'émission des données. On note que la conversion au format H.263-SQCIF a un effet significatif sur l'utilisation des ressources du réseau.

5.5. *Impact sur la charge de calcul du client*

Les adaptations réalisées sur le site proxy peuvent également avoir un effet important sur la charge de calcul imposée sur la machine cliente pour décoder la vidéo.

Afin d'observer l'influence des adaptations sur la machine cliente, nous avons mesuré au niveau de l'outil de visualisation vidéo de JMF la cadence d'affichage des images sur la machine cliente, en faisant varier la charge sur la machine cliente (en exécutant des programmes de calcul intensif). Ces mesures ont été effectuées pour les deux formats que notre proxy sait traiter à la volée, à savoir une transmission directe en MPEG et une adaptation au format H.263 en taille SQCIF. Les résultats, présentés sur la figure 8, montrent la sensibilité des deux formats à la capacité de calcul du terminal client.

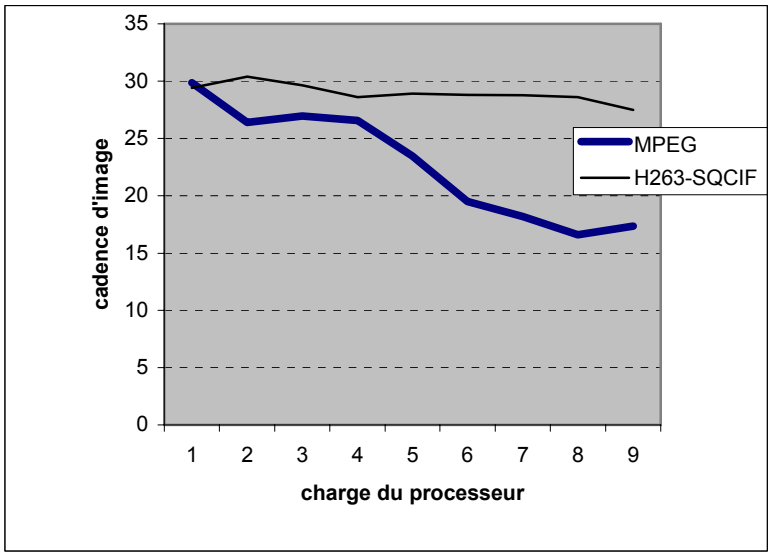

Figure 8. *Charge de la présentation sur la machine cliente*

On observe sur la figure 8 que lorsque que la charge sur la machine cliente augmente, la cadence de présentation des images diminue beaucoup plus avec le format MPEG qu'avec le format H.263-SQCIF. Nous en déduisons qu'une limitation de la capacité de calcul sur la machine cliente affecterait moins la présentation de la vidéo lorsque celle-ci a été convertie au format H.263-SQCIF par l'adaptation sur le site proxy.

5.6. *Résumé des mesures*

Nous avons évalué l'impact de nos adaptations (sur le site proxy) sur les performances d'une d'application multimédia répartie. L'adaptation consiste à convertir cette vidéo de l'encodage MPEG à l'encodage H.263, et à passer la vidéo de la taille CIF à la taille SQCIF. Cette évaluation se résume de la façon suivante :

- Le proxy peut effectuer cette adaptation en temps réel.

- Le volume d'information transféré sur le réseau est réduit de façon significative.

- La charge de calcul pour le décodage de la vidéo sur la machine cliente est sensiblement allégée.

Ces expérimentations ont été menées sur l'environnement Java Media Framework de Sun [SUN 99]. Bien que cet environnement se caractérise par des performances médiocres, il nous a permis de montrer que cette approche était réaliste. Nous avons plus récemment reproduit cette expérience sur l'environnement DirectShow de Microsoft [DX 01]. Les résultats préliminaires montrent que, pour une infrastructure similaire, les gains de performance sont substantiels, ce qui renforce la validité des résultats présentés dans cet article.

De nombreuses autres adaptations peuvent être effectuées pour améliorer les performances, mais également pour modifier d'autres aspects de l'application comme le montrent nos exemples d'adaptations décrits dans la section 3.

6. Travaux apparentés

Notre objectif est de montrer que le code mobile permet d'adapter les applications multimédia réparties afin de prendre en compte des besoins spécifiques provenant de l'environnement sous-jacent ou afin d'étendre les fonctionnalités de l'application.

Nous comparons notre expérimentation à d'autres effectuées dans des projets qui utilisent des techniques d'adaptation pour mieux gérer des présentations multimédia réparties.

De nombreux travaux se sont intéressés à la gestion de la Qualité de Service (Qos) de bout en bout dans des architectures spécifiquement conçues pour traiter ce problème de QoS [HAF 98]. Ces travaux s'intéressent plus particulièrement à l'utilisation des ressources du réseau et se basent souvent sur des techniques de réservation de bande passante comme dans [BIA 00][MAR 00]. Ces propositions sont difficiles à intégrer dans les architectures actuelles, car elles supposent l'installation préalable des fonctions de gestion de la QoS sur l'ensemble des machines (y compris l'infrastructure réseau). Notre proposition vise justement à déployer dynamiquement, sous la forme d'agents mobiles, les mécanismes de gestion de la QoS qui sont souvent dépendants des besoins spécifiques de l'application et de l'environnement sous-jacent.

