Université de Montréal

Applications multimédia et gestion de qualité de service dans l'environnement du World Wide Web

par

Erika Jeannette Madja
Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.) en informatique

Octobre 1997

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé :

Applications multimédia et gestion de qualité de service dans l'environnement du World Wide Web

présenté par:

Erika Jeannette Madja

a été évalué par un jury composé des personnes suivantes :

Président du jury : Rudolf K. Keller
Directrice de recherche: Rachida Dssouli
Codirecteur de recherche : Jan Gecsei
Membre du jury : Esma Aïmeur

Mémoire accepté le : 12 décembre 1997

Sommaire

Développé à l'origine pour permettre l'accès à des documents statiques comme les fichiers Ascii et les images, le World Wide Web a connu un développement très rapide ces dernières années avec l'introduction de données de plus en plus médiatisées. Cependant l'introduction des données multimédia crée de nouveaux besoins dans la mesure où les protocoles du Web n'étaient pas conçus à l'origine pour ce genre de trafic. Outre le problème des protocoles se pose également celui de la gestion de la Qualité de Service (QdS) offerte par les systèmes multimédia distribués.

La nécessité de gérer la QdS dans les applications multimédia distribuées s'impose pour certaines raisons. Il s'agit d'une part des contraintes temporelles de certains média appelés média continus et des besoins considérables en ressources pour les manipuler. D'autre part la nature des systèmes distribués implique que les diverses composantes du système n'ont pas les mêmes caractéristiques ni les mêmes performances. Les réseaux qui relient les clients et les serveurs du système peuvent offrir différentes largeurs de bande et différents délais. La capacité de prendre en compte la variété de performance des diverses composantes du système et de s'adapter aux fluctuations de ces niveaux de performance est primordiale pour les applications multimédia distribuées. En effet ces performances ont une influence directe sur la qualité de la présentation des média telle que perçue par l'utilisateur. Ceci est particulièrement crucial dans le contexte particulier du Web où le réseau sous-jacent, l'Internet, est très hétérogène. Divers utilisateurs possèdent différents équipements avec des capacités diverses, les liens d'accès à l'Internet allant de 14.4 Kbits/s au réseau Ethernet ou ATM.

C'est sur cette problématique de gestion de QdS dans le contexte du Web que nous nous sommes penchée. Notre application se veut être un outil d'accès à des documents multimédia via le Web. Elle offre une fonction de négociation de QdS où

l'utilisateur dispose d'une interface pour configurer les paramètres qui influent sur la qualité de présentation des documents. Les documents sont disponibles sous plusieurs versions correspondant à différentes QdS. En fonction des valeurs des paramètres choisies par l'utilisateur et des contraintes de composantes du système, une négociation de QdS est menée. Celle-ci a pour objet de choisir parmi les versions disponibles, celle qui convient le plus à l'utilisateur et qui tient compte des contraintes du système.

Notre travail se situe dans le cadre d'un projet du CITR (Canadian Institute for Telecommunications Research) où un prototype permettant l'accès, via un réseau à haut débit, à des documents multimédia répartis sur divers sites a été développé. Des mécanismes de gestion de ressources pour permettre la négociation et l'adaptation de la QdS ont été développés et utilisés dans le prototype.

Le but de notre projet est de porter ces mécanismes dans l'environnement du Web. En d'autres termes, il s'agit de produire une version adaptée au Web du prototype existant. Deux versions du système ont été produites. Des extensions du langage HTML (HyperText Markup Language) ont été utilisées pour coder les informations nécessaires à la négociation de la QdS.

Mots clés : Négociation de Qualité de Service, Application multimédia, World Wide Web, HTML.

Remerciements

Je tiens à remercier tout d'abord mes superviseurs Rachida Dssouli et Jan Gecsei pour leur disponibilité (malgré leur année sabbatique) et leurs précieux conseils dans les choix d'orientation et prises de décision tout au long de cette maîtrise.

J'adresse aussi mes remerciements à Grégor v. Bochmann qui m'a donné l'opportunité de prendre part au projet CITR et a également suivi mon travail.

Je remercie Lucie Lévesque pour son amabilité ainsi que tous mes collègues du laboratoire Téléinformatique pour la bonne ambiance de travail.

Je remercie mes collègues du projet CITR, en particulier David Evans, Dwight Makaroff, Roland Mechler, Abdelhakim Hafid et Mohamed Salem pour les échanges qui furent indispensables à la réalisation de mon travail.

Je remercie le Ministère de l'Éducation du Québec et le CITR qui m'ont offert le soutien financier nécessaire pour mener à bien ce programme de maîtrise.

Je tiens à remercier mes parents et mes amis qui m'ont soutenue de diverses manières.

Enfin, ma reconnaissance infinie à Gilles qui, malgré la distance, m'a encouragée, conseillée et comblée de son attention tout au long de cette maîtrise.

Table des matières

Liste des tableaux

Liste des figures

Liste des sigles et abréviations

API	Application Programming Interface
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband Integrated Services Digital Network
CGI	Common Gateway Interface
CITR	Canadian Institute of Telecommunications Research
CMFS	Continuous Media File Server
DBMS	Database Management System
DTD	Document Type Definition
FTP	File Transfert Protocol
QdS	Qualité de Service
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
MIME	Multipurpose Internet Mail Extensions
RSVP	Resource ReSerVation Protocol
RTP	RealTime Transport Protocol
SGML	Standard Generalized Markup Language
TCP	Transport Control Protocol
UDP	User Datagram Protocol
URL	Uniform Ressource Locator
VDP	Video Datagram Protocol
W3C	World Wide Web Consortium
WWW	World Wide Web

I. Introduction

Ce chapitre introductif donne un aperçu des questions abordées dans le mémoire. Nous introduisons la notion d'applications multimédia distribuées et présentons les questions de gestion de QdS impliquées par celles-ci. Nous exposons ensuite les problèmes relatifs à leur intégration dans le World Wide Web (WWW).

I.1 Les applications multimédia distribuées et la gestion de QdS

Le terme multimédia réfère à la combinaison d'une forme variée d'informations incluant le texte, les données graphiques, l'image, le son et la vidéo. La manipulation de ces informations multimédia dans les applications donne naissance aux applications multimédia. Les applications multimédia distribuées sont celles qui opèrent dans un environnement réseau. Un tel environnement doit supporter la communication multimédia c'est-à-dire la transmission en temps réel d'informations multimédia. Les applications multimédia distribuées doivent donc être en mesure de gérer et de délivrer de bout en bout des données multimédia satisfaisant un certain niveau de QdS (Qualité de Service). Ceci demande une grande vitesse de la part des stations de travail et un débit élevé de la part du réseau pour transmettre ces données en temps réel. L'évolution des applications multimédia distribuées est justement rendue possible par l'accroissement de la puissance des stations de travail, les nouvelles techniques de compression vidéo et surtout l'apparition des réseaux à hauts débits. Parmi ces réseaux, on cite le FDDI (Fiber Distributed Data Interface) dans la catégorie des réseaux locaux offrant une largeur de bande de 100 Mbps. Le réseau numérique à intégration de service (B-ISDN, Broadband Integrated Digital Network) offre une intégration de la voix, de la vidéo et des données informatiques sur une même infrastructure avec une largeur de bande de quelques centaines de Mbps tandis que les réseaux à longue distance précédents se limitent à quelques dizaines de Kbps. Le réseau ATM (Asynchronous Transfer Mode) offre quant à lui des possibilités de réservation de ressources pour satisfaire les contraintes liées à la

transmission des données. Les différentes formes de média impliquées dans une application multimédia peuvent être regroupées en deux catégories :

1) Les données statiques qui sont les données informatiques ordinaires (textes et images)
2) Les données à flot continu comme l'audio et la vidéo qui ont une dimension temporelle.

La deuxième catégorie de média a besoin d'un système de communication très performant (hauts débits, etc.) et d'une gestion au niveau de l'application capable de préserver les contraintes temporelles. Le besoin de performance varie d'un média à l'autre. La transmission de la voix demande typiquement moins de ressources que celle de la vidéo tandis que la voix est plus sensible à la perte de données. D'autre part, la nature distribuée de ces systèmes implique que les différentes composantes n'ont pas les mêmes caractéristiques ni les mêmes performances. Ainsi, on ne peut s'attendre à ce que les serveurs et les clients du système offrent les mêmes performances. De plus les réseaux qui relient ces machines peuvent offrir différents niveaux de QdS (largeur de bande, délai etc.).

La QdS peut être définie comme un ensemble de paramètres techniques ou autres permettant de mesurer la performance des composantes du système. Pour un réseau de transmission de données, un exemple de paramètre de QdS est le débit du réseau. Ces paramètres influent en général sur la qualité de la présentation telle que perçue par l'utilisateur. La nécessité de gérer la QdS dans les applications multimédia distribuées vient non seulement du fait de la diversité des niveaux de performance offerts par les composantes du système mais surtout des contraintes temporelles de certains media. La capacité de prendre en compte la variété de performance offerte par les composantes du système et la capacité de s'adapter aux fluctuations de ces niveaux de performances sont primordiales pour les applications multimédia distribuées. Pour qu'une application multimédia distribuée puisse s'adapter aux variations de la QdS causée par le réseau et les serveurs de média, il faut développer des méthodes de gestion des ressources pour la négociation et l'adaptation de la QdS dans un environnement distribué.

Dans un tel environnement où divers utilisateurs possèdent différents équipements avec des capacités différentes, il est d'autre part important de prendre en compte la satisfaction de chaque usager. C'est justement sur cette problématique de gestion de QdS que le projet initial du CITR (Canadian Institute of Telecommunications Research) [Haf96], dont notre projet de maîtrise est une suite, s'est penché.

I.2 Les applications multimédia dans l'environnement du WWW

Le WWW vu d'une manière simplifiée comme étant une interface d'accès aux ressources disponibles sur le réseau Internet est de plus en plus utilisé pour le transfert d'informations de diverses natures. Cet outil est désormais le support pour différents trafics tels que les transactions commerciales, la téléphonie, l'enseignement à distance, la téléconférence, la publicité etc. De nos jours, le Web est un des systèmes les plus populaires utilisés pour la transmission de documents multimédia incluant le texte, le son et la vidéo.

Le WWW a été développé à l'origine pour permettre l'accès à des documents statiques tels les fichiers Ascii et l'image par la transmission complète des fichiers. Le protocole utilisé pour la transmission des données entre les clients et les serveurs Web est HTTP (Hypertext Transfer Protocol) qui est basé sur le protocole TCP (Transport Control Protocol) pour le transport fiable de l'information. Mais la transmission des données sensibles au temps n'est pas une tâche triviale étant donné le service "meilleur effort" qui caractérise l'Internet aujourd'hui. Le meilleur effort se traduit par le fait qu'aucune garantie n'est offerte sur l'utilisation des ressources disponibles sur le réseau, ni aucune garantie en terme de performance telle que le délai maximum de délivrance, largeur de bande minimum etc. Les ressources disponibles sont partagées au mieux entre les utilisateurs soumettant ainsi le réseau aux instabilités hors du contrôle des serveurs et des clients. Le protocole HTTP basé sur ce service s'est révélé inadéquat pour les besoins des médias à flux continu que sont le son et la vidéo. Cette inadéquation s'explique par les raisons suivantes :

- Le protocole HTTP n'est pas conçu pour le transfert en mode continu. Pour des

données audio et vidéo, le temps de transfert peut être trop long pour être pratique. Les fichiers multimédia sont en effet très lourds. La version numérisée d'un extrait musical de 30 secondes peut facilement atteindre une taille de 3 Mo. Or, la plupart des applications telles que la vidéophonie requièrent que l'information soit délivrée en temps réel. Autrement dit, le délai de délivrance de l'information doit être suffisamment limité pour être acceptable pour l'utilisateur.

- Le protocole TCP est lui aussi inadéquat pour la vidéo et le son. En effet, TCP impose son propre contrôle de flux et de fenêtrage sur les flots de données. Ceci détruit les relations temporelles entre les trames de vidéo et les paquets d'audio.

- Les données audio et vidéo sont plus tolérantes aux pertes de trames de données que ne le sont les données informatiques traditionnelles même si cela peut affecter la qualité de l'affichage. En effet certains paquets de vidéo ont moins d'impact sur la qualité de la présentation que d'autres. Ceci rend inutiles les retransmissions systématiques du TCP dans certains cas. De plus, cette retransmission introduit des délais gigue ("jitter") entre différentes trames d'un même flot de données (audio ou vidéo) ou entre différents flots de données synchronisés (audio et vidéo). La gigue se définit comme étant la variation du délai introduite par le réseau ou le serveur de données. Par exemple, des paquets d'audio envoyés de la source avec un décalage de 20ms entre chaque paquet peuvent arriver à destination avec des décalages différents dus au temps d'attente dans les routeurs.

D'autre part, la nature temps réel des données à flux continu ainsi que le besoin de ressources qu'implique leur manipulation rend importante la gestion de QdS dans les applications multimédia distribuées en général et dans l'environnement du WWW en particulier. Fondamentalement, l'Internet est un réseau très hétérogène offrant un service meilleur effort et composé de sous réseaux ayant des capacités très diversifiées. Cette hétérogénéité affecte inévitablement la qualité de l'information

délivrée par le réseau. La notion de QdS bout en bout intervient dans la nécessité de gérer cette diversité et cette hétérogénéité pour accommoder divers utilisateurs tout en essayant d'offrir un niveau optimal de QdS. Cette gestion de QdS est avant tout utile pour rendre transparente à l'utilisateur l'instabilité du réseau.

En somme, le constat est que, jusqu'à un passé très récent, le Web et les protocoles associés ne supportaient pas convenablement le trafic de données à flot continu. Les données étaient plutôt téléchargées entièrement avant d'être affichées par un "helper application". L'intégration de contenu multimédia temps réel dans le Web lance donc un nouveau défit aux chercheurs. Parmi les problèmes généralement évoqués, on peut citer les suivants :

1) Besoin de nouveaux protocoles pour la transmission des documents multimédia.

- Ces protocoles doivent prévoir les mécanismes pour délivrer l'information en temps réel, autrement dit, minimiser le délai d'attente de l'utilisateur avant que l'information lui soit délivrée. Les objectifs sont multiples. Vu le trafic énorme engendré par le transfert des données multimédia en temps réel, des mécanismes de contrôle sont nécessaires pour éviter que les ressources du réseau soient inondées et préserver ainsi la disponibilité du réseau autant que possible. D'autre part, sans aucune garantie sur la qualité de présentation des données audiovisuelles, celles-ci perdent leur utilité.
- Ces protocoles doivent offrir des facilités d'accès à des portions données d'un film vidéo avec des fonctions telles que "fast foward", "rewind" et "seek" offertes par les systèmes audio/vidéo traditionnels. Pour offrir cette facilité, le protocole doit permettre la communication bidirectionnelle contrairement aux protocoles FTP (File Transfert Protocol) ou HTTP.

2) Synchronisation des différents flots de données qui composent le document multimédia.

3) Besoin d'extension du langage HTML pour permettre la description des

documents multimédia et leur insertion dans des pages Web.

4) Gestion de la Qualité de Service dans l'environnement du WWW afin d'accommoder l'utilisateur face à :

- L'instabilité du réseau (variation de la disponibilité des ressources par exemple),
- L'hétérogénéité des liens d'accès à l'Internet,
- La diversité des capacités des machines clientes et les besoins accrus et spécifiques en ressources machines des données multimédia.

Heureusement, divers groupes de développement de standards ou d'applications pour le Web travaillent activement dans ce domaine et la situation a beaucoup évolué depuis le début de ce projet. Dans le chapitre 2, nous ferons un survol de quelques systèmes qui ont été développés jusque là et des standards proposés ou naissants. Ces systèmes implantent en partie des solutions aux problèmes évoqués ci-dessus. Pour ce qui concerne notre projet de maîtrise, nous nous sommes intéressés aux points 3) et 4). Les points 1) et 2) ont été couverts dans le projet initial du CITR que nous allons décrire ci-dessous.

I.3 Mise en contexte du projet

Ce projet de maîtrise se situe dans le contexte du projet du CITR où un prototype permettant l'accès à distance à des nouvelles sur demande a été développé. Ce système permet l'accès à distance à une base de données multimédia via un réseau ATM ou Ethernet. Le système offre également une fonction de gestion de la QdS. Une interface utilisateur permet à l'utilisateur de sélectionner le document de son choix. Les documents sont stockés sous plusieurs versions chacune ayant des qualités de présentation (qualité du son, résolution de la vidéo etc.) différentes et pouvant être sur différents sites. La gestion de la QdS permet la sélection d'une configuration optimale du système qui satisfait aux exigences de l'utilisateur et tient compte des capacités des systèmes impliqués. Elle permet également une adaptation automatique en cas de changement (congestion par exemple) dans le réseau. Le prototype qui a été développé et qui est appelé prototype de Nouvelles-sur-Demande est une intégration

de différentes composantes logicielles développées par différents sous-groupes qui collaborent dans le projet CITR : La base de données multimédia distribuées DBMS (Database Management System) développée à l'Université d'Alberta, le serveur de données à flux continu CMFS (Continuous Media File Server) développé à l'Université de Colombie Britannique, le module de synchronisation des media de l'Université d'Ottawa et le module de négociation de QdS développé à l'Université de Montréal. Nous décrirons ce système plus en détail dans le chapitre 3.

Le but de ce projet de maîtrise est de produire une version du prototype de Nouvelles-sur-Demande qui sera compatible avec le Web. (Dans la suite nous allons nous référer à ce prototype comme le prototype initial du CITR). Il s'agit essentiellement de porter le mécanisme de négociation réalisé pour le projet du CITR dans l'environnement du Web. En effet, dans le cadre du projet du CITR, les concepts de base pour la gestion de QdS dans les applications multimédia distribuées ont été développés. Il a semblé intéressant d'étendre ces concepts dans l'environnement particulier du WWW. L'idée de base, comme dans le prototype initial du CITR, est de disposer de choix de versions alternatives provenant de diverses sources et d'en choisir une selon les préférences de l'utilisateur et l'état du réseau. La conception du présent projet s'appuie essentiellement sur les idées de base du prototype initial du CITR, en particulier en ce qui concerne la structure des documents multimédia, les paramètres de QdS etc. Ce projet hérite également de certaines composantes logicielles développées dans le prototype initial du CITR comme le protocole de transport, le serveur de données continues conçu pour le stockage approprié et le transport en temps réel des données multimédia. La négociation de QdS en utilisant différentes versions (avec diverses qualités) est d'autant plus intéressante dans l'environnement du Web que le réseau Internet est assez hétérogène. Les liens d'accès au réseau varient largement en capacité. Aussi, l'infrastructure du Web doit accommoder aussi bien l'usager utilisant un modem de 28.8 Kbps que la compagnie qui dispose de liens ATM ou Ethernet. En d'autres termes, le présent projet prévoit de rendre accessible via le Web, des documents multimédia répartis sur différents serveurs hétérogènes incluant les serveurs Web, CMFS et d'autres serveurs de media

continu tels que Vosaic et RealAudio et d'intégrer la gestion de QdS. Les fonctionnalités attendues du système résultant sont :

- Naviguer parmi les documents HTML disponibles et en choisir un comme on peut choisir un document multimédia dans le prototype initial du CITR.
- Négocier la QdS du document multimédia que l'utilisateur désire voir.
- Déclencher la présentation du document choisi avec la QdS convenue.

Pour réaliser ces fonctionnalités nous avons :

- Défini les extensions du langage HTML nécessaires pour encoder les méta-données des documents multimédia et pour les intégrer dans une page Web. Les documents multimédia sont des objets composites dont les constituants (tels que la vidéo, l'audio, l'image, le texte etc.) ont des caractéristiques ou paramètres de QdS. Ces informations sur le document multimédia que nous appelons *méta-données* sont utiles à la gestion de la QdS. Elles permettent par exemple de spécifier qu'une composante audio ou vidéo est disponible sous plusieurs versions et aussi d'indiquer les caractéristiques de chaque version. Cette extension HTML proposée permet à un auteur d'une page HTML d'offrir des représentations alternatives d'un même document multimédia à l'usager.
- Défini et implanté un analyseur simplifié pour lire les méta-données incluses dans une page HTML et créer la structure de document correspondant.
- Défini et implanté une interface utilisateur pour la négociation de la QdS. Ceci donne la possibilité à l'utilisateur de spécifier la qualité de présentation (ou QdS) de média qu'il désire. On associe également un coût à différentes qualités de présentation. Vu que l'interface usager du protocole de négociation du projet initial nous a paru un peu complexe, nous en avons défini une autre, le but étant d'être le plus proche possible de l'utilisateur. Nous avons également introduit la notion de priorité qui permet à l'usager de spécifier l'importance qu'il accorde aux paramètres de QdS des différents média ou l'importance qu'il accorde à chaque média. Cette notion de priorité nous permet de classifier les différentes offres que nous avons pour un document multimédia. Nous choisissons à tout moment la meilleure alternative que le système peut supporter.

- Défini et implanté un protocole de négociation de la Qualité de Service. Pour satisfaire les préférences de l'utilisateur tout en tenant compte de l'état actuel du réseau et des serveurs, un tel protocole est utile pour permettre de trouver le meilleur compromis possible pour une session de consultation.
- Défini une interface d'accès aux différents serveurs que nous avons intégré dans notre du système.

Comme la technologie du Web supporte déjà assez bien le texte et l'image, nous avons décidé de lui laisser la charge de ces deux composantes tandis que la négociation de QdS porte seulement sur les composantes audio et vidéo des documents multimédia. L'usager du Web qui consulte un document multimédia de ce projet accède directement aux composants textes et images de la manière habituelle dont il accède aux informations sur le Web. Les composantes audio et vidéo sont alors accessibles par un lien hypertexte. En suivant un tel lien, l'usager se voit offrir une interface où il peut mener une négociation de façon explicite sur la QdS à utiliser ou laisser l'application négocier de manière implicite. Après la négociation, l'application présente à l'utilisateur l'offre (combinaison de versions de monomédia continus qui composent le document) que le système peut supporter. L'adaptation se fait en considérant la meilleure offre suivante dans la liste ordonnée des offres.

Dans la recherche de solutions, nous avons expérimenté une première approche qui utilise la négociation de QdS du prototype initial du CITR. Cette approche nous a permis de porter directement le prototype existant dans l'architecture du Web. Dans notre deuxième approche, nous avons implanté une version révisée de l'agent de négociation de QdS dont l'aperçu est donné plus haut dans la présente section.

Ce mémoire est composé de 6 chapitres. Le chapitre 2 présente un état de l'art sur les applications multimédia dans l'environnement du WWW. Le projet initial du CITR dont notre projet de maîtrise est une suite est résumé dans le chapitre 3. Les chapitres 4 et 5 portent sur le travail accompli dans ce projet maîtrise. Le chapitre 4 décrit notre première de solution tandis que le chapitre 5 donne une description détaillée de la deuxième approche. Le chapitre 6 conclut le travail.

II. Les applications multimédia dans l'environnement du WWW

II.1 Aperçu des applications multimédia sur le Web

Plus qu'une interface, le WWW est une technologie qui offre une interface attractive de navigation à travers l'immense ressource de l'Internet. C'est en quelque sorte une méta-interface qui incorpore toutes les autres interfaces d'accès aux ressources disponibles sur l'Internet (FTP, Archie, Wais etc.) [Dec94]. Inventé en 1989 par Tim Berners-Lee [TIM], le Web a été conçu au départ pour permettre de partager des documents Ascii dans un système d'hypertexte où des documents peuvent être raccordés facilement à d'autres à l'échelle mondiale. Le caractère multimédia du Web se limitait alors à l'inclusion d'images statiques.

De nos jours, l'inclusion de l'audio et de la vidéo dans le Web est en pleine évolution. Alors qu'au départ, le son et la vidéo sont téléchargés au complet par les navigateurs avec un délai de transmission souvent inacceptable, plusieurs outils nous sont proposés aujourd'hui pour jouer ces média en temps réel. Ces outils sont pour la plupart implantés comme extension aux fonctionnalités des navigateurs et prennent le nom de "helper application" ou de "plug-in" selon leur mode de fonctionnement. Parmi les systèmes commerciaux et expérimentaux les plus visibles, on peut citer RealAudio [Pro97], Vosaic [Zhi95] etc. L'introduction par Sun du langage Java qui permet d'écrire des programmes qui peuvent être téléchargés et exécutés sur une machine à distance favorise également l'évolution du Web [Bol96].

Les *helper applications* sont des programmes externes que les logiciels navigateurs du Web utilisent pour permettre l'affichage de différents types de formats de média qu'ils ne supportent pas traditionnellement. A chaque helper est associé un type MIME (Multipurpose Internet Mail Extensions) avec une extension qui lui est associée. Tout fichier contenant un média de ce type MIME doit porter l'extension correspondante. Lorsqu'un navigateur rencontre un fichier de ce type, c'est-à-dire un

fichier ayant une extension donnée, il transmet la donnée au helper qui s'occupe de l'afficher. Le terme *External Viewer* est utilisé pour désigner le même mécanisme.

Les *plug-ins* [Net96] sont conçus spécialement pour s'intégrer au navigateur Netscape afin d'en étendre la capacité. Ils sont également associés à un ou plusieurs types MIME. Ce qui permet au navigateur de les activer. Au démarrage, Netscape énumère les plug-ins disponibles pour la plate-forme en question et enregistre chaque plug-in pour le type MIME associé. Lorsqu'il rencontre une donnée d'un type MIME enregistré pour le plug-in, il charge le plug-in en mémoire (si cela n'avait pas été fait auparavant) et crée une nouvelle instance du plug-in. Les plug-ins communiquent avec Netscape via un API (Application Programming Interface). Par contre la seule communication que le navigateur a avec un helper est de lui passer, au moment de l'appel, les arguments nécessaires à son fonctionnement.

Le type *MIME* est un standard utilisé dans l'Internet pour encoder la grande variété de types de fichier sur les différentes plates-formes qui composent l'Internet. Le standard MIME permet une représentation des données qui est indépendante de toute plate-forme. Les types MIME sont utilisés par les serveurs Web et les navigateurs pour identifier les types de données qui sont transférées entre ces deux systèmes. Pour une définition plus complète du standard MIME, se référer à [Bor93].

Comme souligné plus haut, le Web n'était pas conçu pour supporter le trafic des données continues. De ce fait, beaucoup de propositions sont faites par les chercheurs pour inclure adéquatement les données multimédia dans l'environnement du WWW. Cela implique pour l'instant un grand nombre de solutions non compatibles. La plupart des approches utilisées remplacent le protocole sous-jacent utilisé dans la plupart dans transactions du Web par un protocole adapté à la transmission des données à flux continu. Dans le même temps les travaux du W3C[1] (World Wide Web Consortium) et de l'IETF[2] (Internet Engineering Task Force) concourent à proposer

---

[1] Le W3C a été fondé en 1994 pour développer des protocoles d'intérêt commun pour l'évolution du WWW [Ber97].

[2] L'IETF est une communauté internationale qui regroupe les concepteurs, vendeurs et chercheurs dans le domaine des réseaux d'ordinateurs particulièrement intéressés à l'évolution de l'Internet [IETF].

des solutions standards au problème d'extension du Web, des protocoles et des langages associés afin de faciliter l'intégration des applications multimédia temps réel dans le Web. Un des résultats obtenus est la spécification du protocole RTP (Realtime Transport Protocol) [Hen97] développé par l'IETF comme alternative au protocole HTTP pour la transmission en mode continu des données multimédia sur l'Internet. De plus le lancement de LiveMedia par Netscape est un pas vers la recherche de solutions intégrées. LiveMedia est une architecture ouverte, basée sur des standards pour permettre un accès facile aux nouvelles applications Internet telles que vidéo et audio sur demande, conférence vidéo, téléphonie Internet. Elle est basée sur le protocole RTP et des standards de compression audio/vidéo comme MPEG, h.261 et GSM. Plusieurs compagnies dont Progressive Networks (RealAudio), Adobe Systems, Silicon Graphics etc. lui ont donné leur support. Dans la section suivante nous allons présenter les défis soulevés par la problématique de l'inclusion du multimédia dans l'environnement du Web, les technologies de base développées par divers groupes de recherche et compagnies. Les questions examinées concernent les protocoles de transmission de données et le langage HTML. La section 3 décrit quelques systèmes existants en soulignant les différences et similarités avec notre projet.

II.2 Besoins et solutions

II.2.1 Protocole de transport

II.2.1.1 Transmission du son et de la vidéo via le protocole HTTP [Bol96]

Jusqu'à très récemment, le son et la vidéo étaient stockés sur les serveurs Web et transmis comme les autres documents. La figure 1 montre l'architecture des systèmes utilisant les serveurs Web comme serveurs de données à flux continu. L'accès aux données se fait via un client Web utilisant le protocole HTTP. Étant donné que les navigateurs courants ne supportent pas la présentation de ces données à l'utilisateur, un programme externe, helper application est requis pour la présentation des média.

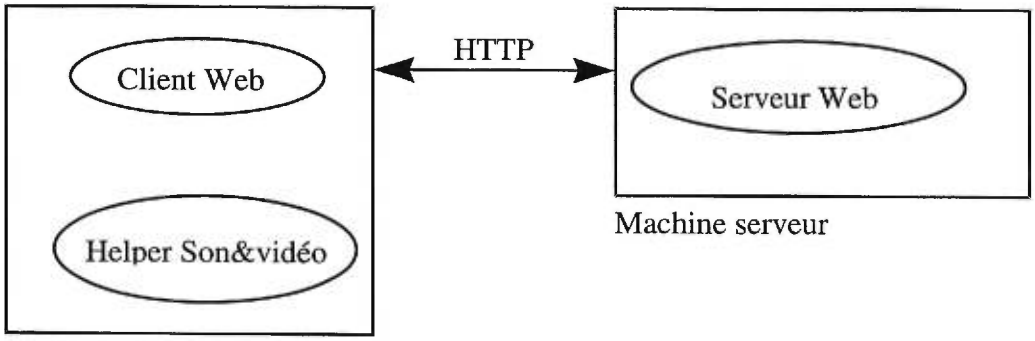

Figure 1 : Architecture d'un système Web pour accéder aux données son et vidéo via HTTP

HTTP est essentiellement un protocole de transfert de fichiers. Le tableau 1 montre les étapes requises dans le transfert d'un ficher audio "son.au" stocké sur un serveur Web. Le client Web fait une requête "Get" au serveur Web qui après avoir reçu la requête envoie le fichier dans un "HTTP response message" (RFC2068) dont l'entête contient le type MIME du fichier. Le client utilise cette information pour lancer le helper application approprié qui se charge alors de présenter l'audio.

Étape	Client Web	Message sur le réseau	Serveur Web
1	L'utilisateur clique sur un lien		
2	Émet une requête pour obtenir le document	GET/son.au	
3	Mise en tampon du ficher. Temps d'attente long pour l'utilisateur	HTTP 1.0 200 OK ... content-type: audio/basic ... 12A5 89C3 7344 A389 ... (long) ...	Transmet un message contenant le fichier son.au

		700 FF34 C3C5 3439	
4	Joue le fichier son.au		

Tableau 1 : Transfert d'un fichier de son via le protocole HTTP

L'inconvénient de cette approche vient du délai de transfert qui est trop long. A première vue une technique similaire à celle utilisée par certains navigateurs qui affichent les documents HTML de manière incrémentielle pourrait être une solution. Cependant, l'expérience montre que TCP est inadéquat aux données continues, surtout lorsque le réseau Internet est utilisé comme réseau sous-jacent où la fiabilité (perte de paquet etc.) n'est pas garantie. Notons que le protocole TCP peut bien se montrer performant dans un réseau local fiable. L'inadéquation de TCP comme souligné dans l'introduction se résume par le fait que les mécanismes de contrôle de flux et de retransmission qu'il utilise brisent les relations temporelles qui caractérisent les données continues et introduisent des délais gigue dans les flots de données résultant en des vides dans la vidéo ou son inintelligible. Le tableau 2 qui suit illustre le temps estimé pour la transmission de quelques formats de fichier en utilisant cette approche.

Format	Débit	Taille d'un clip de 5 mn	Temps de réponse sur un modem de 14.4 Kbit/s
Fichier vidéo MPEG	1.5 Mbit/s	56.4 Mo	8h
Fichier audio (.au)	64 Kbit/s	2.4 Mbyte	22mn
Fichier audio GSM	13 Kbit/s	412.5 Kbyte	4 mn

Tableau 2 : Temps de réponse pour la transmission avec HTTP d'un clip de 5mn dans une application de nouvelle-sur-demande

II.2.1.2 Protocole de transfert en mode continu ("Streaming Protocol")

Comme nous l'avons montré ci-dessus, un premier besoin est celui de protocoles spécialisés dans la transmission de données continues, assurant un transfert

en mode continu (*Streaming*) plutôt qu'un téléchargement complet avant la lecture. Le but du transfert en mode continu est de commencer la présentation des média le plus tôt possible, c'est-à-dire dès que le client dispose suffisamment de données pour être affichées. Ceci réduit le temps d'attente de l'utilisateur. Habituellement ces protocoles sont construits sur le protocole UDP (User Datagram Protocol). Le protocole TCP est un protocole orienté connexion qui offre un service de transport fiable de bout en bout. Les pertes de paquets sont corrigées par une retransmission des paquets perdus. Le protocole UDP par contre envoie des paquets de données de manière indépendante sur le réseau sans s'assurer que les données arrivent effectivement à destination. C'est en général le protocole construit sur UDP qui se charge du contrôle d'erreur en tenant compte du média particulier. Plusieurs approches sont possibles pour construire un système de transport adapté aux besoins des données à flot continu :

1) Réservation de ressources

Une approche possible est de se baser sur des garanties de performance pour maintenir une certaine QdS. Cette approche nécessite que des mécanismes de contrôle d'admission, de réservation de ressources soient implantés dans le réseau. En effet pour offrir une garantie (largeur de bande minimum, délai maximum), il peut être nécessaire qu'une certaine quantité de ressources (largeur de bande, tampon etc.) soit réservée pour une connexion donnée. Étant donné que les ressources du réseau sont limitées, il n'est pratiquement pas possible de répondre à toutes les demandes de réservation de ressources. En vu de maintenir la charge du réseau à un niveau tel que toutes les garanties de QdS puissent être assurées, l'architecture du réseau doit prévoir un algorithme de contrôle d'admission qui détermine quelles demandes de réservation de ressources peuvent être agrées ou non. Le mécanisme de contrôle d'admission peut être aussi implanté au niveau des serveurs dont la performance a une influence directe sur la QdS bout-en-bout.

Le protocole RSVP (Realtime ReserVation Protocol) [Zap96], dont la conception est motivée par les besoins des applications multimédia sur l'Internet, est

basé sur ce concept. Il permet la réservation de ressources au niveau des routeurs de l'Internet. RSVP propage la réservation de ressources à travers le réseau, visitant chaque nœud impliqué dans la transmission de la donnée pour la laquelle la réservation de ressources a été initiée par l'application réceptrice. Il n'offre pas des fonctions de routage mais est conçu pour être implanté au-dessus des protocoles de routage existants.

Parmi les groupes de recherches qui travaillent sur la question d'inclusion des données à flux continu dans l'architecture du Web, on peut citer le groupe de Lakshman (University of Kentucky) dont les propositions décrites dans l'article [Lak96] se basent sur la même approche, c'est-à-dire la réservation de ressources au niveau des routeurs du réseau. Le prototype qu'il propose et que nous allons décrire dans la section 3 utilise une implantation étendue du protocole HTTP qui supportent des requêtes avec réservation de ressources. A long terme, le développement de la technologie ATM va certainement favoriser cette solution puisque la réservation de ressource fait partie intégrante de cette technologie.

2) Contrôle de la transmission

Une autre approche possible est de "masquer" les imperfections du réseau. Les solutions utilisant cette approche prennent en compte un ou plusieurs aspects suivant de la transmission des média :

a) Contrôle de perte de paquets

Tandis que dans certaines applications multimédia, on ne réagit pas systématiquement aux pertes de paquets par une retransmission automatique, d'autres optent pour une retransmission sélective et ce en fonction de codage de la vidéo ou de l'audio. En somme, l'idée de base est d'être plus tolérant aux pertes de paquets. Le système Vosaic utilise au niveau du client une telle approche en étant sélectif dans les demandes de retransmission en cas de perte de paquet.

b) **Contrôle du délai**

Outre la perte de paquets, un autre inconvénient du réseau est l'introduction de délai gigue entre les paquets de données audio et/ou vidéo. La gigue est souvent corrigée en introduisant au niveau du récepteur un tampon de taille appropriée pour "amortir" les variations de délai. En effet, les paquets dans ce cas ne sont pas utilisés par le client aussitôt à leur arrivée mais après un certain délai. Ce délai peut être déterminé en fonction d'un étiquetage temporel ou *timestamping* (instant où la donnée a été échantillonnée) marqué sur la donnée à la source. Le protocole RTP offre cette fonction d'étiquetage temporel.

c) **Contrôle du débit de transmission et adaptation dynamique**

L'idée de base est d'adapter la vitesse de transmission des média de manière dynamique en tenant compte de la largeur de bande disponible pour éviter la congestion du réseau et la dégradation brusque et perceptible de la qualité de la présentation. Un des avantages de cette approche est que cela ne demande aucun support spécial (allocation de ressources, contrôle d'admission) au niveau du réseau. Elle peut donc être appliquée sur l'Internet tel qu'il est actuellement. Différents mécanismes de contrôle sont possibles pour réaliser cette adaptation [Bol94]; en général, ils s'appuient sur les informations reçues en écho sur l'état du réseau.

Un tel protocole a été implanté dans l'architecture de Vosaic. C'est le protocole VDP (Video Datagram Protocol). L'objectif visé dans la conception de VDP est donc de faire bon usage des ressources (largeur de bande au niveau du réseau et CPU au niveau du client) disponibles non pas de les réserver. VDP utilise un algorithme d'adaptation pour trouver la largeur de bande optimale. Chaque connexion VDP utilise deux canaux: Un canal non fiable pour la transmission des données non critiques et un canal fiable pour la transmission des informations de contrôle ("play", "stop" etc.). Notons que Vosaic utilise un mécanisme de contrôle d'admission au niveau de son serveur. Une fois que le contrôleur d'admission accepte une requête, le serveur attend que l'utilisateur envoie la commande play pour commencer la

transmission. L'algorithme d'adaptation adapte de manière dynamique le débit au niveau du serveur non seulement en fonction des conditions du réseau mais aussi de la charge de la machine cliente. Le client informe le serveur de l'état de la connexion par des feed-back qui sont de deux types : le nombre de trames larguées par la machine cliente du fait de sa lenteur et le nombre de trames perdues au niveau du réseau. Cette information est fournie par la partie client du protocole qui surveille l'application et met à jour le taux de perte si celle-ci ne reçoit pas assez vite les trames. Une adaptation est initiée lorsque le taux de perte dépasse un certain seuil.

Le serveur de données continues implanté dans le cadre du projet CITR est également basé sur le contrôle de la transmission plutôt que la réservation de ressources dans le réseau. Dans ce système, la réservation de ressource se fait par contre au niveau du serveur de données. La particularité de l'approche sous-jacent au système de serveur CMFS se trouve dans le type d'information utilisé par le serveur pour adapter le débit de la transmission. Le serveur dispose en effet des informations précises sur la présentation des média. Ceci indique le débit utile pour chaque seconde de la présentation. Le serveur dispose également des informations sur les capacités de la machine cliente en terme de taille de tampon disponible pour recevoir les données, vitesse de traitement des données etc. Sur la base de ces informations, le serveur peut à tout moment utiliser la largeur de bande maximale disponible sur le réseau pour alimenter adéquatement les tampons au niveau de la machine cliente. Le mécanisme de contrôle de flux utilisé dans le système CMFS est détaillé dans [Neu96].

Le standard RTP qui a été défini par l'IETF pour la transmission en temps réel des données multimédia sur l'Internet ne supporte pas non plus la réservation de ressources et n'offre pas non plus des garanties de QdS. Ce standard a été conçu pour les besoins des applications multimédia conversationnelles aussi bien que présentationnelles. RTP offre des fonctions de transport bout-en-bout incluant : identification des types de charge, numérotation des paquets, étiquetage temporel, gestion de QdS de la transmission. RTP consiste en deux protocoles : le protocole **RTP** et le protocole **RTCP**.

Un paquet de donnée RTP consiste en une entête suivie par un "payload data"

(donnée utile ou charge contenue dans le paquet de donnée) qui peut être une trame vidéo ou des échantillons d'audio. Quelques champs importants de l'entête sont :

- Type de charge : le format de la charge RTP, par exemple le format H.261 pour la vidéo
- *Timestamp* (étiquetage temporel) : indique l'instant où le paquet de données a été généré. Ce champ est utilisé pour la synchronisation et le calcul de certains délais (gigue, aller-retour d'un paquet RTP).
- Numéro de séquence : Ce numéro est incrémenté de un chaque fois qu'un paquet de données est envoyé. Peut être utilisé par le récepteur pour calculer les taux de perte et détecter aussi les paquets qui ne viennent pas dans l'ordre.