Alors que cette dernière approche ne traite que la gestion de la ressource réseau, d'autres approches s'intéressent à gérer la QoS dans des applications multimédia réparties en adaptant le format de la vidéo en fonction de l'environnement sous-jacent, réseau ou terminal client.

Une première technique est de gérer, sur le serveur rendant disponible une vidéo, différentes versions de la vidéo, chacune de ces versions répondant à différentes contraintes de QoS. Cette approche a notamment été explorée dans le projet Odyssey [NOB 97]. L'avantage est la simplicité de l'approche, mais elle manque grandement de flexibilité, car le serveur ne peut gérer toutes les versions correspondant à tous les cas possibles. De plus, une nouvelle contrainte (par exemple un nouveau type de terminal) nécessite une prise en compte sur tous les serveurs.

Une autre approche intéressante est l'encodage adaptatif [LI 98] qui propose un encodage multi-niveau permettant de dériver différentes qualités de la même vidéo à partir du même flux vidéo originel. Cette technique, bien que très prometteuse, nécessite aussi de disposer de cette représentation de la vidéo sur tous les serveurs et ne prend pas en compte les vidéos existantes. Cependant, notre infrastructure de proxy adaptable peut tirer parti de l'encodage adaptatif et appliquer différents filtres sur de telles vidéos.

Plusieurs travaux ont déjà adressé l'adaptation à la volée du contenu d'un flux vidéo, en déployant des filtres d'adaptation sur les nœuds intermédiaires de l'infrastructure réseau [ANG 98][BAH 98]. Comme indiqué dans [ANG 98], ces travaux reposent essentiellement sur des simulations. De plus, ces travaux se sont principalement focalisés sur des adaptations au niveau des flots média (figure 5). Les adaptations travaillent sur les données encodées de la vidéo et consistent soit à supprimer des images, soit à supprimer des niveaux de qualité dans un encodage adaptatif. Nos travaux visent à utiliser le déploiement de code mobile sur des sites proxy afin d'adapter la vidéo à un niveau quelconque parmi ceux de la figure 5. En particulier, ces adaptations peuvent changer de format d'encodage de la vidéo, la redimensionner ou la passer en noir et blanc. Ces adaptations peuvent également compenser des pannes ou modifier le contenu sémantique de la vidéo, comme dans les scénarios décrits en section 3.2 et 3.3.

7. Conclusion et perspectives

Nous avons présenté une expérience consistant à utiliser du code mobile pour adapter une application multimédia répartie.

Dans cette expérimentation, une vidéo est transmise depuis un site Web vers une machine cliente qui présente la vidéo. Le flot vidéo passe à travers un site intermédiaire appelé *proxy*. Le proxy peut être configuré à distance avec des agents d'adaptation qui appliquent une transformation sur le flot vidéo transmis à la machine cliente. Le mise en œuvre de cette application repose sur l'infrastructure Java Media Framework.

Notre infrastructure peut être étendue pour implanter différents scénarios d'adaptation, visant soit à adapter le contenu de la vidéo, soit à traiter des pannes temporaires du réseau entre le proxy et le terminal client, soit à améliorer les performances de l'application en adaptant la vidéo en fonction de l'environnement sous-jacent.

Afin de valider cette approche, nous avons effectué des mesures dans le cas où l'adaptation vise l'amélioration de performance. Un agent mobile d'adaptation est utilisé pour configurer le proxy, afin qu'il transforme une vidéo MPEG de taille CIF en vidéo H.263 de taille SQCIF (de plus petite taille). Les mesures montrent que le proxy (matérialisé par un PC) peut effectuer cette adaptation en temps réel et que cette adaptation a une influence bénéfique et significative sur le volume de données transmises sur le réseau et sur la charge de calcul nécessaire au décodage de la vidéo sur la machine cliente.

Cette première expérimentation montre l'intérêt pour des applications multimédia de fournir des machines proxy configurables et que la configuration de ces proxy nécessite le déploiement de code mobile.

Elle ouvre de nombreuses perspectives que nous explorons à l'heure actuelle.

En premier lieu, notre expérimentation s'est limitée à l'utilisation de stations de travail, étant donné les limitations des environnement de développement d'applications multimédia sur les machines mobiles comme les PDA. Nous disposons à l'heure actuelle d'une plate-forme à base de PDA (iPAQ) communicants par un réseau sans fil (de type 802.11). Cette plate-forme nous permettra d'effectuer des mesures plus précises concernant le bénéfice des adaptations dynamiques de la vidéo sur les sites proxy.

Deuxièmement, si notre expérimentation avec JMF nous a amené à des résultats prometteurs, l'utilisation d'un environnement logiciel plus efficace comme DirectShow de Microsoft et le code mobile .Net ouvre de nouvelles perspectives en ce qui concerne l'efficacité des adaptations sur le site proxy.