La fonction principale du protocole RTCP est de véhiculer des "comptes rendus" sur la qualité de la transmission pour des fins de gestion de QdS. Ce n'est donc pas le protocole RTP qui effectue la gestion de la QdS mais elle offre des fonctions pour permettre aux systèmes impliqués de la réaliser. RTCP transporte deux types d'information : SR (Sender Report) et RR (Receiver Report). Les deux contiennent des statistiques de performance comme : le nombre de paquets perdus, le numéro de séquence le plus élevé qui a été reçu, des mesures sur le délai gigue ou le délai d'un aller retour etc. Plusieurs systèmes existent déjà qui utilisent leur propre implantation du protocole RTP. Le projet Fast Web [Fry96] que nous allons décrire dans la section 3.2 utilise ce protocole. La dernière version du système CMFS démontrée à la conférence annuelle du CITR (août 1997) utilise également ce protocole.

II.2.1.3 Intégration des protocoles de transfert en mode continu dans l'architecture du Web [Bol96]

Dans cette section nous allons décrire comment les protocoles de transfert en mode continu (pour la plupart utilisant le protocole UDP) peuvent être utilisés en combinaison avec un client HTTP (utilisant donc le protocole TCP). L'idée est de trouver un moyen pour commencer avec HTTP lorsque l'utilisateur clique sur un lien et de passer en mode continu lorsque les données à flux continu doivent être affichées.

La solution courante est d'ajouter une paire client/serveur multimédia au système Web. La figure 2 montre un exemple d'une telle architecture. Le client Web offre ses fonctions habituelles comme afficher les pages HTML. Le client multimédia (helper ou plug-in) se charge de jouer l'audio et/ou la vidéo et d'exécuter le streaming protocol avec le serveur multimédia. Le client peut offrir d'autres fonctions de contrôle telles que : "Play", "Stop", "Fast Foward" etc. Cette architecture répartie est transparente pour l'utilisateur qui ne voit en fait que le client Web. Les serveurs Web et multimédia peuvent éventuellement s'exécuter sur différentes machines. Ceci présente l'avantage de décharger la machine du serveur Web.

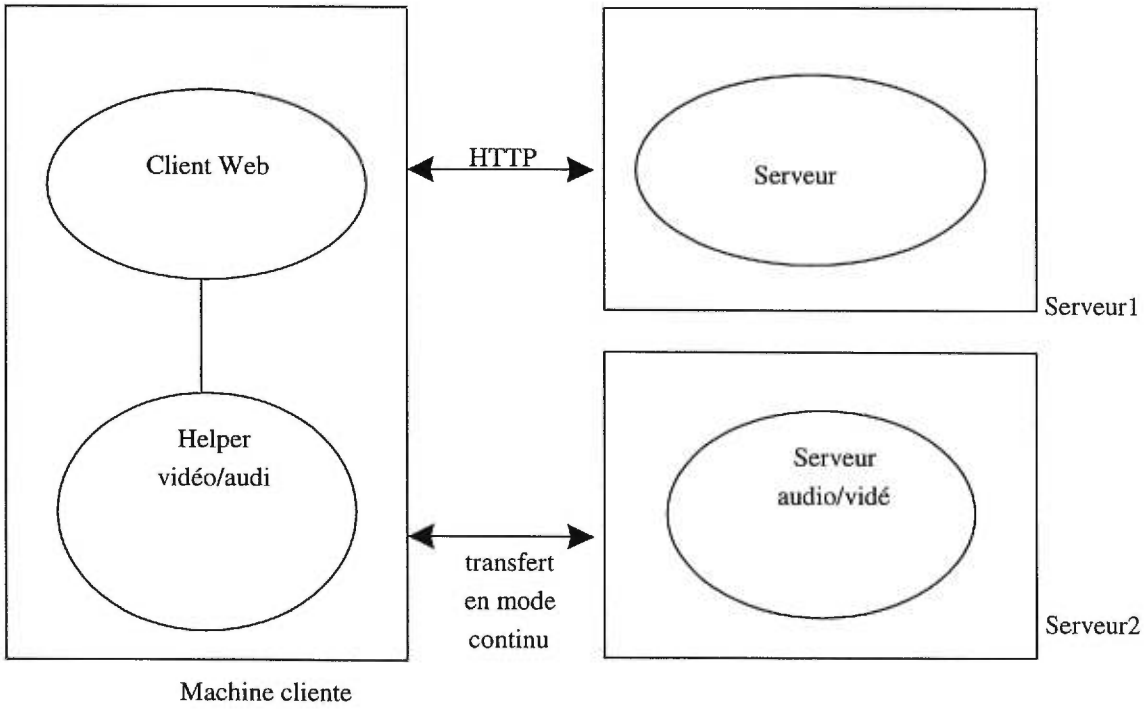

Figure 2 : Introduction d'un protocole en mode continu par addition d'une paire client/serveur

Le tableau 3 montre les étapes du fonctionnement d'une telle architecture pour retirer et présenter le fichier ″son.au ″. La différence essentielle avec le fonctionnement décrite sur la figure 2 réside dans l'étape 3. La réponse du serveur Web ne contient pas le fichier audio. Il contient plutôt les paramètres nécessaires au

client multimédia pour contacter le serveur multimédia et récupérer le fichier audio. Le client Web, se basant sur le type de contenu indiqué par le serveur dans sa réponse lance l'exécution du helper ou plug-in approprié qui se charge du reste.

Étape	Client Web	Réseau	Serveur Web
1	L'utilisateur clique sur un lien		
2	Émet une requête pour obtenir le document	GET/son.foo	
3	Reçoit le fichier paramètre	HTTP 1.0 200 OK ... content-type: application/x-foo ... 138.96.24.75 -f son.au	Transmet une réponse contenant les paramètres du helper pour accéder au fichier audio
4	Localise le helper et l'active avec ses paramètres		
	Helper	Réseau	Serveur audio
5	Se connecte au serveur audio, faite une requête pour avoir le fichier puis le joue	Transfert en mode continu	Transmet le fichier son.au

Tableau 3 : Transfert d'un fichier de son en utilisant un protocole en mode continu

II.2.2 Le langage HTML (HyperText Markup Language)

II.2.2.1 Définition

HTML [Rag97] est un langage simple de marquage qui permet de créer des documents hypertextes indépendants de toute plate-forme. Le langage définit un ensemble d'éléments qui permet de décrire la structure logique du document. HTML est une application de ISO 8879 :1986- SGML [Con97].

SGML (Standard Generalized Markup Language) est un métalangage qui décrit la structure logique d'un document en utilisant des marqueurs pour délimiter les éléments logiques du document. L'approche généralisée utilisée par SGML sépare la description du document du traitement qui en est fait. Les éléments logiques sont délimités par un marqueur de début et un marqueur de fin ("start tag", "end tag") et ont un identificateur générique qui leur sont assignés. Dans l'exemple suivant, l'élément <titre> peut être un marqueur pour délimiter le titre du document : <titre> Gestion de QdS dans le Web </titre> . Le programme de traitement du document peut par exemple en lisant ce code décider de mettre le texte contenu dans cet élément en gras avec une certaine police de caractère. L'idée de base ici est que SGML n'offre que le support pour décrire la structure logique du document (titre, paragraphe etc.). Il appartient aux programmes utilisant ce code de décider du traitement à en faire.

HTML étant une application de SGML, est caractérisé par un ensemble d'éléments logiques et une sémantique qui leur sont associée. Cette caractéristique, comme pour toute application de SGML, est décrite sous forme d'une spécification formelle que l'on appelle DTD. Le DTD, écrit en SGML, décrit de façon formelle la structure logique des types d'élément qui composent une catégorie donnée de document (HTML, document multimédia ou rapport technique par exemple).

Certains associent le succès du WWW au langage HTML. Parmi ses principaux avantages, on peut citer sa simplicité, sa capacité à pointer sur n'importe quelle ressource disponible sur l'Internet. De plus, elle est indépendante de toute plate-forme. Cependant, l'inclusion des données multimédia dans l'environnement du WWW crée des besoins d'extension du langage HTML qui sont assez diverses.

II.2.2.2 Inclusion de documents multimédia dans un document HTML

La seule forme d'inclusion de média initialement offerte par HTML (version 2.0) était limitée aux images avec le *tag* <IMG>. En vue de supporter l'inclusion de média de plus en plus variés, plusieurs nouveaux éléments ont été introduits par diverses compagnies : <DYNSRC> introduit par Microsoft pour l'inclusion d'audio/vidéo, <EMBED> introduit par Netscape pour incorporer différentes ressources dont celles lues par les plug-ins, <APPLET> par Sun pour les codes exécutables. Le groupe de Vosaic a également proposé initialement les marqueurs <audio>, <video> <vidaud> avant d'adopter <OBJECT> décrit ci-dessus.

Par souci d'offrir une solution flexible et extensible qui soit profitable à toute la communauté du Web, le groupe W3C a produit au mois d'avril 1996 une spécification définissant un nouveau tag, <OBJECT> [Kin96] qui regroupe les fonctionnalités de ceux introduits précédemment, offre une solution générale pour supporter l'inclusion des nouveaux media (objets) tout en préservant la compatibilité avec les anciens navigateurs. Le terme objet ici désigne tout ce que les gens voudraient inclure dans un document HTML à savoir : les applets, les plug-ins, les programmes de présentation de média etc. À l'heure actuelle, <OBJECT> n'est pas encore implanté par les navigateurs couramment utilisés. Seuls, les navigateurs Grail [Gail] et Vosaic [Zhi95] le supportent. Dans notre projet, nous avons opté pour l'utilisation de ce nouveau tag à cause de son caractère potentiellement standard. Nous décrirons l'élément <OBJECT> plus en détail dans le chapitre 5.

II.2.2.3 La structure et les méta-données des documents multimédia

Les documents multimédia sont des objets ayant une structure complexe et constitués de composantes primitives (monomédia) telles que la video, l'audio, l'image, le texte etc. Les composantes du document ont des relations spatiales, temporelles ou logiques entre elles. De plus, chaque composante peut être disponible sous plusieurs versions chacune ayant des paramètres de QdS qui les caractérisent. Ces informations sur le document multimédia que nous appelons méta-données doivent être modélisées explicitement et stockées pour permettre une meilleure

manipulation du document. Les méta-données décrivant la composition du document multimédia ainsi que les paramètres de QdS qui caractérisent chaque version de monomédia concerné sont indispensables à la gestion de la QdS. Dans notre projet de maîtrise, nous avons codé ces informations directement en HTML. La solution que nous proposons est basée sur la structure qui a été définie dans le projet initial CITR pour caractériser les documents multimédia (Chapitre 3). Le chapitre 5 expose notre proposition concernant la description des méta-données des documents multimédia en HTML.

Vue d'une manière plus générale, la question qui se pose est de trouver une syntaxe formelle qui décrit la relation entre les composantes et leur comportement dynamique (relations spatio-temporelle entre les monomédia). Mais d'après les travaux du sous-projet CITR de l'Université d'Alberta, le DTD de HTML ne permet pas la spécification des contraintes spatio-temporelles entre les composantes. C'est le standard HyTime[3] qui a été utilisé pour représenter cette méta-information sur le document. Dans le projet initial, c'est le module de synchronisation qui consulte la base de données multimédia pour recueillir les informations spatio-temporelles. L'agent de négociation de QdS se charge quant à lui de rechercher les méta-données qui lui sont utiles. Aussi, dans le présent projet, nous nous sommes occupés de coder les méta-données utiles à la gestion de la QdS et celles utiles aux serveurs multimédia pour localiser les données. En d'autres termes, l'extension HTML que nos proposons ne s'applique pas aux aspects dynamiques de la description des documents.

D'autres considérations concernent par exemple le contrôle de la présentation d'un média inclus dans une page HTML. Plusieurs types de présentations sont possibles. Par exemple, le son peut être utilisé en arrière plan ou présenté à la manière traditionnelle où l'utilisateur dispose d'une interface de contrôle avec les fonctions play, rewind etc. Pour l'instant aucun standard n'est défini pour offrir ces fonctions dans une page HTML. Une approche possible consiste à définir des attributs standards pour le tag <OBJECT> ou plus généralement offrir une interface standardisée (API)

---

[3] HyTime, Hypermédia/Time-based Structuring Language est une application et une extension de SGML

permettant ainsi à l'auteur du document d'une page HTML de contrôler la présentation d'un média [Hos96].

II.3 Quelques applications multimédia dans l'environnement du WWW

Dans cette section nous allons décrire brièvement quelques travaux similaires qui ont déjà été réalisés dans l'environnement du WWW. La plupart d'entre eux ne considèrent pas la gestion de la QdS du point de vue l'utilisateur. Du moins, ils ne fournissent pas de mécanisme pour prendre en compte l'opinion de l'utilisateur. Les systèmes existant sont généralement conçus sous formes de client/serveur et intégrés dans l'architecture du Web selon l'approche présentée à la section 2.1.3. Le client est en général disponible sous forme de plug-in ou helper comme Vosaic ou RealAudio. En vue d'étendre notre application, nous l'avons conçue de manière à ce que n'importe quel de ces clients puisse être utilisé pour jouer le média, une fois la négociation réalisée. Le chapitre 5 donne plus de détails sur l'intégration de ces systèmes dans notre application.

II.3.1 Le système Vosaic

Né d'un projet de recherche au département d'Informatique de l'Université d'Illinois, le système Vosaic [Zhi95] est devenu au courant de l'année 1996, une entité commerciale. Implanté au départ comme une extension du navigateur Mosaic (d'où le nom de Vosaic comme Video Mosaic) pour supporter la vidéo, Vosaic est un des premiers et des plus connus à offrir Vidéo et Audio sur le Web en transmission continu. Actuellement, le client Vosaic est déjà disponible sous forme de *plug-in* sur une variété de plate-forme.

Partant du constat que le protocole TCP est inadéquat pour la transmission des données continues, les auteurs du système ont inventé le protocole VDP spécialisé dans le transfert de données continues sur Internet et dont les caractéristiques ont été décrites plus haut. En résumé, le protocole VDP, grâce à son mécanisme d'adaptation masque à l'utilisateur les pertes de paquets de données dues au réseau résultant en une dégradation brusque de la qualité de la présentation ou les délais introduits par la machine cliente dus à sa capacité limitée. VDP dégrade plutôt gracieusement la

qualité de la présentation en réduisant l'utilisation de la largeur de bande, prévenant ainsi la congestion du réseau ou la surcharge de la machine cliente. Pour réduire l'utilisation de la largeur de bande le serveur largue tout simplement certaines trames.

Dans le système Vosaic, c'est la partie client du protocole qui est responsable de la gestion de perte de trames. Les trames étant numérotées, les pertes peuvent être détecter au niveau du client qui décide quelle trame doit être retransmise selon l'impact de ce dernier sur la qualité de la présentation.

En plus des extensions apportées à HTML pour inclure audio et vidéo, Vosaic permet également l'accès aléatoire à des séquences vidéo dans le flot de données sans faire une recherche séquentielle. Il a aussi l'avantage de supporter plusieurs formats standards de codage de données. Cependant, la gestion de la QdS ne fait pas intervenir l'utilisateur.

II.3.2 Le système Fast Web

L'approche développée par le projet Fast Web [Fry96] pour inclure les données multimédia dans l'environnement du WWW tient compte, comme dans notre projet, de la gestion de la QdS en faisant intervenir l'utilisateur. De manière globale, l'architecture de Fast Web comprend une composante client et serveur pour les données continues (Continuous Media, CM). Le serveur est contrôlé à partir du client Web via le serveur HTTP par le mécanisme de CGI[4] .

Le protocole de négociation proposé vise à trouver le meilleur compromis entre les capacités du serveur et du client CM, les conditions du réseau et les préférences de l'utilisateur. Pour un même clip vidéo, on définit plusieurs niveaux de QdS en faisant varier différents paramètres de QdS qui sont : le débit (frame rate), la taille, la couleur, la résolution de l'image. Chaque niveau de QdS a un besoin donné en largeur de bande. L'utilisateur indique également l'importance qu'il accorde à chaque paramètre de QdS. En fonction de ces données de l'utilisateur, le protocole de négociation de QdS détermine une liste ordonnée (chemin de dégradation) des différentes qualités de vidéo qu'on peut jouer à l'utilisateur. Les niveaux de QdS qui

[4] CGI (Common Gateway Interface) est un standard qui permet à des serveurs WWW d'exécuter des programmes externes, http://hoohoo.ncsa.uiuc.edu/cgi/overview.html

ne peuvent être supporter par la machine sont ignorés. Le client CM détermine alors la largeur de bande actuellement disponible, choisit dans la liste de dégradation le niveau de QdS qui correspond à cette largeur de bande et le présente à l'utilisateur. En cas de violation (perte de données élevée), la vidéo suivante dans le chemin de dégradation est automatiquement choisie.

La transmission se fait par le protocole RTP. Le protocole RTCP permet au client d'envoyer des comptes rendus au serveur sur la qualité de la transmission. Le serveur peut alors entreprendre une adaptation. L'interface de négociation de QdS est offerte à l'utilisateur au niveau du client Web par l'entremise des formulaires HTML (HTML Form). Cette interface permet de sélectionner la vidéo à jouer, les valeurs de paramètres de QdS désirées, les niveaux de priorité entre ces paramètres et le type de la machine cliente.

Le projet Fast Web ne définit pas une structure générale sur les documents multimédia et les paramètres de QdS comme dans notre projet. D'autre part leur projet initial n'offre pas une interface de négociation de QdS pour l'audio. Dans notre projet, les niveaux de QdS correspondent à des versions ; ces versions sont des représentations physiques différentes (réparties éventuellement sur différents serveurs) de la même vidéo. Dans le projet Fast Web les niveaux de QdS sont calculés en variant différents paramètres de QdS sur la même vidéo. Par contre, nous utilisons des principes similaires concernant la négociation de QdS.

II.3.3 Le système RealAudio

Le système RealAudio [Pro95] qui est d'une très grande popularité délivre la voix et depuis peu de la vidéo de bonne qualité sur des modems de 14,4 à 28.8 Kbps ou via des lignes ISDN. Le système a été implanté par la compagnie Progressive Networks qui a lancé dernièrement RealVideo. C'est un système client serveur comprenant les modules suivants :

1) **"Le player"** joue le média. Cette composante est implantée sous diverses (helper, plug-in, ActiveX). Un API est également disponible sur certaines plates-formes pour contrôler le player (sous forme de plug-in) à partir d'un applet Java.

2) **L'encodeur** permet d'encoder les données digitales dans un format reconnu par le player.
3) **Le serveur RealAudio/Video** délivre les média continus.
4) **Le client et le serveur Web** servent les pages Web qui contiennent des pointeurs vers le player.

Le fonctionnement du système est celui décrit dans la section II.2.1.3. A cause de contraintes commerciales, la compagnie n'offre pas d'information sur les techniques utilisées pour gérer la QdS. Cette gestion ne fait pas intervenir l'utilisateur de manière explicite.

II.3.4 Le projet de l'université de Kentucky sur les applets temps-réel et la Qualité de service

Ce projet conduit à l'université de Kentucky [Lak96] est encore à une phase de développement. Les objectifs visés dans ce projet sont assez similaires aux nôtres. Parmi les problèmes essentiels abordés dans ce projet, on peut citer la définition de mécanismes permettant d'ajouter le support de la QdS dans l'infrastructure du Web. La plupart des extensions proposées impliquent une réimplantation des outils et langages de programmation du Web tels que HTML, HTTP, le langage Java etc. En somme, un tel système n'utilise pas les implantations de client/serveur Web existants tel que décrit dans la section 2.1.3. Autrement dit, pour l'utiliser, il faut disposer de tout le système client/serveur étendu. Les mécanismes de spécification de QdS ont été étudiés pour différentes composantes :

1) Spécification de QdS dans HTML : Comme dans notre projet, une extension au langage HTML a été proposée pour décrire les différentes versions ou alternatives disponibles pour un media et les paramètres de QdS associés. Le client Web pourrait choisir, pour une demande d'accès à une audio ou vidéo, la variante qui accommoderait l'usager selon la capacité de son lien d'accès à l'Internet. Ceci ne demande pas l'intervention de l'utilisateur. Cette extension utilise des étiquettes HTML (*tag*) définies par eux et n'est donc pas standard. Notre extension s'appuie

par contre sur des étiquettes proposées par le W3C.

2) Extension au protocole HTTP : Une proposition est faite pour étendre le protocole HTTP pour permettre à un client Web de spécifier ses besoins de QdS au serveur Web à l'intérieur d'une requête pour retirer un document.
3) Extension au langage Java et aux librairies de Unix pour la manipulation des sockets: L'extension au langage Java a pour but de permettre aux *applets* Java de spécifier leurs contraintes en termes de délai et de besoin en ressources comme largeur de bande, CPU au niveau de la machine cliente hôte qui l'exécute. De plus, les commandes du client Web (incluant donc des requêtes de réservation de ressources au niveau de la machine cliente) doivent pouvoir se convertir en commandes équivalentes au niveau de la couche transport. Une réimplantation des librairies de Unix pour la manipulation des *sockets* est donc nécessaire pour que le système d'exploitation puisse comprendre ces commandes.
4) Définition d'une interface au niveau du système d'exploitation qui permet aux *applets* Java d'utiliser les *threads* temps réel. Le but est de permettre aux programmes clients de maximiser l'utilisation qu'ils font des ressources de la machine hôte. Ce mécanisme a été également développé dans le projet initial du CITR. En effet le CMFS est basé sur ce mécanisme de sorte qu'un client CMFS doit utilisé la librairie RTT (RealTime Threads) implantée pour supporter le mécanisme.

D'après l'article cité en référence (septembre 1996), toutes les extensions proposées n'ont pas encore été encore implantées.

II.4 Conclusion

Ce chapitre montre que le développement des applications multimédia dans l'environnement du Web est en pleine évolution. La situation a beaucoup changé depuis le début de notre projet. Notre évaluation porte essentiellement sur les applications de type présentationnelles. Plusieurs solutions non standard sont déjà disponibles. Notre application se situe dans cette catégorie mais l'accent est mis sur la gestion de la QdS tout en tenant compte de l'opinion de l'utilisateur. Parmi les systèmes présentés, seul le projet FastWeb offre la gestion de la QdS. Nous nous sommes basés sur la notion de priorité et de chemin de dégradation utilisé dans ce projet. Cependant, leur système n'est pas aussi complet que le nôtre. Le prochain chapitre donne un aperçu du projet initial du CITR dont le nôtre est une suite.

III. Le projet CITR

III.1 Aperçu du projet CITR

Dans le contexte d'un projet de recherche lancé par l'Institut Canadien pour la Recherche en Télécommunication et mené en collaboration avec d'autres universités canadiennes, un prototype de Nouvelles-sur-Demande (prototype initial du CITR) a été développé. Cette application a été choisie comme exemple représentatif des applications multimédia présentationnelles [Haf96] sur lesquelles les recherches menées au sein du projet ont été portées. Les systèmes multimédia distribués devraient pouvoir fournir à l'utilisateur un accès efficace aux informations. Cependant, vu que les objets multimédia sont des objets volumineux et complexes ayant des caractéristiques temporelles, leur transfert, leur manipulation et leur présentation à l'utilisateur demandent beaucoup de ressources de la part du système. Il est donc essentiel de gérer ces ressources afin d'offrir à l'utilisateur une bonne qualité de présentation. De ce fait, la gestion de QdS apparaît comme une fonction essentielle dans les systèmes multimédia distribués. La fonction de gestion de QdS est de contrôler et de garantir le niveau de QdS que le système est en mesure d'offrir à l'utilisateur. Le rôle du gestionnaire de QdS est d'examiner toutes les alternatives pour répondre à une requête de l'utilisateur et parmi les différentes possibilités, d'en choisir une qui satisfait aux exigences de QdS exprimées par l'utilisateur et celles imposées par les différentes composantes du système. Les exigences de l'utilisateur concernent de manière typique la qualité de l'information et le coût associé à la livraison de l'information. Les contraintes du système incluent les caractéristiques de la machine cliente, (type de l'écran par exemple), celles du serveur, du système de transport aussi bien que celles des objets multimédia.

Dans le cadre du projet CITR, l'Université de Montréal a été concernée par les aspects du système qui sont essentiels à la négociation et à l'adaptation de la QdS. Le but de ce sous-projet mené par l'Université de Montréal est de concevoir et de

développer des méthodes pour gérer les ressources nécessaires à la négociation et l'adaptation de la QdS dans un environnement distribué. Ce chapitre donne un perçu du projet CITR en mettant l'accent sur les aspects de la conception du système qui concernent la gestion de la QdS. C'est le prototype décrit dans ce chapitre qui nous a servi de *helper* dans notre première approche. D'autre part, la plupart des principes développés dans ce projet (comme la modélisation des documents multimédia) nous a servi de base pour la conception de notre deuxième approche de solution.

III.2 Aperçu du système global

Le système de Nouvelles-sur-Demande fonctionne sur une architecture distribuée où les documents multimédia sont stockés sur divers sites et sont accessibles à partir de différentes machines via le réseau. La figure 3 représente l'architecture globale du système.

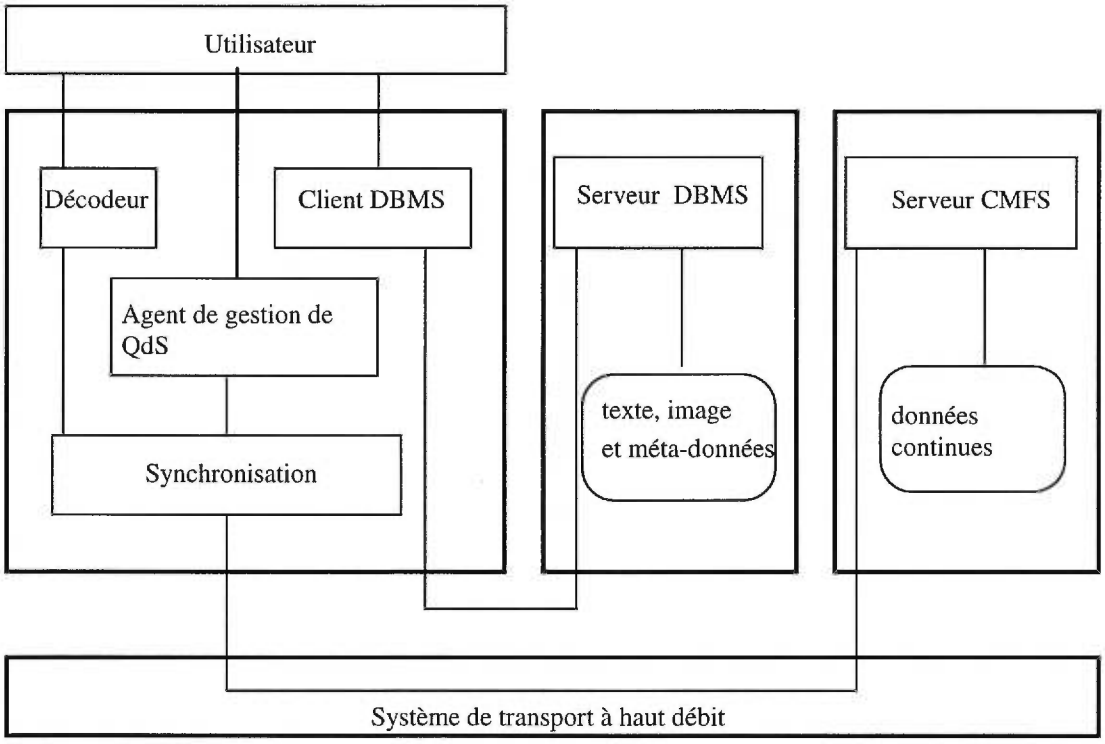

Figure 3 : Architecture globale du prototype initial du CITR

Le prototype de Nouvelles-sur-Demande est une intégration de composantes logicielles développées par divers sous-groupes du CITR à savoir : La base de données multimédia distribuée DBMS, le serveur de données continues CMFS, le module de synchronisation des média, l'agent de gestion de QdS. Dans les prochaines sous-sections, nous allons décrire chacune de ces composantes.

1) La base de données multimédia distribuée (DBMS)

Comme composante du système, une base de données multimédia a été conçue par l'Université d'Alberta pour répondre aux besoins de stockage et de retrait de l'information posés par les applications multimédia. Le but de ce sous-projet est de concevoir une base de données capable de gérer efficacement des données multimédia éparpillés à travers plusieurs sites. Les documents multimédia se caractérisent par leur large volume, la présence de relations spatio-temporelles (par exemple la synchronisation) entre les différentes composants des documents multimédia. Chaque composante peut avoir diverses représentations physiques appelées versions correspondant à différents formats de codage. Ces informations doivent être modélisées et stockées de manière explicite. De ce fait, l'information stockée comprend aussi bien des données utiles (données multimédia) que les informations de gestion (méta-données) utilisées pour faciliter la recherche, le transfert et la livraison (localisation des données, synchronisation des composantes) des documents et aussi pour la gestion de la QdS. La base de données DBMS est décrite plus en détail dans [Vit94].

2) Le serveur de Média Continu (CMFS)

Le but essentiel de cette composante est de fournir des fonctions de livraison des données continues. Comme nous l'avons souligné dans le chapitre I, les protocoles de transport fiables conventionnels ne sont pas adaptés aux besoins des données continues vu les délais que peuvent introduire la retransmission des données

perdues. Cependant, les informations de contrôle moins sensibles au temps doivent être transportées de manière fiable. Le système de serveur CMFS utilise le protocole de transport MT (Media Transport) [Mec96] pour le transfert des media. Ce protocole utilise UDP comme protocole sous-jacent. Le protocole TCP est utilisé comme protocole de transport des informations de contrôle utilisées pour l'établissement d'une connexion. Le contrôle de flux se fait au niveau du serveur qui dispose d'informations précises sur les exigences de la présentation des documents (méta-informations). Le serveur peut donc envoyer les données avec le débit approprié. En plus, la connaissance des capacités de traitement et celles des tampons de la machine cliente permet au serveur d'utiliser la largeur de bande maximale permise par le réseau. A la demande de connexion initiée par un client pour obtenir une donnée continue, le serveur récupère les attributs de la donnée en vu de déterminer le débit (éventuellement variable) du flot de données. Basé sur ces informations, il établit une connexion de transfert en temps réel avec le client en lui indiquant les ressources minimales en terme de largeur de bande et de tampons à consacrer à cette connexion. Le client répond en indiquant les ressources qu'il est en mesure de consacrer à la connexion. Le serveur retourne le contrôle au client avec une indication de succès si les paramètres indiqués par le client sont acceptables pour le serveur. Le transfert sera basé sur les valeurs des paramètres établies lors de la connexion. En somme, le contrôle de flux est réalisé par le serveur et est basé sur les attributs du flot de données à transmettre (méta-données) et les capacités de la machine cliente. Cette négociation se fait au moment de la connexion. Le système de serveur CMFS ainsi que est le mécanisme utilisé pour le contrôle de flux sont décrits dans [Neu96]. Le système CMFS a été intégré dans chacune des deux approches que nous avons développées.

3) La synchronisation des données multimédia

Ce sous - projet mené par l'Université d'Ottawa a pour objectif de concevoir les mécanismes nécessaires pour pouvoir présenter les documents multimédia en respectant les contraintes temporelles et spatiales entre les différents média

constituant le document. Les flots de données synchronisés peuvent provenir de serveurs différents. Les mécanismes développés dans ce projet pour assurer la synchronisation sont décrits dans [Lam94].

4) La négociation et l'adaptation de QdS

Comme souligné plus haut, l'objectif de ce projet est d'étudier l'impact du changement dynamique de la QdS sur la conception des applications multimédia et de développer des méthodes pour la négociation de la QdS avec l'utilisateur et la gestion de la QdS relative aux ressources dans un environnement distribué. Une méthodologie a été développée pour permettre la conception d'applications multimédia qui peuvent de manière dynamique s'adapter aux variations de la QdS des composantes du système et des besoins de l'utilisateur.

La recherche des documents multimédia se fait au niveau de la machine cliente. Pour un document multimédia choisi par l'utilisateur, la négociation se fait pour chaque composante monomédia constituant le document. Ceci conduit à un choix de version (pour chaque monomédia) qui correspond aux exigences de l'utilisateur (exprimées dans son profile) ainsi qu'aux contraintes de tous les acteurs du système impliqués dans le transfert et la présentation du document. Ces contraintes sont identifiées par des paramètres de QdS qui caractérisent les composantes du système. Les informations sur la QdS peuvent être statiques telles que l'environnement de la machine cliente tandis que d'autres peuvent être dynamiques. Les sections 3 et 4 donnent plus de détail sur ces informations.

Le retrait d'un document implique plusieurs composantes du système. Les objets composants le document sont localisés sur un serveur qui peut être soit la base de données (données non - continues ou méta-données) ou le serveur de média continu. L'utilisateur choisit le document à consulter via une interface graphique offerte par l'application. L'agent de négociation de QdS consulte la base de données pour récupérer les méta-données du document, puis négocie avec l'utilisateur la qualité de la présentation. Ensuite, le module de Synchronisation consulte la base de

données pour recueillir les informations sur les contraintes temporelles entre les monomédia afin de planifier la livraison du document. Le serveur de données continues utilise les informations sur l'emplacement des données continues pour effectivement délivrer ces données.

III.3 Architecture du système de gestion de la QdS

Cette section décrit les entités importantes du système développé par le sous-projet de l'Université de Montréal : Le document multimédia dont la structure est gérée par la base de donnée, le gestionnaire de profile, le négociateur de QdS, le contrôleur du réseau.

III.3.1 Structure d'un document multimédia

La figure 4 donne une vue conceptuelle de la structure des objets multimédia utilisée dans la négociation de la QdS. La description est faite dans la notation OMT (Object Modelling Technique) [Ram91]. Un document multimédia est composé de plusieurs objets monomédia habituellement synchronisés les uns avec les autres et éventuellement partagés par différents documents multimédia. Dans le modèle décrit dans la figure 4, une entité multimédia est décrite comme une agrégation de monomédia avec des attributs qui décrivent les relations spatiales et temporelles entre les composantes monomédia. Ces relations sont utilisées durant le transfert et l'affichage du document. D'autres informations comme le prix, des attributs de recherche sont associées au document pour faciliter la recherche dans la base de données. Un objet monomédia est défini dans un médium particulier : texte, image, graphique, séquence audio, séquence vidéo, séquence audio-vidéo. Chaque composante monomédia peut avoir plusieurs représentations physiques correspondant à différents formats ou qualités. Une séquence vidéo peut exister en format MPEG2 ou MJPEG et sous différentes résolutions. Chaque document monomédia peut donc avoir différents niveaux de qualité de présentation définis par ses différentes versions. La qualité d'un objet monomédia est définie par un certain nombre de paramètres qui dépendent du type de médium. Ces paramètres peuvent être le format de codage, la taille du fichier, la couleur de la vidéo. D'autres paramètres comme la localisation

physique du médium sont aussi pris en compte. Les différentes qualités d'un monomédia sont donc associées à des éléments physiques (versions), définies dans le même médium et ayant le même contenu (même valeur informative). C'est cette structure de document que nous avons adoptée dans notre travail. Cependant, notre implantation ne tient pas compte des informations concernant les relations spatiales et temporelles entre les composantes, car celles-ci n'interviennent pas dans la négociation de QdS.

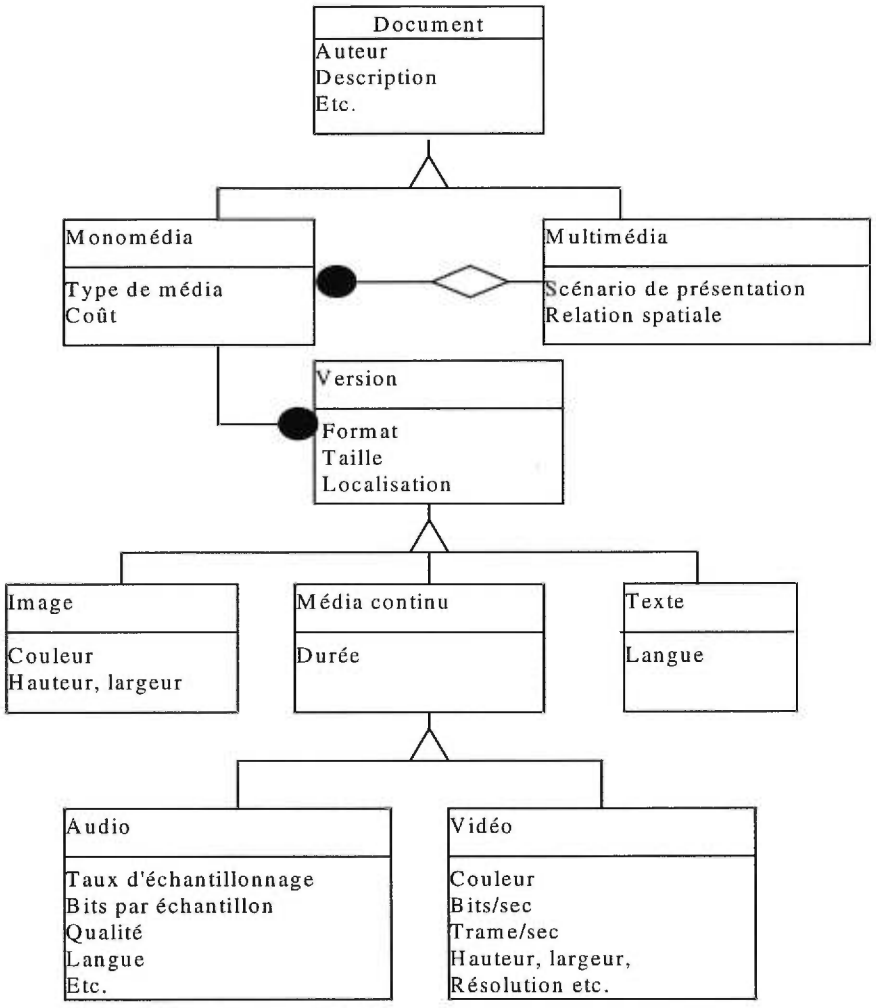

Figure 4 : Description de la structure d'un document multimédia dans le formalisme OMT

III.3.2 Le gestionnaire de profile

Dans la section précédente, nous avons souligné que la gestion de QdS

consiste à offrir à l'utilisateur la qualité de présentation qui correspond à ses préférences. Le gestionnaire de profile a pour tâche d'offrir un moyen à l'utilisateur d'exprimer ses préférences à travers un profile utilisateur. Un profile utilisateur décrit les préférences de l'utilisateur en termes de :

(1) Valeur de paramètres de QdS pour la vidéo, l'audio, le texte et l'image.

(2) Le prix qu'il est prêt à payer pour une qualité donnée

(3) Les contraintes de temps qu'il impose pour la livraison du document (par exemple délai maximal de livraison)

Vu que ces paramètres sont typiquement des paramètres techniques, il faut trouver un langage plus proche de l'usager qui exprime ces paramètres de QdS. Le profile de l'utilisateur est spécifié comme un ensemble de valeurs de différents paramètres de QdS. A chaque type de monomédia (vidéo, audio etc.) est associé un ensemble de paramètres de QdS.

III.3.3 Négociateur de Qualité de Service

La gestion de QdS doit tenir compte des préférences de l'utilisateur aussi bien que des contraintes imposées par les différentes composantes du système comme: les ressources disponibles au niveau de la machine cliente, les formats de codage supportés par la machine cliente, le type de l'écran, la largeur de bande disponible sur la machine cliente. De ce fait, l'activité de gestion de QdS est basée sur les informations de gestion (méta-données) associées aux différentes composantes du système. Les objets du système global pertinents à la négociation de QdS sont: La base de données, le document, la machine cliente, le serveur, le réseau et le gestionnaire de QdS. Le gestionnaire de QdS réalise la négociation en interagissant avec les différentes composantes du système. La négociation consiste à choisir, pour chaque monomédia, une version qui répond aux préférences de l'utilisateur et aux contraintes du système. Elle échoue si pour un monomédia donné, aucune version ne satisfait ces contraintes. La figure 5 schématise les interactions du gestionnaire de QdS avec les autres acteurs. Cette figure introduit deux nouveaux éléments, représentant l'application et les versions de monomédia etc. La section 4 donne plus

de détails sur ces interactions.