Dans des environnements à base de machines mobiles et de réseaux sans fil, les ressources disponibles peuvent fortement fluctuer (en particulier les ressources du

réseau), ce qui nécessite des techniques de reconfiguration des proxy, afin de répondre dynamiquement à ces variations, comme par exemple la modification en cours d'exécution des codecs utilisés sur le proxy pour adapter la vidéo, afin de mieux compenser les déficiences du réseau.

Enfin, le déploiement de code mobile sur les proxy, pour être utilisable, doit traiter tous les problèmes liés à la sécurité des sites proxy. Le site proxy doit être en mesure de contrôler qu'un agent d'adaptation n'effectue pas des opérations malveillantes. Il doit également contrôler l'utilisation des ressources des proxy par les agents d'adaptation. Un premier niveau de sécurité peut être obtenu en réservant les opérations de configuration des proxy aux opérateurs de télécommunication offrant ce service à leurs usagers.

8. Bibliographie

[ANG 98] O. ANGIN, A. CAMPBELL, M. KOUNAVIS, R. LIAO. The Mobiware toolkit: programmable support for adaptive mobile networking. IEEE Personal Communications Magazine, 5(4), p. 32-43, Août 1998.

[BAH 98] P. BAHL. Supporting digital video in managed wireless network. IEEE Communications Magazine, p. 94-102, Juin 1998.

[BIA 00] G. BIANCHI, A. CAMPBELL. A Programmable MAC Framework for Utility-Based Adaptive Quality of Service Support. IEEE Journal on Selected Areas in Communications, 18(2), Février 2000.

[DX 01] Microsoft DirectX 8.0, 2001. http://www.microsoft.com/directx/

[FUG 98] A. FUGGETTA, G. PICCO, G. VIGNA. Understanding code mobility. IEEE Transactions on Software Engineering, Vol. 24(5), p. 342-361, 1998.

[GOS 95] J. GOSLING, H. MCGILTON. The Java Language Environment: a White Paper, Sun Microsystems Inc., 1995.

[HAF 98] A. HAFID, G.V BOCHMANN. Distributed Multimedia Applications and Quality of Service : a review. Electronic Journal on Network and Distributed Processing, n° 6, p. 1-50, 1998.

[HAG 00] D. HAGIMONT, L. ISMAIL. Agents mobiles et client/serveur : évaluation de performance et comparaison. Technique et Science Informatiques, Vol. 19(9), 2000.

[IETF 00] IETF. RTP: A Transport Protocol for Real-Time Applications. Internet Draft, version 8, revision of RFC 1889, Juillet 2000.

[ISO 93] ISO/IEC 11172-2:1993, The MPEG standard, Part 2: Video.

[ITU 96] ITU Telecom. Standardization Sector of ITU, Video coding for low bitrate communication. ITU-T Recommendation H.263, Mars 1996.

[LI 98] Y-C LI, T-H WU, Y-C CHEN, A SCENE. Adaptive Hybrid Video Coding Scheme Based on the LOT. IEEE Transactions on Circuits and Systems for Video Technology. Vol. 8(1), p. 92-103, 1998.

[MAR 00] M. MARGARITIDIS, G. POLYZOS. MobiWeb : Enabling adaptive continuous media applications over wireless links. IEEE International Conference on Third Generation Wireless Communications, Silicon Valley, San Francisco, Juin 2000.

[NOB 97] B. NOBLE, M. SATYANARAYANAN, D. NARAYANAN, J. TILTON, J. FLINN, K WALKER. Agile application-aware adaptation for mobility. Proceedings of the 16th ACM Symposium on Operating Systems Principles, Saint-Malo, France, Octobre 1997.

[SUN 99] Sun Microsystems Inc. Java Media Framework API Guide. Novembre 1999.

Article reçu le 5 février 2001
Version révisée le 4 octobre 2001
Rédacteur responsable : Abdelkader Hameurlain

Daniel Hagimont *a soutenu en 1993 une thèse de doctorat à l'Institut National Polytechnique de Grenoble (INPG). Après un séjour d'une année à l'Université de Colombie Britannique (Vancouver), il occupe actuellement les fonctions de chargé de recherches à l'unité Rhône-Alpes de l'INRIA. Il a également soutenu en 1998 une Habilitation à Diriger des Recherches à l'INPG. Ses travaux de recherche portent sur les systèmes d'exploitation et systèmes répartis.*

Nabil Layaïda *a soutenu en 1997 une thèse de doctorat à l'Université Joseph Fourier, Grenoble. Après un séjour d'une année à l'Université d'Ottawa au Canada, il occupe actuellement les fonctions de chargé de recherches à l'unité Rhône-Alpes de l'INRIA. Il est membre du groupe de normalisation SYMM (Synchronized Multimedia) au sein du World Wide Web Consortium (W3C) et co-auteur des langages SMIL 1.0 et SMIL 2.0 (Synchronized Multimedia Integration Language). Ses thèmes de recherche portent sur le traitement de documents multimédia structurés, l'adaptation automatique de contenu et la gestion de la qualité de service pour les objets communicants.*