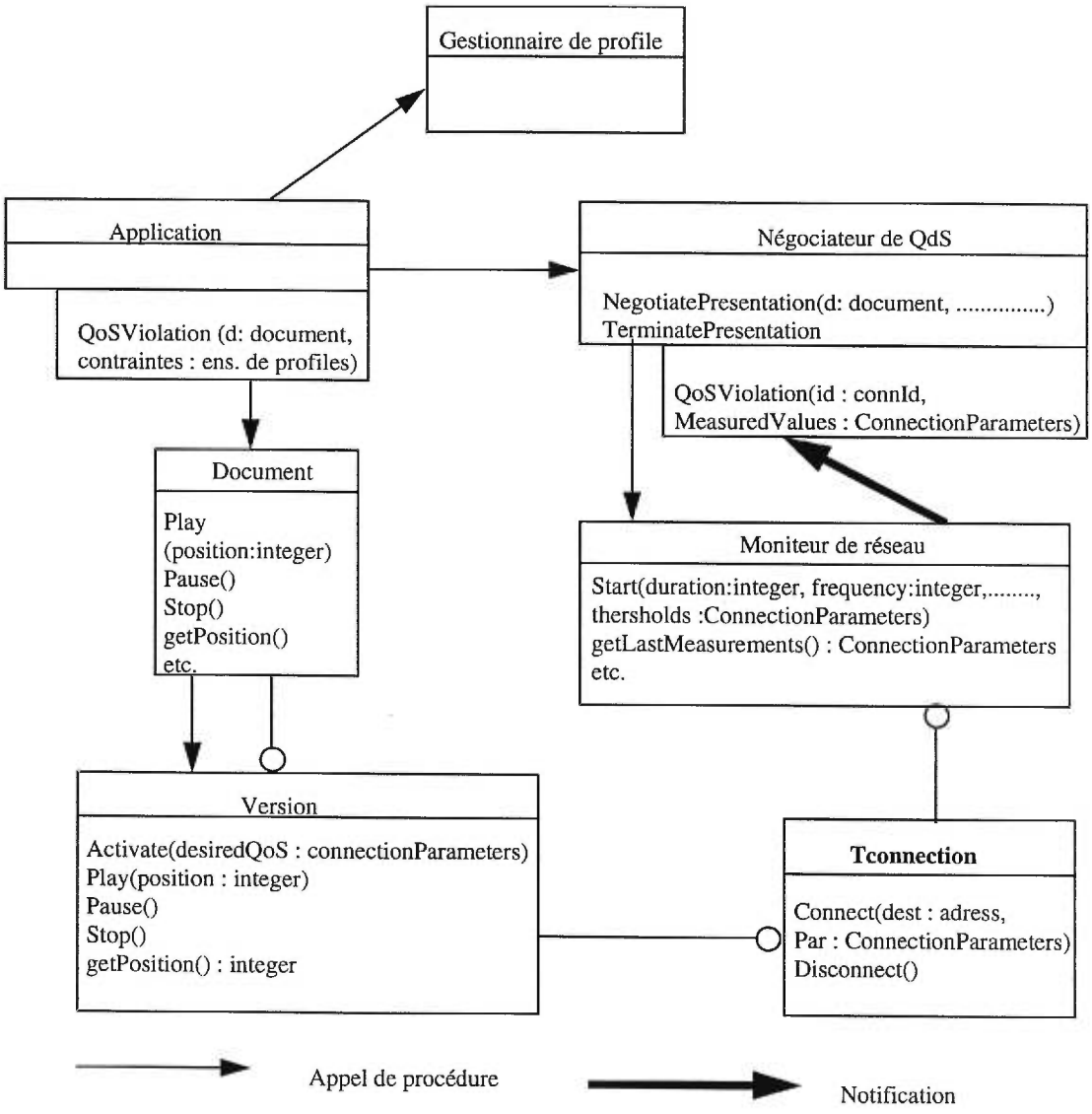

Figure 5 : Interactions nécessaires à la négociation de la QdS dans le prototype initial du CITR

III.3.4 Le moniteur du réseau

Dans le cas où le réseau n'offre aucune garantie sur la QdS convenue lors de l'établissement de la connexion (comme c'est le cas avec l'Internet), il peut être utile

de surveiller le réseau pour déterminer la QdS effectivement fournie. Lorsque celle-ci ne correspond effectivement pas à celle désirée par l'utilisateur, il est possible de passer à une autre configuration du système sans l'intervention explicite de l'usager. C'est le gestionnaire de QdS qui active le moniteur de réseau en lui fournissant les bornes de QdS acceptables pour l'usager. Le moniteur du réseau avertit le gestionnaire de QdS lorsque ses mesures détectent une violation de ces bornes.

III.4 Définition du comportement du système

Après avoir identifié les principales entités intervenant dans le processus de gestion de QdS, nous allons donner un aperçu informel du comportement des acteurs du système de négociation et d'adaptation de la QdS.

III.4.1 Établissement d'une session de présentation

L'application de manière typique commence par déterminer le profile usager à utiliser durant la présentation. Pour cela, il fait appel au gestionnaire de profile. La deuxième étape consiste à recueillir les méta-données en s'adressant à la base de données. L'application fait ensuite appel à l'opération NegociatePresentation du négociateur de QdS en vue de trouver la meilleure configuration du système qui correspond aux préférences de l'utilisateur.

Pour chaque version de monomédia continue choisie, une connexion de transport en temps réel est établie et par laquelle les données seront acheminées du serveur CMFS à la station cliente. Une fois que la connexion est établie pour la version, l'opération ActivateVariant peut être appelée sur cette dernière. Cette opération communique avec le serveur CMFS pour réserver les ressources nécessaires pour le transfert. Ces ressources incluent les mécanismes de synchronisation. Finalement, la présentation du document est amorcée en appelant l'opération Play du document.

III.4.2 La négociation avec l'utilisateur

Comme nous l'avons mentionné ci-dessus, dans le cas où le gestionnaire de QdS ne trouverait aucune configuration adéquate, il retourne un ensemble de

configurations disponibles. L'application a alors un des choix suivants : Abandonner la session de présentation, présenter le document en utilisant une des configurations proposées par le gestionnaire de QdS ou laisser l'utilisateur décider.

III.4.3 Adaptation automatique

Lorsque pendant la session de présentation un problème surgit dans le système (congestion du réseau par exemple), cela peut aboutir à dégradation de la QdS. Supposons que l'application ait activé la surveillance du réseau. Si la moyenne des mesures d'un paramètre de QdS descend en dessous de la limite acceptable, le moniteur de QdS avertit le gestionnaire de QdS. Ce dernier peut ignorer le message et continuer la session ou initier une reconfiguration automatique.

III.5 Définition des paramètres de Qualité de Service

Comme nous l'avons souligné dans le chapitre I, la Qualité de Service est vue comme un effet global de la performance du service offert par un système. Afin d'évaluer cette performance, un certain nombre de paramètres observables et mesurables a été défini dans le contexte du projet CITR [Haf96]. Cette section décrit les paramètres de QdS pour les différentes composantes du système impliquées dans la négociation de QdS. Ces composantes sont regroupées en trois composantes principales qui sont : Le client MM (multimédia), le serveur MM et le système de transport MM. Le client MM offre les fonctions de recherche de document, de visualisation de document et de négociation locale de la QdS. Il inclut les entités suivantes : la pile de protocole de transport, le module de synchronisation, le décodeur, les périphériques de présentation de l'information. Le serveur MM a pour fonction de stocker et gérer les documents multimédia qui sont des objets distribués. Il inclut les composantes suivantes : la pile de protocole de transport, le serveur de base de données et le serveur de média continus. Dans notre projet, nous avons implanté un sous-ensemble des paramètres de QdS décrit dans cette section..

III.5.1 Paramètres de QdS au niveau du serveur MM

L'interface de QdS offerte par le serveur MM est récapitulée dans le tableau 4.

Paramètres	Valeurs
Type de média	texte, image, audio, vidéo, vidéo/audio
Format	ASCII, postscript, gif, JPEG, MPEG, DVI , etc.
Résolution vidéo	pixels/line
Couleur de la vidéo	bits/pixel
taux de trame	trames/sec
fréquence d'échantillonnage	échantillons/sec (Hz)
quantification	bits/échantillon
Débit	bits/sec
Délai	sec
Gigue	sec
Type de garantie	garantie, meilleur effort
Coût	$

Tableau 4 : Paramètres de QdS au niveau du serveur MM

La signification de ces paramètres est comme suit :

Le **type de média** indique le genre de media.

Le **format** indique l'encodage utilisé pour la version de média concerné.

Le **délai** est celui introduit par le serveur.

Le **débit** est le nombre de bits par seconde que le serveur peut fournir à l'application. Il dépend entre autre de la vitesse de lecture sur les disques de stockage.

La **gigue** est la variation de délai (entre différents paquets) introduite par le serveur.

Le **type de garantie** correspond à ce que le serveur s'engage à faire envers l'usager. Lorsqu'il s'agit d'une garantie, le serveur va typiquement utiliser des mécanismes de contrôle d'admission et de réservation de ressources.

Le **coût** correspond à ce que le système exige pour donner accès au document. Il s'agit essentiellement du droit d'auteur.

III.5.2 Paramètres de QdS au niveau du client MM

L'interface de QdS offerte par le client MM est récapitulée dans le tableau 5. Il s'agit essentiellement de la qualité des périphériques et des possibilités des logiciels.

Paramètres	Valeurs
Caractéristiques de l'écran	1bit, gris 8-bits, couleur 8-bits, couleur 24-bits
Périphérique audio	8-bits, 16-bits
Type de média	texte, image, audio, vidéo, vidéo/audio
Format de codage	ASCII, postscript, gif, JPEG, MPEG, DVI, etc.
Débit	bits/sec
Délai	sec
Type de garantie	garantie, meilleur effort

Tableau 5 : Paramètres de QdS au niveau du client MM

La **capacité de l'écran** a un impact sur la qualité de l'image.

Le **périphérique audio** indique le type de carte de son. Selon la carte de son, on peut obtenir du son de qualité "téléphone" , CD ou de qualité intermédiaire.

Le **format** de codage est celui supporté par le décodeur qui se charge de décompresser les données (délivrées par le système de transport) avant leur présentation à l'utilisateur.

Le **délai** est celui introduit par le décodeur. La valeur de ce paramètre peut dépendre de la disponibilité de CPU et du mécanisme d'ordonnancement de tâches utilisé par le système d'exploitation.

III.5.3 Système de transport MM

L'interface de QdS offerte par le système de transport MM est récapitulée dans le tableau 6.

Paramètres	Valeurs
Débit	bits/sec
Délai	sec
Gigue	Sec
Type de garantie	garantie, meilleur effort
Fiabilité	fiable, taux d'erreur
Coût	$

Tableau 6 : Paramètres de QdS au niveau du système de transport MM

Ces paramètres sont comme ceux définis pour le serveur.

La **fiabilité** indique si le service est fiable ou non. Le mode fiable indique que le service se fait sans erreur. Dans le cas d'un service non fiable, une mesure quantitative indique le taux d'erreur (perte de paquets) qui caractérise le service.

Le **coût** représente le coût par minute ou par unité de données chargé par le système de transport.

III.5.4 Les paramètres de QdS au niveau usager

L'usager spécifie ses préférences de QdS via une interface graphique. Cette interface doit rendre transparent le plus que possible les paramètres de bas niveau (débit, gigue etc.) à l'usager. Elle doit permettre de décrire les paramètres de QdS de caractéristiques observables par l'utilisateur lors de la présentation du document. Le tableau 7 récapitule les paramètres de QdS identifiés au niveau utilisateur.

Paramètres	Valeurs
qualité audio	CD, téléphone
langue (audio/texte)	français, anglais
couleur (vidéo/ image)	Blanc & noir, gris, couleur, super couleur

résolution (vidéo/image)	HDTV, TV, minimum
taux de trame	HDTV, TV, gelé
délai	sec
synchronisation	Acceptable, bon, excellent
coût	$

Tableau 7 : Paramètres de QdS au niveau utilisateur

La **qualité de son** "telephone" correspond à celle fournie par le téléphone domestique. Pour cette qualité, le son est échantillonné à une fréquence de 8 kHz avec une quantification à 8 bits. La qualité CD (Compact Disque) fournit un son à une fréquence d'échantillonnage de 44,1 kHz avec une quantification à 16 bits.

Une **résolution** de qualité HDTV correspond à la valeur de 1920 pixels /ligne tandis qu'une résolution minimum correspond à 10 pixels/ligne.

Pour ce qui concerne le **taux de trame**, l'usager peut spécifier n'importe quelle valeur entre 1 (gelé) et 60 (HDTV).

Le **délai** est le temps d'attente que l'utilisateur accepte pour la livraison du document tandis que le **coût** indique le prix que l'utilisateur est prêt à mettre pour consulter le document dans la qualité demandée.

La synchronisation indique le degré de cohérence entre les différents monomédia composant le document lors de la présentation.

Comme on peut le voir, les paramètres utilisés au niveau utilisateur sont exprimés en valeurs quantitatives plus proches de l'utilisateur. Ces valeurs doivent être traduites par le gestionnaire de QdS en des valeurs que le système peut gérer. Le taux de trame peut par exemple se convertir en débit lorsqu'on connaît la taille d'une trame. En ce qui concerne la qualité du son, la résolution de la vidéo, les conversions sont celles ébauchées ci-dessus. Cette section met en évidence les ensembles de paramètres de QdS nécessaires à la négociation de QdS.

III.6 Conclusion

Dans ce chapitre, nous avons présenté le prototype initial du CITR en mettant l'accent sur les composantes du système qui interviennent dans la gestion de QdS, sous projet du CITR mené par l'Université de Montréal. La gestion de QdS est basée sur un cadre général permettant au gestionnaire de QdS de construire de façon dynamique une configuration optimale du système qui répond aux exigences de l'utilisateur en termes de qualité de présentation et de coût. Les paramètres de QdS sont définis sur l'ensemble des composantes du système permettant ainsi de gérer la performance globale du système de bout en bout et ainsi de maximiser la satisfaction de l'utilisateur. C'est le prototype décrit dans ce chapitre que nous avons adapté et utilisé sous forme de helper dans la première version de notre système. Le prochain chapitre décrit les modifications que nous avons apportées au prototype pour obtenir une version adaptée au Web.

IV. Utilisation du prototype initial du CITR comme Helper Application

IV.1 Aperçu

Face à l'utilisation de plus en plus répandue des protocoles du Web pour accéder aux informations disponibles sur l'Internet, l'idée est née d'étendre les principes développées dans le prototype initial du CITR au Web. Ce projet de maîtrise qui se situe dans ce contexte a donc pour but de produire une version adaptée au Web du prototype de Nouvelles-sur-Demande décrit dans le chapitre 3. Pour cela, il est nécessaire de redéfinir l'architecture du système existant afin d'y inclure les interfaces nécessaires à son intégration dans le Web. La redéfinition de l'architecture doit être considérée du point de vue conceptuel et d'implantation. Il a été prévu initialement que la nouvelle architecture sera conçue sans trop de modification à l'architecture initiale du système.

Dans le chapitre 3 nous avons indiqué que les documents étaient répartis entre la base de données DBMS (composantes textes, images et méta-données) et le serveur de données continues CMFS. ObjectStore [Lam91], une base de données orienté-objet a été choisie comme DBMS. ObjectStore offre toutes les fonctionnalités pour implanter un DBMS. Cependant, c'est une base de données peu répandue et elle n'est pas disponible sur toutes les plates-formes. En vue d'aboutir à une solution plus générale, qui ne soit pas liée à une plate-forme donnée, nous avons envisagé de produire une version du système qui soit indépendante de ObjectStore, c'est-à-dire stocker les méta-données ailleurs que dans Objectstore. Deux solutions sont possibles : Les stocker dans un simple fichier ou les coder en HTML.

Pour obtenir la version Web, nous avons dans un premier opté pour une intégration rapide dans le Web du prototype existant. L'approche qui pouvait rendre une telle intégration aussi rapide était de configurer le prototype sous forme de helper application. Dans le même ordre d'idée, l'utilisation d'un simple fichier pour stocker

les méta-données était indiquée. Cette version du système constitue notre première approche de solution. Dans la deuxième approche, nous avons codé les méta-données en HTML et implanté une version révisée de l'agent de gestion de la QdS. Ce chapitre décrit les détails de la mise en œuvre notre première approche.

IV.2 Caractéristiques du prototype adapté au Web

En vue d'intégrer le prototype initial du CITR dans l'architecture du Web, nous l'avons adapté et configuré sous forme de helper. Contrairement au prototype initial qui offre une interface graphique pour consulter les documents disponibles et en choisir un, les documents seront accessibles via le Web. L'utilisateur consulte les documents disponibles en naviguant sur le Web. Ceci remplace l'interface utilisateur du prototype initial du CITR. De manière typique chaque document multimédia est constitué d'une partie texte, éventuellement d'une image, d'une vidéo et/ou d'audio. Les parties textes et images du document lui sont affichées par le navigateur Web de la manière habituelle. L'utilisateur accède aux composantes vidéo et audio en cliquant sur le lien approprié. Ceci déclenche la négociation de la QdS sur le document. A partir de ce moment, la session de consultation se déroule selon la description donnée dans la section III.4.

L'idée générale de cette solution est de coder directement en HTML les composants texte et image des documents multimédia et de mettre les méta-données relatives aux composantes vidéo et audio dans un simple fichier Ascii à part. Les données audio et vidéo restent dans le CMFS comme dans le prototype initial du CITR. Elles sont alors incluses dans les pages Web sous forme de liens hypertextes. Ces liens hypertextes pointent sur les fichiers de méta-données qui sont associés aux documents en question. En effet, à chaque document multimédia est associé un fichier de méta-données disponible sur le serveur Web. Les fichiers de méta-données sont configurés pour le type MIME associé au helper. Lorsque l'utilisateur clique sur un lien hypertexte qui pointe sur un fichier de méta-données, le helper est activé avec le nom de ce fichier comme paramètre. Une adaptation est requise sur le prototype

existant pour obtenir cette version du système. La figure 6 qui suit permet d'illustrer le fonctionnement du système.

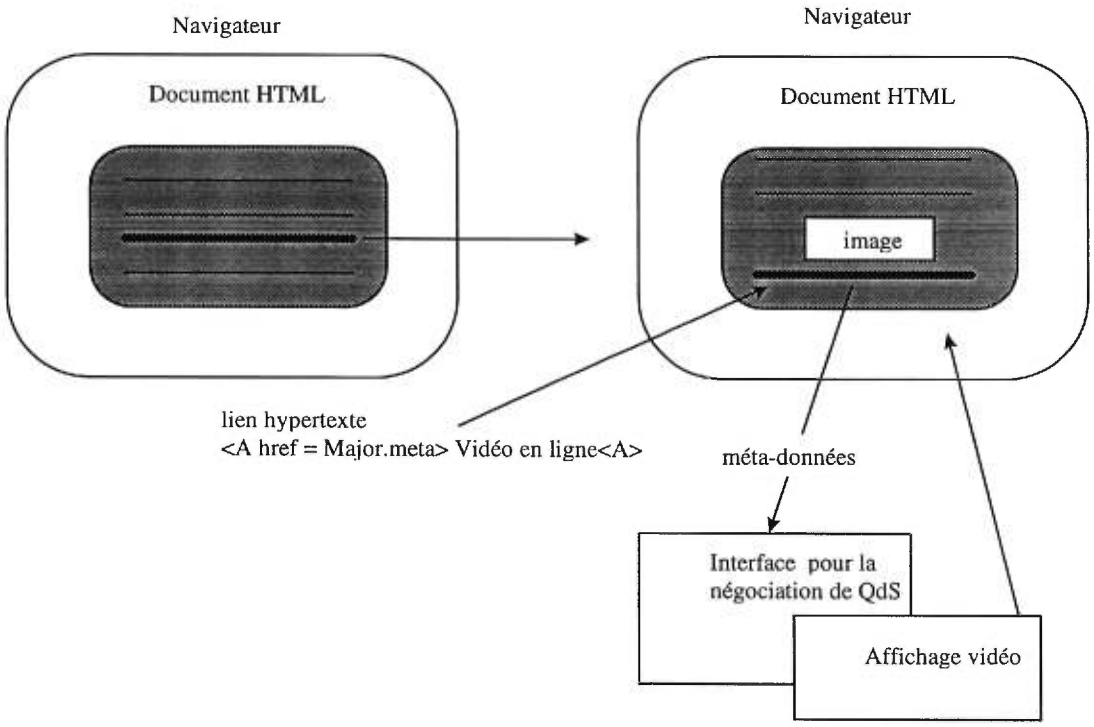

Figure 6 : Fonctionnement du prototype initial intégré dans le Web comme Helper

IV.3 Architecture du système

Comme le montre la figure 7, la nouvelle architecture du système découle de celle existante (figure 3). La partie cliente est essentiellement composé d'un agent de gestion de QdS, d'un module de synchronisation, d'un décodeur et du navigateur Web. La partie serveur est maintenant constituée du serveur Web qui fournit les composantes textes et images des documents multimédia (en utilisant le protocole HTTP) ainsi que des méta-données (en utilisant le protocole FTP). Les média à flux continu sont fournis par le CMFS en utilisant le protocole MT (chapitre 3). Le module de synchronisation constitue le client du serveur CMFS.

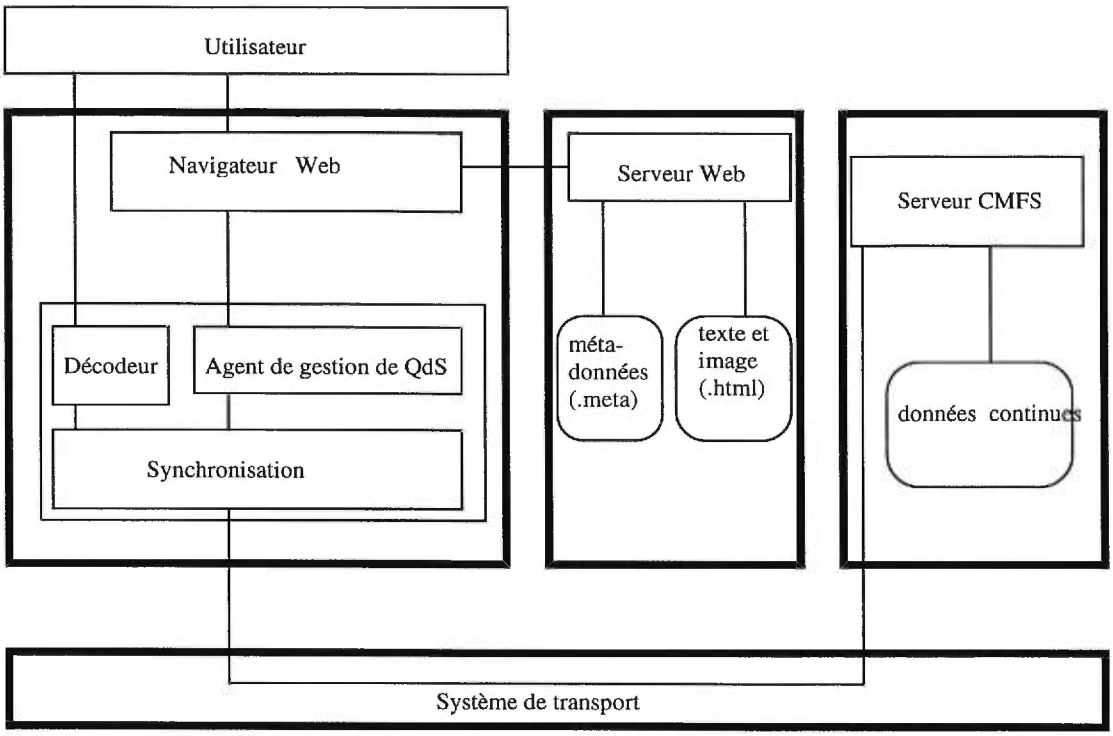

Figure 7 : Nouvelle architecture du système au niveau conceptuel (architecture du prototype initial du CITR adaptée au Web)

IV.4 Implantation

IV.4.1 Structure initiale

La tâche essentielle dans la mise au point de cette version du prototype est de définir et d'implanter les nouvelles interfaces et modules impliqués par l'ajout et la suppression de composantes logicielles. La figure 8 qui suit montre la structure interne de l'agent de négociation de QdS et ses interactions avec les autres composantes du système. Cette structure interne nous permet de bien identifier les interfaces ou modules à ajouter ou à modifier (voir figure 9).

Le processus ou module (1) implante les fonctions de :

- Connexion (login etc.)
- Recherche de documents (par mots clés, par attributs etc.)
- Contrôle de l'affichage telles que les fonctions PLAY, STOP etc.

Les processus 2 et 3 représentent l'agent de gestion de QdS. Il comprend les composantes suivantes : gestionnaire de profile, négociateur de QdS et moniteur du réseau. Ces composantes ont été décrites à la section 3 du chapitre 3. Il comprend aussi un contrôleur qui tient de programme principal. Le processus 2 fait appel aux routines offertes par le client DBMS pour récupérer les méta-données.

L'utilisateur active le processus (1) qui constitue l'interface graphique du prototype. C'est le processus (1) qui à son tour crée le processus (2). Une fois crée, ce processus se met dans une boucle attendant le message PLAY. La réception de ce message déclenche les étapes suivantes :

1 - Le contrôleur fait une requête pour obtenir les méta-données du document pour lequel le message PLAY a été envoyé. Cette requête se fait via l'interface DBMS du processus 2.

2- Le contrôleur obtient le profile utilisateur du gestionnaire de profile.

3- Le contrôleur lance la négociation de la QdS en activant le négociateur de QdS. Les arguments à cette fonction sont le document (concrètement les méta-données) et le profile utilisateur.

4- Le négociateur de QdS utilise les routines offertes par l'interface Réseau&serveur pour vérifier la disponibilité de ressources sur le réseau et au niveau du serveur pour supporter la requête.

5- Le négociateur de QdS retourne le résultat de la négociation au contrôleur.

6- Si l'utilisateur accepte le résultat de la négociation, le contrôleur active l'affichage du document en appelant la fonction correspondante au niveau de l'interface Réseau&serveur pour les média continus et en activant le module d'affichage du processus 2.

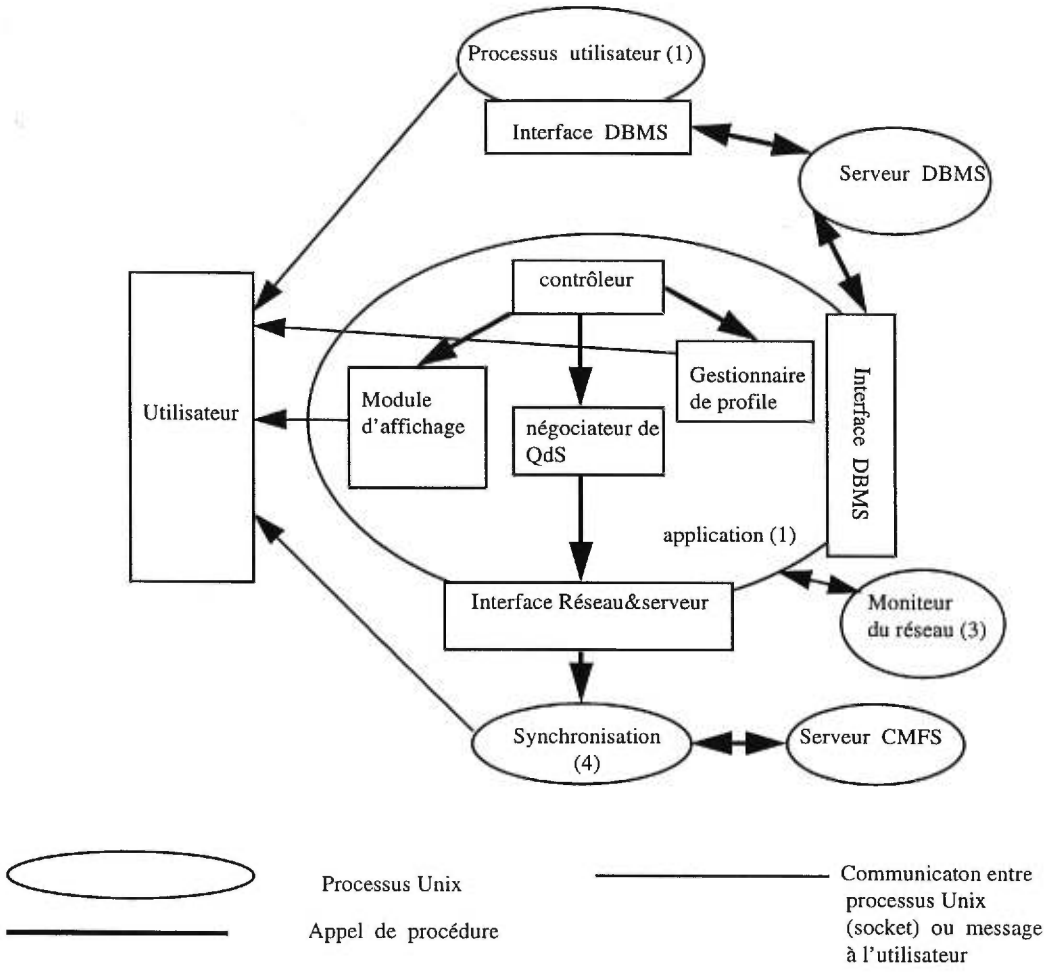

Figure 8 : Architecture du système initial du CITR au niveau d'implantation

IV.4.2 Nouvelle architecture

La figure 9 met en évidence les modifications apportées au logiciel existant. Elles se résument essentiellement en :

(1) Interface avec le navigateur Web en remplacement du processus utilisateur (processus 1).

(2) Implantation des routines initialement offertes par l'interface DBMS du processus 2. C'est la partie grisée de la figure 9. Comme on peut le voir, le système DBMS (implanté par ObjectStore) a été ôté de l'architecture. Textes, images et méta-données (sous forme d'un fichier) proviennent désormais du serveur Web.

Le déroulement d'une session de consultation commence par le navigateur. Comme le montre la figure 8, le processus 1 constituait le programme principal qui active les autres modules. La version modifiée ne comprend pas ce processus. C'est le processus 2 qui est activé par le navigateur lorsque l'utilisateur clique sur un lien pointant sur un fichier de méta-données. Il est activé avec un argument qui est le nom du fichier de méta-données. Le contrôleur (processus 2) fait alors une requête pour obtenir les méta-données via le nouveau module que nous avons implanté. Puis la session continue en suivant les étapes 2 à 6 décrites plus haut.

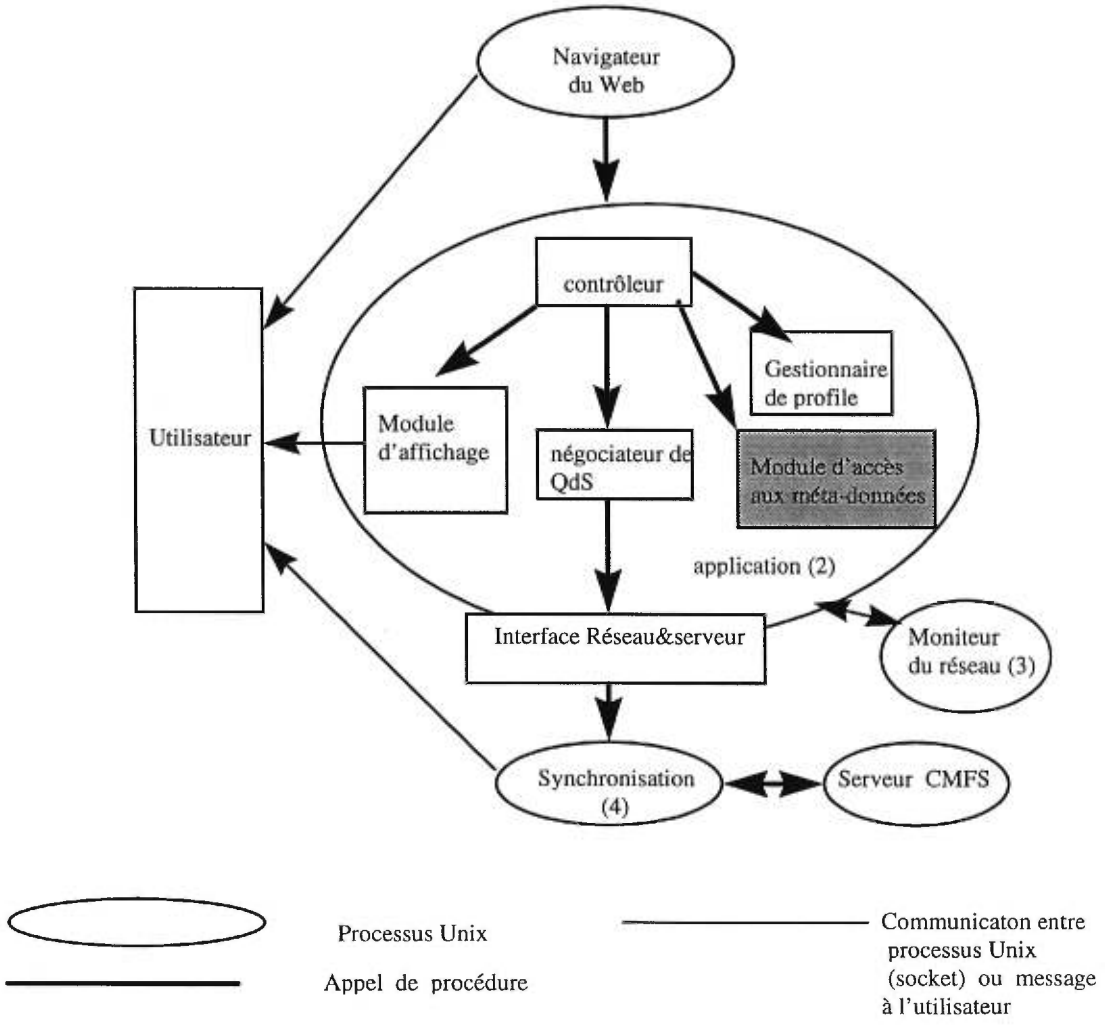

Figure 9 : Nouvelle architecture au niveau d'implantation

IV.4.3 Interface avec le navigateur Netscape

Pour permettre au navigateur Netscape d'exécuter le programme, nous avons défini un type MIME expérimental, **application/x-macro** avec une extension ".meta" qui lui est associée. Le helper a été implanté sous forme d'une routine UNIX, **run_demo** qui réalise les initialisations nécessaires à l'exécution du programme (initialisations des variables d'environnement etc.) et active le processus 2 ensuite. La configuration de Netscape pour reconnaître le type MIME se fait comme suit :

Netscape 2.0

- Ajouter au fichier *.mimes.types* la ligne *application/x-macro* meta
- Ajouter au fichier *.mailcap* la ligne *application/x-macro xterm -e run_demo %s*

Netscape 3.0

Choisir le menu Options, puis Preferences puis helpers et entrer les informations suivantes :

- Type : *application/x-macro*
- Suffix : *.meta*
- Handle by : *xterm -e run_demo %s*

Ceci a pour effet, lorsque l'utilisateur clique sur un lien comme <A href = Major.meta> dans une page Web, de déclencher le prototype adapté du projet CITR (si toutefois le logiciel client est installé sur la machine de l'utilisateur).

IV.5 Remarques sur l'implantation

(1) Pour l'instant le logiciel n'offre pas les fonctions de contrôle STOP, FASTFOWARD etc. Seule la fonction PLAY est implantée. Il faudra ajouter à l'architecture de la figure 9, un module qui implante cette interface graphique.

(2) Le logiciel est implanté sur deux machines IBM RS/6000 (avec le système d'exploitation IBM_AIX) jouant respectivement le rôle de client et de serveur.

IV.6 Conclusion

Cette approche nous a permis d'expérimenter l'utilisation du prototype initial du CITR dans l'architecture du Web et montre la faisabilité de celle-ci. En plus d'offrir le transfert en mode continu, le prototype assure la synchronisation des média, ce qui n'est pas encore disponible à l'heure actuelle dans l'environnement du Web.

Cependant, la complexité du prototype initial du CITR (beaucoup de composantes logicielles développées par divers sous-groupes qui collaborent dans le projet CITR) ne la rend pas pratique comme helper. L'installation du logiciel client n'est pas une tâche à la portée d'un utilisateur ordinaire du Web.

La nature modulaire du logiciel nous a facilité la redéfinition de l'architecture du système au niveau conceptuel. Mais l'implantation qui est essentiellement une modification du module client a été une tâche beaucoup moins triviale. Le logiciel est assez lourd et complexe. L'adaptation au niveau de l'implantation a été très coûteuse en temps.

Dans le chapitre suivant, nous allons décrire notre deuxième approche de solution dont le but est essentiellement d'avoir un logiciel portable.

V. Deuxième approche pour la gestion de la Qualité de Service dans le WWW

V.1 Aperçu

Après avoir expérimenté l'intégration du prototype existant dans l'architecture du Web, nous avons décidé de réaliser une deuxième approche qui consiste à développer une version simplifiée de l'agent de gestion de QdS. Le but visé dans cette approche est d'avoir un logiciel moins complexe et plus pratique pour l'utilisateur. Plusieurs alternatives sont possibles pour implanter un tel agent de gestion de QdS. En effet une application basée sur le Web peut être implantée sous la forme d'un helper, d'un plug-in, d'un CGI [Cgi96] ou d'un applet Java. En vue de pallier à la contrainte de devoir installer et configurer le logiciel nous avons initialement opté pour la programmation d'un applet Java qui sera téléchargé et exécuté sur la machine cliente. De plus le langage Java nous assure la portabilité du logiciel. D'autre part dans cette deuxième approche nous avons codé des méta-données en langage HTML comme cela a été prévu initialement dans le projet. Le langage HTML offre un moyen pour inclure directement des images dans une page Web. Notre objectif était de trouver une extension HTML qui nous permette de réaliser la même chose pour les données audio et vidéo. Une telle extension doit aussi permettre la description des méta-données attachées aux média en question. Les caractéristiques de cette deuxième solution se résument aux points essentiels suivants :

- Conception et implantation en Java d'un nouvel agent de gestion de QdS.

- Définition d'un nouvel algorithme de négociation de QdS. Nous allons décrire notre algorithme dans la section 4.

- Codage des méta-données en langage HTML de sorte que le texte, l'image et les méta-données relatifs à un document multimédia soient centralisés sur le serveur Web.

Dans le chapitre 3, nous avons indiqué que les données continues sont stockées sur le serveur CMFS développé à l'Université de Colombie Britannique. Dans l'environnement du Web, d'autres serveurs de données continues sont également disponibles. C'est le cas du serveur RealAudio et du serveur Vosaic qui offrent le transfert en mode continue comme dans le cas du CMFS. En vue d'élargir notre système, nous avons intégré dans notre architecture ces différents serveurs de sorte que les données continues peuvent provenir de n'importe quel d'entre eux.

Dans la réalisation de cette approche, nous nous sommes particulièrement penchés sur les aspects suivants : étudier les extensions de HTML nécessaires au codage des méta-données et définir une interface aux différents serveurs à intégrer. Quant à la conception de l'agent de gestion de QdS, nous nous sommes essentiellement basés sur les concepts de base utilisés dans le prototype initial du CITR. Cependant, nous avons proposé une nouvelle approche sur la manière dont l'utilisateur spécifie ses préférences de QdS. Le but recherché est de rendre la spécification aisée et plus naturelle.

V.2 La négociation dans le Web [Apache] [Bri96]

Avant de présenter notre approche pour la négociation dans le Web, il est utile de montrer la forme de négociation existante dans l'architecture du Web. En effet, la notion de négociation existe déjà dans le Web. La négociation fait référence au processus qui consiste à choisir la meilleure version parmi celles disponibles pour une adresse URL (Uniform Resource Locator), dans le cas où la ressource associée à cette adresse serait disponible sous plusieurs versions. Cette notion n'est pas généralisée comme dans notre projet où toutes les composantes du système participent à la négociation. Les dimensions (paramètres de QdS) actuellement prévues dans les spécifications du protocole HTTP, selon lesquelles les versions peuvent varier sont :

- Le type du média : Il s'agit du type MIME associé à la donné à cette version. Des exemples de type MIME courant sont : text/html, image/gif, text/plain etc.
- La langue
- Le type d'encodage peut indiquer par exemple si donnée est compressé ou traitée

d'une manière particulière.

- Le jeu de caractères : ISO-8859-1 ou autre.
- La taille du fichier.

Bien que ce mécanisme ait été introduit tôt dans l'architecture du Web, il est resté peu déployé. Les serveurs et clients qui l'implantent sont assez rares. Plusieurs approches ont été proposées pour implanter ce mécanisme : La négociation peut être réalisée par le serveur ou par le client. Mais c'est la solution alliant les deux approches qui est généralement acceptée. L'idée est de permettre au client Web d'indiquer au serveur ses préférences qui sont en fait ses capacités à pouvoir présenter telle ou telle type de données. Comme on peut le voir, dans le contexte du Web, c'est le client Web qui reçoit le service et qui dans ce cas décide pour l'utilisateur.

La spécification de la version 1.0 du protocole HTTP prévoyait déjà des entêtes (dans les messages) par lesquelles le client Web peut spécifier ses préférences de QdS. Dans une requête pour obtenir un document, un client Web peut spécifier dans l'entête, la langue voulue, les formats de donnée supportés etc. Lorsque le client peut supporter plus d'un type de média, il peut indiquer l'importance relative qu'il accorde aux différents types en associant un facteur de qualité à chaque type ou alternative (un algorithme similaire utilisant les facteurs d'importance a été implanté dans le projet initial du CITR que nous allons décrire dans la section 4.1). Ceci permet au serveur de trouver la version de document qui répond le mieux aux souhaits du client.

De son coté, le serveur (par un mécanisme non encore clairement défini), associe des facteurs de qualité aux versions dont il dispose. L'algorithme, situé au niveau du serveur, utilise ces valeurs pour déterminer la meilleure version. La collaboration du client réside dans le fait qu'il informe initialement le serveur de ses préférences.

Cependant, le mécanisme est encore peu déployé dans le Web. Bien que les idées sous-jacentes sont assez compatibles avec les nôtres, notre système ne pouvait pas encore se baser sur ces solutions. En effet c'est le client Web qui initie la

négociation. Il faudrait alors disposer de la part du client Web, une interface pour lui transmettre la spécification de QdS de l'utilisateur. D'autre part, les paramètres de QdS reconnus par le serveur ne couvrent pas ceux utilisés dans notre travail. Il était donc nécessaire de conduire la négociation au niveau même de notre agent de QdS et de coder les paramètres de QdS utiles à la négociation.

V.3 Description des méta-données du projet CITR en HTML

Comme nous l'avions indiqué dans le chapitre 3, les méta-données sont nécessaires à l'activité de gestion de QdS. L'idée d'inclure les méta-données en HTML est née du besoin d'inclure directement les données audio et vidéo dans des pages Web en utilisant des extensions HTML. Comme ces données sont disponibles sous plusieurs versions, il est intéressant de pouvoir exprimer cela dans le même formalisme. Ceci permettrait de créer des sortes d'icônes enrichies (média possédant des versions) dans des pages Web.

Une étude a donc été menée sur les standards et propositions existantes du langage HTML qui concernent la représentation des méta-données pour la gestion de la QdS. Il ressort de notre étude que le besoin existe aussi au niveau des groupes de standardisation des protocoles et langages du Web. Cependant, aucun résultat directement exploitable n'existe encore. Notre étude a abouti cependant à deux nouveaux éléments proposés par le W3C, <OBJECT> et <RESOURCE>. Ces éléments font encore l'objet de discussion au sein du W3C pour leur éventuelle standardisation. C'est par soucis d'aboutir à une extension standard ayant un intérêt général que nous nous sommes gardés de définir des éléments HTML juste nécessaires pour implanter nos méta-données. Nous avons donc étudié les différentes propositions existantes et avons retenu ces deux éléments qui semblent bien convenir à la description des méta-informations sur les documents multimédia en HTML ainsi que leur insertion dans une page Web. Dans cette section nous allons décrire les éléments OBJECT et RESOURCE et montrer leur application à notre projet. Une description plus complète de ces éléments se trouve dans [Kin96] et [Ber95].

V.3.1 L'élément OBJECT

V.3.1.1 Description

<OBJECT> est une extension de HTML pour permettre l'insertion d'objets multimédia dans un document HTML. Ici l'expression "objet multimedia" est utilisée au sens large et inclut les applets java et les plug-ins. Par définition, <OBJECT> permet de spécifier la donnée et/ou les paramètres qui permettent d'initialiser l'objet à insérer dans le document HTML ainsi que le code qui sert à manipuler/afficher la donnée. C'est une généralisation de l'élément IMG qui permet d'inclure des images dans les pages HTML. La définition de OBJECT convient assez bien à notre besoin. En effet, il nous est possible de spécifier le programme qui sert à traiter la vidéo et/ou l'audio ainsi que les paramètres de QdS nécessaires à l'activité de négociation de QdS. Cependant, la spécification actuelle de <OBJECT> couvre seulement la syntaxe et la sémantique pour insérer des objets multimédia dans un document HTML. Les questions relatives à l'architecture et à l'API en ce qui concerne la manière dont les objets communiquent avec le document HTML et avec autres objets du document HTML n'ont pas été abordées. Il est prévu que les spécifications à venir seront plus claires sur ces aspects. D'autre part, elle n'est pas suffisamment flexible pour la description des ressources existant en plusieurs versions. C'est pour cela que nous avons eu besoin d'un autre élément en l'occurrence RESOURCE.

V.3.1.2 Syntaxe

L'élément OBJECT comprend un tag (marqueur) de début <OBJECT> et un tag de fin </OBJECT>. Un ou plusieurs éléments <PARAM> peuvent de manière optionnelle être insérés entre le tag de début et le tag de fin. <OBJECT> a les attributs suivants :

ID: Cet attribut permet de définir un identificateur unique pour l'objet. Le ID doit permettre d'identifier l'objet de manière unique.

DECLARE: Cet attribut indique une déclaration de l'objet (Par opposition à une instantiation de l'objet).

CLASSID: C'est une adresse URL qui identifie une implantation de l'objet, c'est à

dire le programme qui implante l'objet. Des exemples de CLASSID sont donnés dans la spécification mais aucune règle ne ressort de cette spécification pour définir un CLASSID ni d'ailleurs un identificateur (id) pour l'objet.

CODEBASE: C'est une adresse URL qui complète celle indiquée par CLASSID. Elle est utile lorsque CLASSID indique une adresse relative. Il joue le même rôle qu'un chemin d'accès dans un nom de fichier.

DATA: C'est une adresse URL qui permet d'identifier la donnée représentant l'objet. Ceci peut être par exemple un fichier GIF.

TYPE: Cet attribut identifie le type de la donnée désignée par DATA.

<OBJECT> a d'autres attributs dont WIDTH, HEIGHT etc. qui permettent de contrôler l'espace d'affichage de l'objet dans la page Web. L'élément <PARAM> permet de décrire les paramètres nécessaires à l'initialisation de l'objet que l'on désire insérer. Il comprend une paire d'attributs représentant respectivement le nom et la valeur du paramètre selon la syntaxe :

<PARAM name = identificateur-du-paramètre value = valeur-du-paramètre >.

Les éléments <PARAM> utilisés en conjonction avec <OBJECT> permettent de spécifier la liste de paramètres de l'objet.

V.3.1.3 Exemple d'utilisation

L'exemple suivant est tiré du site de Vosaic à l'adresse URL http://www.vosaic.com/videos/CommonDisater/modem.html. Rappelons que Vosaic est une des rares compagnies à avoir implanté <OBJECT>. Le code ci-dessus ne peut être interprété par n'importe quel navigateur et a été mis en commentaire comme on peut le voir.

```
<!--<OBJECT ID="VosW32Ax1"
 CLASSID="CLSID:A8459A43-EB39-11CF-B73E-9CC8F070777B"
CODEBASE=http://vosaic.com/release/plugins/Win32/VosW32Ax.ocx#Version=1,0,
2,1
 WIDTH=160 HEIGHT=160>
   <PARAM NAME="szUrl" VALUE="modem.vos">
   <PARAM NAME="autostart" VALUE=TRUE>
   <PARAM NAME="controls" VALUE=TRUE>
</OBJECT>-->
<embed src="modem.vos" controls=true autostart=true width=160 height=160
statusbar=TRUE>
```

Le code :

```
<embed src= "modem.vos" controls=true autostart=true width=160 height=160
statusbar=TRUE>
```

représente une version alternative du code :

```
<OBJECT ..... .....</OBJECT>
```

pour les navigateurs qui ne supportent pas encore cet élément et nous permet de comprendre cet exemple d'utilisation de OBJECT. En effet, OBJECT est également une généralisation de EMBED. La syntaxe EMBED est destinée à Netscape qui sur la base du nom du fichier à insérer (portant une certaine extension) identifie lui même le programme ou plug-in approprié pour traiter la donnée. Tandis que la syntaxe OBJECT qui est plus générale, offre les moyens d'indiquer explicitement quel programme le navigateur doit activer pour traiter la donnée. Dans cet exemple, on insère le fichier "modem.vos", avec les attributs comme "controls", "autostart", "width" et "height" dans la page Web. On peut se rendre compte que le code OBJECT utilise les même paramètres et la même donnée pour réaliser une fonction équivalente.

V.3.2 L'élément RESOURCE

V.3.2.1 Description

L'élément RESOURCE par définition, offre une manière naturelle de décrire une ressource quelconque, c'est à dire n'importe quelle ressource pouvant être identifiée par une adresse URL. Il permet de donner des informations sur un document à l'intérieur d'un autre document HTML. Cet élément pourrait typiquement être utilisé pour donner des méta-informations (titre, taille de l'information etc.) sur un document destiné à être inclus dans un autre document HTML ou pour décrire différentes versions d'une ressource. Il est similaire à l'élément <HEAD> qui sert à donner des méta-informations sur le document HTML lui-même.

V.3.2.2 Syntaxe

L'élément RESOURCE comprend un tag (marqueur) de début <RESOURCE> et un tag de fin </RESOURCE>. Il peut être utilisé en conjonction avec les éléments <LINK> qui permettent des spécifier les différentes versions d'une même ressource, laquelle est dite ressource générique tandis que les versions sont qualifiées de ressources spécifiques. Lorsqu'ils sont utilisés avec <RESOURCE>, les éléments <META> permettent de donner des informations sur la ressource sans qu'elles soient affichées par le client Web. Les éléments LINK peuvent donc servir à indiquer les différentes versions de la ressource tandis que les éléments META peuvent servir à spécifier les informations sur chaque version. De manière typique, ils servent à indiquer des informations concernant l'auteur du document, les mots clés etc. Ces informations sont utilisées par le serveur Web et le client Web pour divers traitements comme l'identification et l'indexation des documents. Ceci est utile aux moteurs de recherche du Web. L'élément RESOURCE comprend les attributs suivants :

ID: Cet attribut permet d'identifier la ressource de manière unique.

HREF: C'est l'adresse URL de la ressource.

Les attributs *name* et *value* de <META> spécifient respectivement le nom et la valeur de la méta-donnée suivant la syntaxe :

<META name = nom-de-la-metadonnee value = valeur-de-la-metadonnee>.

<LINK> répond à la syntaxe suivante :

<LINK REL = type-de-relation HREF = adresse URL de la version>.

L'attribut "REL" de <LINK> permet d'indiquer que la ressource décrite est une ressource générique. Chaque élément LINK utilisé avec cet attribut indique donc une représentation spécifique de la ressource. L'attribut "HREF" indique l'adresse URL de la ressource spécifique. Le type de relation permet d'indiquer le paramètre selon lequel les versions varient. Plusieurs types de relation sont possibles, comme "content-type-specific" qui indique que les versions ont des types MIME différents, "content-language-specific" qui indique que les versions sont dans des langues différentes etc.

V.3.2.3 Exemple d'utilisation

Dans l'exemple suivant, <resource> permet de décrire une image qui existe en trois versions, chacune dans un format particulier.

```
<resource  href = myimage>
<link rel = "content-type-specific"  href = "myimage.png">
<link rel = "content-type-specific"  href = "myimage.gif">
<link rel = "content-type-specific"  href = "myimage.avi">
</resource>
```

Le code HTML <img src = "myimage"> réfère à une image générique qui existe en trois versions. Le choix de la version à présenter à l'utilisateur relève d'un algorithme de sélection implanté au niveau du client Web. Un autre type de relation, "choice-specific", permet d'offrir une alternative à l'utilisateur, c'est à dire le choix de la version est laissé à la discrétion de l'utilisateur.

Dans l'exemple suivant, différentes versions du document utilisent différents éléments de style :

```
<resource url="mystyle">
<title>Style options for the Kooltown telegraph</title>
 <link rel=choice-specific href="mystyle-old.css"  title="Original Kooltown telegraph style">
```

<link rel=choice-specific href="mystyle-new.css" title="New Kooltown telegraph style">
</resource>
Le navigateur peut par exemple demander le choix de l'utilisateur avant de décider quel type d'élément de style utiliser. En général les différents types de relation sont prévus pour permettre d'indiquer la caractéristique qui fait varier les variantes entre elles tandis que "choice-specific" suggère l'intervention directe de l'utilisateur pour le choix de la variante.

V.3.3 Application de OBJECT et RESOURCE pour la description des méta-données

Pour appliquer ces éléments de HTML à notre problème, il faut identifier l'objet, les paramètres, la donnée et le code qui implante l'objet. Sur la base de la définition de OBJECT, nous avons considéré que l'objet multimédia à insérer est le document multimédia. Cependant, étant donné que la négociation ne concerne pas le document entier, l'objet se résume concrètement à la composante vidéo et/ou à la composante audio du document multimédia ; le code qui implante cet objet est l'agent de gestion de QdS et d'affichage des média. Les paramètres du programme sont les méta-données c'est-à-dire les paramètres de QdS. La donnée qui représente l'objet à insérer est déterminée de manière dynamique. Mais cela ne devrait pas empêcher d'utiliser la syntaxe prévue d'autant plus que cet attribut est optionnel.

Si <OBJECT> permet d'insérer le document multimédia dans la page Web et de spécifier ses paramètres, il ne nous permet de décrire les méta-données. Les méta-données décrivent en effet les différents monomédia qui constituent le document et pour chaque monomédia, la liste des différentes versions qui la composent. <OBJECT> ne permet pas une telle description. <RESOURCE> est l'élément de HTML, qui se prête le mieux à cette description.

Les éléments <META> utilisés habituellement dans HTML sont destinés à décrire des informations pertinentes à l'indexation des documents, lesquelles sont

utilisées par les logiciels clients du Web pour la recherche des documents. Les éléments <META> proposés dans la spécification de <RESOURCE> sont eux aussi bien limités. Ils ne concernent pas les méta-données prévues dans notre application. Pour que <RESOURCE> soit utile à notre application, il faut une extension de l'élément <META> pour permettre la description de méta-données spécifiques à une application. Pour le moment, nous avons utilisé la forme expérimentale de META pour décrire nos méta-données. C'est une convention établie par le W3C, selon laquelle, les auteurs peuvent indiquer la spécificité de leur méta-donnée en faisant précéder le nom du paramètre par "x-".

Pour la description des versions de monomédia, aucun des "types de relation" de <LINK> actuellement définis par les auteurs de RESOURCE n'a été convenable à notre besoin. En effet une relation du type "content-type-specific" par exemple, indique simplement que chaque version de la ressource (image ou vidéo par exemple) est codée dans un format particulier. Or la structure des documents multimédia telle que nous l'avons définie, prévoit que les versions peuvent varier par une combinaison de plusieurs paramètres de QdS et cette précision est utile pour notre application. Aussi avons nous proposé une relation "QoS-specific" pour indiquer que les versions ont différentes caractéristiques de QdS. (Cette proposition a été soumise au W3C pour validation, mais nous n'en avons reçu aucune suite pour l'instant.)

Basé sur ces syntaxes, il nous est possible d'insérer un document multimédia composé de vidéo et/ou d'audio et les méta-données s'y rattachant à l'intérieur d'un document HTML en utilisant les deux éléments <RESOURCE> et <OBJECT>. Prenons un document multimédia composé d'une vidéo existant en une version et d'un audio existant en deux versions. On pourrait exprimer les méta-données de la manière suivante :

1- <resource href = audio>
<link rel = QoS-specific href = audio1>
<link rel = QoS-specific href = audio2>
</resource>

2- <resource href = audio1>

<meta name = url value =" pnm://audio.realaudio.com/audiofile1.ra">

<meta name = format value = g728>

<meta name = bit-per-sample value = 8>

<meta name = sample-rate value = 8000>

<meta name = language value = french>

<meta name = cost value = 3>

</resource>

3- <resource href = audio2>

<meta name = url value = " pnm://audio.realaudio.com/audiofile2.ra">

<meta name = format value = g728>

<meta name = bit-per-sample value = 16>

<meta name = sample-rate value = 44100>

<meta name = language value = english>

<meta name = cost value = 5>

</resource>

4- <resource href = video>

<meta name = url value = "UOI-texte-file.uoi">

<meta name = format value = h261>

<meta name = bitrate value = 30000>

<meta name = framerate value = 5 >

<meta name = cost value = 5>

</resource>

L'insertion de l'objet multimédia pourra se faire par le code suivant :

5- <OBJECT id = "multimedia-object" classid = "negotiation_agent">

<PARAM name = audioComponent value = "audio">

<PARAM name = videoComponent value = "video">

</OBJECT>.

Notons ici que les numéros en gras ne font pas partie du code HTML. Nous les

avons ajoutés dans le simple but de pouvoir faire référence à des portions du code. Le code **5** définit un objet "multimedia-object" qui est implanté par le programme "negotiation_agent". Il a deux paramètres : audioComponent et videoComponent" dont la valeur indique respectivement les identificateurs des composantes audio et vidéo de l'objet. La composante audio est décrite par le code **1**, qui indique des pointeurs vers les codes **2** et **3**. Ces derniers décrivent à leur tour les versions. La composante vidéo n'a qu'une version et est décrite par le code 4.

Une telle spécification, d'après la définition de OBJECT devrait permettre d'inclure (afficher) dans la page Web la donnée multimédia résultant de la négociation. Remarquons toutefois que l'attribut DATA qui indique le nom de la donnée à insérer ne semble pas utile dans le cas de notre application dans la mesure où c'est la négociation qui détermine le fichier audio et/ou vidéo qui va être réellement inséré dans la page Web. En d'autre terme, ce paramètre de l'objet va être déterminé dynamiquement. Il reste à voir comment les implantations de OBJECT supporteront une telle dynamique.

L'attribut "url" indique la localisation de la donnée sur un serveur donné. Son format réel dépend du serveur où réside la donnée. Nous donnerons plus de détails sur les choix d'implantation dans la section 6.

Comme déjà mentionné, ni RESOURCE, ni OBJECT ne sont encore supportés par les navigateurs existants. Par ailleurs, aucune architecture n'est actuellement prévue encore pour permettre aux programmes externes (plug-in, applet etc.) d'accéder aux informations contenues dans RESOURCE, c'est-à-dire qu'actuellement, il n'existe pas à notre connaissance, un API qui permettrait aux clients Web de passer ces informations aux programmes externes. Dans la section 5, nous allons décrire notre implantation de ces éléments.

V.4 Négociation de QdS

La tâche principale de l'agent de négociation de QdS est de déterminer parmi les versions existantes d'un document multimédia, celle qui satisfait le mieux l'utilisateur en tenant compte des caractéristiques des composantes du système dont la

machine cliente. Dans cette section nous allons exposer notre approche pour la négociation de QdS dans l'environnement du Web après avoir donné un aperçu de l'algorithme utilisé dans le projet initial du CITR. L'objectif visé dans la conception de notre algorithme est que la spécification des préférences de QdS doit être le plus simple et le plus naturel possible pour l'utilisateur. C'est dans ce but que nous avons proposé une nouvelle forme de spécification de QdS. Ceci donne lieu également à un nouvel algorithme pour la négociation.

V.4.1 La méthode initiale du CITR

Dans le prototype initial du CITR, il a été proposé un algorithme pour déterminer la meilleure offre. Une offre est typiquement composée d'une version d'audio, de vidéo et de texte. Dans cette approche, l'utilisateur spécifie pour chaque paramètre de QdS, la valeur qu'il désire et la valeur minimale qu'il peut tolérer. Cet algorithme utilise également la notion de facteur d'importance pour indiquer l'importance que l'utilisateur accorde à la qualité de la présentation relativement au coût. L'idée est d'associer à chaque valeur de paramètre de QdS un certain poids (ou facteur d'importance) qui indique "à quel point l'utilisateur désire cette valeur". Ces facteurs d'importance servent à leur tour à déterminer le poids de chaque offre disponible.

Dans l'interface utilisateur qui a été proposée, l'utilisateur indique le facteur d'importance qu'il accorde à un certain nombre de valeurs prédéterminées pour chaque paramètre de QdS. Par exemple pour le taux de trame, l'utilisateur affecte un facteur d'importance aux valeurs clé de ce paramètre comme HDTV, TV, GELE (voir section III.5.4). Pour toute autre valeur intermédiaire, le facteur d'importance est calculé par le système. Le facteur d'importance d'une offre s'obtient en faisant la somme du facteur d'importance de tous les paramètres de QdS et en soustrayant le facteur d'importance associé au coût. La meilleure offre est celle qui a le facteur d'importance le plus élevé.

Bien que cet algorithme fonctionne bien, nous pensons que la détermination du facteur d'importance approprié peut se révéler difficile pour un utilisateur

inexpérimenté. Pour faciliter la spécification de QdS, nous avons proposé une approche alternative où l'utilisateur indique le niveau de priorité qu'il accorde aux paramètres de QdS. La négociation classifie alors les offres par ordre d'importance des paramètres selon les priorités que l'utilisateur aura spécifiées. D'autre part, l'utilisateur indique simplement pour chaque paramètre de QdS, le seuil qu'il accepte.

V.4.2 La nouvelle méthode proposée

Notre approche pour la négociation étant basée sur la prise en compte de l'opinion de l'utilisateur, la spécification des paramètres de QdS par l'utilisateur tient une place importante dans la négociation. La nécessité de définir un nouvel algorithme est né du fait que nous avons utilisé une nouvelle approche pour la spécification des paramètres de QdS par l'utilisateur. Notre approche est basée sur celle utilisée dans le projet FastWeb en ce qui concerne l'utilisation des priorités [Fry96] (section II.3.2). Le processus de négociation de la QdS peut être schématisé par la figure 10. La négociation prend comme paramètres le profile de l'utilisateur, les caractéristiques de la machine, celles du réseau, les versions de document disponibles et détermine la meilleure version pour la présenter à l'utilisateur.

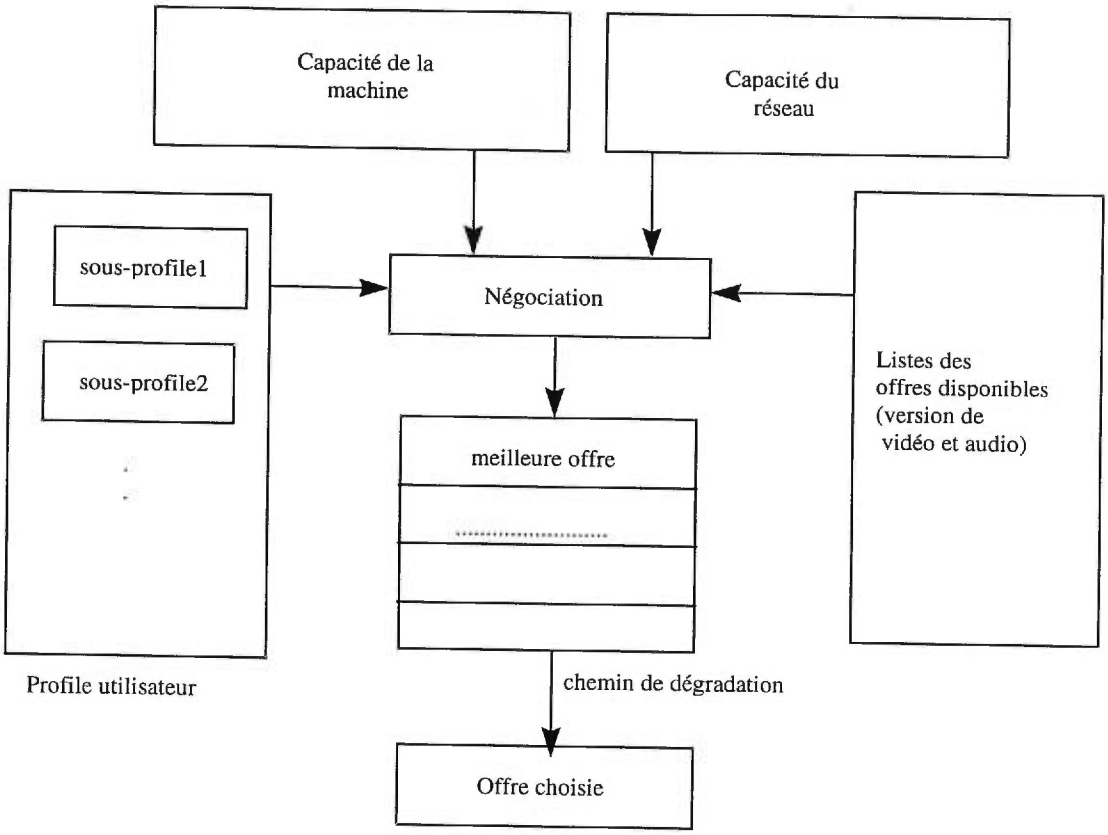

Figure 10 : Principe et éléments de la négociation

Dans la section III.5.4 nous avons identifié un certain nombre de paramètres de QdS au niveau utilisateur. Dans la section qui suit, nous allons citer ceux qui ont été pris en compte dans notre travail. Nous allons également présenter notre approche pour permettre à l'utilisateur d'indiquer ses préférences. Ensuite nous donnerons plus de détails sur le processus de négociation.

V.4.2.1 La spécification des préférences de QdS

Les paramètres de QdS sont définis individuellement par type de média. Pour la vidéo nous avons retenu trois paramètres. Ce sont : le taux de trame (nombre de trames par seconde), la résolution, et la couleur. Pour l'audio, nous avons retenu comme paramètre la qualité du son et la langue. Le coût est spécifié globalement pour tout le document global. L'utilisateur indique un seuil sur la valeur de chaque paramètre de QdS pour spécifier ses préférences. Il ne s'agit donc pas dans ce cas

d'une valeur désirée ; toute valeur supérieure ou égale au seuil est acceptable pour l'utilisateur. L'utilisateur peut également définir une priorité sur les paramètres de QdS. Le tableau 8 qui suit résume les paramètres et valeurs retenus.

Paramètres	**Valeurs**
qualité audio	CD, téléphone
langue (audio/texte)	français, anglais
couleur vidéo	blanc & noir, couleur
résolution vidéo	HDTV(1920 pixels/ligne), TV, minimum (10 pixels/ligne)
taux de trame	HDTV (60 t/s), TV (25 t/s), 15 t/s, 5 t/s
coût	(1-100) $

Tableau 8 : Paramètres de QdS inclus dans notre interface utilisateur

Nous avons estimé que la spécification sera plus flexible si on permettait à l'utilisateur de définir plusieurs niveaux de QdS en leurs associant différentes contraintes de coût. En effet, il se peut que l'utilisateur s'accommode de plus d'un niveau de QdS mais qu'il veuille varier les contraintes de coût en conséquence. Il peut par exemple accepter un taux de trame de 15 pour 3 dollars et un taux de trame de 5 pour pas plus que 1 dollar. En d'autres termes, l'utilisateur définit des profiles multiples représentant chacun une certaine qualité de présentation et un coût associé. Un profile utilisateur est donc constitué de plusieurs tuples (que nous allons appeler sous-profiles) ayant différentes valeurs de paramètres de QdS. Cependant les priorités ne varient pas d'un sous-profile à l'autre. En effet, nous avons estimé que quelque soit les différentes qualités de présentation que l'utilisateur peut définir, son avis sur la priorité entre les paramètres ne devrait pas changer. Il importe de préciser ici que la notion de profiles multiples utilisée dans ce travail est différente de celle évoquée dans le projet initial du CITR [Haf96]. En effet, dans le projet initial, l'utilisateur peut définir plusieurs profiles parmi lesquels il choisit un pour réaliser la négociation. Ces

différents profiles représente en quelque sorte différents avis de l'utilisateur par contre dans notre travail, ces différents avis forment globalement le profile de l'utilisateur. C'est pour faire ressortir cette nuance que nous les appelons des sous-profiles.

Les priorités sont définies globalement sur l'ensemble des paramètres audio, vidéo et le coût. Il aurait été possible de définir un niveau de priorité hiérarchique où l'utilisateur spécifie ses priorités au niveau de chaque type de média (c'est-à-dire entre les paramètres d'un même média), puis un autre niveau priorité entre les différents média et le coût. Mais nous avons jugé qu'il serait plus naturel pour l'utilisateur de mettre tous les paramètres au même niveau que d'avoir à départager entre différents types de média et le coût.

Le système de priorité permet, lors de la négociation de départager les offres disponibles et de les classer. Cette classification se fait même si aucune offre disponible ne correspond au profile de l'utilisateur. L'idée est non seulement de trouver l'offre qui satisfait le mieux au profile de l'utilisateur mais la meilleure parmi celles disponibles en tenant compte des priorités de l'utilisateur. Pour ne pas alourdir l'interface et l'algorithme de négociation, nous avons limité le nombre de niveaux de priorité à trois. Un exemple de spécification pourrait être :

Taux de trame $\geq$ 15	Coût document $\leq$ 20
Résolution = minium	Priorité1 = taux de trame
Couleur = couleur	Priorité2 = qualité audio
Qualité audio = CD	Priorité3 = coût
Langue = français	

En vue de décharger l'utilisateur de cette tâche de spécification à chaque exécution du programme, la spécification est sauvegardée sous forme de profile. Le profile de l'utilisateur donne le plus d'information possible sur l'utilisateur pour permettre au programme de décider à sa place.

V.4.2.2 La négociation

Lorsque l'utilisateur a défini son profile, il peut lancer la négociation en appuyant sur le bouton NEGOTIATE de la figure 13. La négociation se fait pour le

document entier et non pour les monomédia séparément. Comme les parties texte et image ne font pas l'objet d'une négociation, cela se résume aux composantes audio et vidéo. Notre algorithme de négociation comporte quatre étapes qui sont :

1) La négociation locale statique avec la machine cliente qui prend en compte les capacités et les contraintes de celle-ci,
2) La comparaison des offres disponibles contre le profile de l'utilisateur en vue de déterminer celles qui sont éligibles,
3) La classification des offres,
4) La présentation du résultat à l'utilisateur.

1) La négociation locale statique

La négociation locale statique consiste à vérifier si la machine de l'utilisateur est en mesure de supporter le profile demandé. Les paramètres de QdS de service de la machine qui interviennent dans la négociation locale sont :

* Le type du périphérique audio (CD ou téléphone)
* La résolution ou le type de l'écran (minimum, TV, HDTV)
* Couleur de l'écran (couleur ou blanc noir)

Ces paramètres sont détaillés dans le chapitre 3. L'algorithme compare ces caractéristiques de la machine cliente à tous les sous-profiles du profile utilisateur. Si aucun sous-profile ne peut être supporté par la machine cliente, on lui propose un profile qui tient compte des contraintes de la machine. Lorsque la négociation locale réussit ou que l'utilisateur accepte la profile qui lui est proposé, l'algorithme passe à l'étape suivante.

2) La comparaison des offres disponibles contre le profile de l'utilisateur

Cette étape a pour but d'identifier les offres qui satisfont au profile de l'utilisateur. Rappelons que les informations sur ces offres sont fournies par les méta-données. Cette étape consiste à comparer les offres par rapport au profile de l'utilisateur. Les offres sont d'abord comparées à certaines caractéristiques de la machine et du réseau en vue de déterminer celles qui ne peuvent être supportées par le

système. Cette comparaison est différente de celle réalisée à l'étape 1). En effet, dans la première étape, c'est le profile de l'utilisateur qui est comparé aux caractéristiques de la machine et non les offres. Les paramètres de QdS qui interviennent dans cette étape sont :

* Les formats de codage de données supportés par la machine ou plus précisément par le décodeur. Dans le cas de notre projet, nous avons intégré plusieurs serveurs de données continues et utilisons divers plug-ins ou helpers comme Vosaic ou RealAudio pour accéder à ces serveurs. Les formats supportés par la machine seraient donc les formats supportés par chaque plug-in , helper ou autre logiciel client qui y est installé.

* Le type du lien d'accès à Internet (débit) : Le besoin en largeur de bande dépend de la version en question (taux de trame, format de codage de la version etc.). Pour l'audio par exemple, le débit nécessaire dépend de la fréquence d'échantillonnage et du nombre de bits codés par échantillon.

Débit = fréquence d'échantillonnage * nombre de bits par échantillon

Selon la vitesse du lien d'accès à Internet de la machine cliente, certaines versions peuvent ne pas être supportables. Par exemple une version de données audio de qualité CD en mode stéréo demandera une largeur de bande de l'ordre de 1400 kbits/s (44100 *16 *2) si on ne tient pas compte de la compression. Le ratio de la compression dépend du format de codage utilisé. Cependant, pour un même format de codage, une audio de qualité CD demandera une largeur de bande plus élevée que celle nécessaire pour transmettre l'audio en qualité inférieure.

Les offres qui ne peuvent être supportées par la machine sont ignorées. Celles qui passent cette étape sont ensuite comparées au profile de l'utilisateur. Un statut (PASS) ou (FAIL) leurs est assigné selon qu'elles satisfont au profile de l'utilisateur ou non. Une offre satisfait au profile de l'utilisateur si la valeur de chaque paramètre est meilleure ou égale au seuil indiqué par l'utilisateur. L'idée de garder les offres qui

ne satisfont pas au profile de l'utilisateur est de pouvoir offrir une solution à l'utilisateur dans tous les cas, et que cette solution soit la meilleure parmi celles disponibles. De plus, en cas d'adaptation, cela permet d'avoir une alternative à offrir à l'utilisateur. Le rôle de l'adaptation est de maintenir autant que possible la disponibilité du système. Un protocole de QdS est exécuté, chaque fois qu'une violation de QdS est détectée par les mécanismes de surveillance, pour réagir et s'adapter aux changements de l'environnement [Haf96]. Plutôt que de jouer un clip audio de qualité CD avec une perte de donnée trop élevée, il peut être plus intéressant de lui offrir un son de qualité inférieure mais que le réseau peut bien supporter.

3) La classification des offres

La liste des offres ainsi marquées est ensuite triée. Le tri est fait selon 4 critères: En majeur sur le statut, puis sur le paramètre de QdS qui a la première priorité et ainsi de suite. La nécessité de trier les offres selon le statut s'explique par le fait que, si les offres étaient systématiquement triées en majeur sur la première priorité, une offre qui ne satisfait pas au profile utilisateur mais qui a la meilleure valeur pour le paramètre ayant la première priorité peut se retrouver en tête de liste ; à moins d'écarter les offres qui ne satisfont pas au profile, ce critère de tri est nécessaire. Supposons le profile suivant :

taux de trame $\geq$ 10	Priorité 1 = taux de trame
qualité audio = CD	Priorité 2 = qualité audio
coût $\leq$ 6	Priorité 3 = coût

Si nous disposons des offres suivantes, le résultat du tri sera différent selon que le statut est pris en compte ou non :

Offre 1 = [Taux de trame = 9, qualité = CD, coût = 4]

Offre 2 = [Taux de trame = 10, qualité = CD, coût = 6]

Offre 3 = [Taux de trame = 15, qualité = Téléphone, coût = 6].

Seule l'offre 2 satisfait au profile de l'utilisateur et a donc un statut PASS. En tenant compte du statut pour réaliser le tri, on obtient l'ordre suivant :

Offre 2 = [Taux de trame = 10, qualité = CD, coût = 6] ----------> PASS

Offre 3 = [Taux de trame = 15, qualité = Téléphone, coût = 6] ----------> FAIL

Offre 1 = [Taux de trame = 9, qualité = CD, coût = 4] ----------> FAIL

Les offres ayant un statut PASS sont prioritaires sur les offres ayant un statut FAIL. L'offre 2 passe donc en tête de liste. Parmi les offres qui ont un statut FAIL, l'offre 3 est la meilleure parce qu'elle a la meilleure valeur du taux de trame qui est la première priorité pour l'utilisateur.

Dans le cas où on ne tient pas compte du critère statut, on obtient l'ordre suivant :

Offre 3 = [Taux de trame = 15, qualité = Téléphone, coût = 6] ----------> FAIL

Offre 2 = [Taux de trame = 10, qualité = CD, coût = 6] ----------> PASS

Offre 1 = [Taux de trame = 9, qualité = CD, coût = 4] ----------> FAIL

L'offre 3 passe en tête de liste parce qu'elle a le meilleur taux de trame mais elle ne satisfait pas pour autant au profile de l'utilisateur à cause de la qualité du son.

Lorsque le coût n'est pas donné comme étant une priorité, il est ajouté comme un cinquième critère de tri. Dans ce cas, pour deux offres qui ont la même valeur d'un certain paramètre qui ne peut les départager, la moins chère sera la meilleure.

4) La présentation des résultats à l'utilisateur

Après l'étape 3, nous aboutissons à une liste ordonnée des offres disponibles que nous appelons chemin de dégradation. Le premier élément de la liste, qui est la meilleure offre, est alors proposé à l'utilisateur qui peut l'accepter ou le refuser. Dans le cas où l'utilisateur accepterait l'offre, on la lui présente sinon l'utilisateur peut changer son profile et recommencer la négociation. Le chemin de dégradation est utile lorsque le système doit réaliser une adaptation automatique. Lorsque la fonction d'adaptation est supportée, celle-ci est activée lorsque le système ne peut plus supporter la QdS en cours de présentation. On arrête alors l'offre en cours de

présentation, on choisit l'offre suivante dans la liste de dégradation et on la propose à l'utilisateur qui peut l'accepter ou la refuser. La figure 11 qui suit schématise notre algorithme.

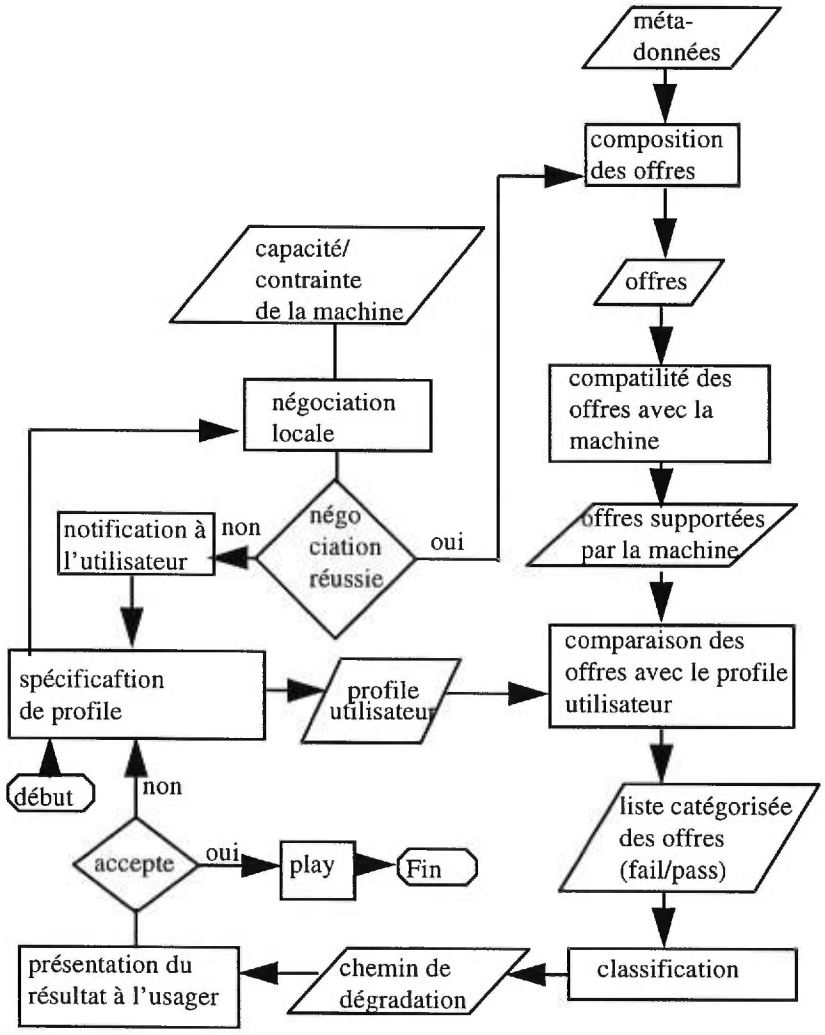

Figure 11 : Algorithme de négociation

V.4.2.3 Quelques exemples de scénario

Dans cette section qui suit, nous allons présenter quelques exemples de profiles et de résultats qu'on peut attendre de notre algorithme de négociation.

On dispose d'un document multimédia composé d'une vidéo et d'une audio. Chacune des deux composantes est disponible sous deux versions qui sont :

Vidéo1 = [taux-de-trame = 5, couleur, résolution = minimum, coût = 3]

Vidéo2 = [taux-de-trame = 15, couleur, résolution = TV, coût = 5]

Audio1 = [qualité = CD langue = anglais, coût = 5]

Audio2 = [qualité = téléphone langue = anglais, coût = 2]

Ces versions résultent en quatre offres de document qui sont :

Offre1 = [taux-de-trame = 5, couleur, résolution = minimum, qualité = CD langue = anglais, coût = 8]

Offre2 = [taux-de-trame =5, couleur, résolution = minimum, qualité = téléphone, langue = anglais, coût = 7]

Offre3 = [taux-de-trame = 15, couleur, résolution = TV, qualité = CD langue = anglais, coût = 10]

Offre4 = [taux-de-trame = 15, couleur, résolution = TV, qualité = téléphone langue = anglais, coût = 5]

Le tableau 9 qui suit donne les résultats pour trois profiles différents (tt = taux de trame, q = qualité audio, R = résolution, c = coût). Les deux premiers profiles ont les mêmes valeurs de paramètres de QdS. Les offres 1 et 3 satisfont à ces valeurs et ont un statut PASS. Ces deux offres passent donc en tête de liste. La meilleure offre dépendra des priorités de l'utilisateur. Dans le premier cas (priorité 1 = coût), l'offre 1 qui a un coût de 8$ sera la meilleure par rapport à l'offre 3. Dans le deuxième cas, l'offre 3 qui a le taux de trame le plus élevé est la meilleure.

Le troisième profile est un profile multiple. Pour une qualité de son CD, l'utilisateur est prêt à payer 10 $ et pas plus que 6$ pour la qualité téléphone. Les offres 1, 3 et 4 satisfont au profile. La priorité étant le coût, l'offre 4 sera la meilleure.

Profiles	**Chemin de dégradation**	**Résultat**
tt = 5, R= min, couleur, q= CD, c = 10$, priorité 1=coût	offre1,offre3, offre4, offre2	offre1
tt = 5, R= min, couleur, q= CD, c = 10$, priorité 1= taux-de-trame	offre3, offre1,offre2,offre4	offre3

tt = 5, R= min, couleur, q= CD, c = 10$, tt = 5, R= min, couleur, q=telephone, c = 6$, priorité 1=coût	offre4, offre1,offre3,offre2	offre4

Tableau 9 : Exemples de résultat de négociation

V.4.2.4 Description de l'interface graphique pour la spécification du profile de l'utilisateur

L'application offre une interface graphique pour permettre à l'utilisateur de spécifier son profile. Elle constitue l'interface de toute l'application. L'interface est composée au plus haut niveau d'une fenêtre (figure 12) qui comporte trois options : La première option permet de sélectionner le profile par défaut. Le profile par défaut peut être celui défini par l'utilisateur auparavant ou un profile par défaut assigné par le système si l'utilisateur n'a pas de profile disponible. La deuxième option, "Set Profile", permet de définir un nouveau profile ou de modifier celui existant. La troisième option met fin à l'exécution du programme.

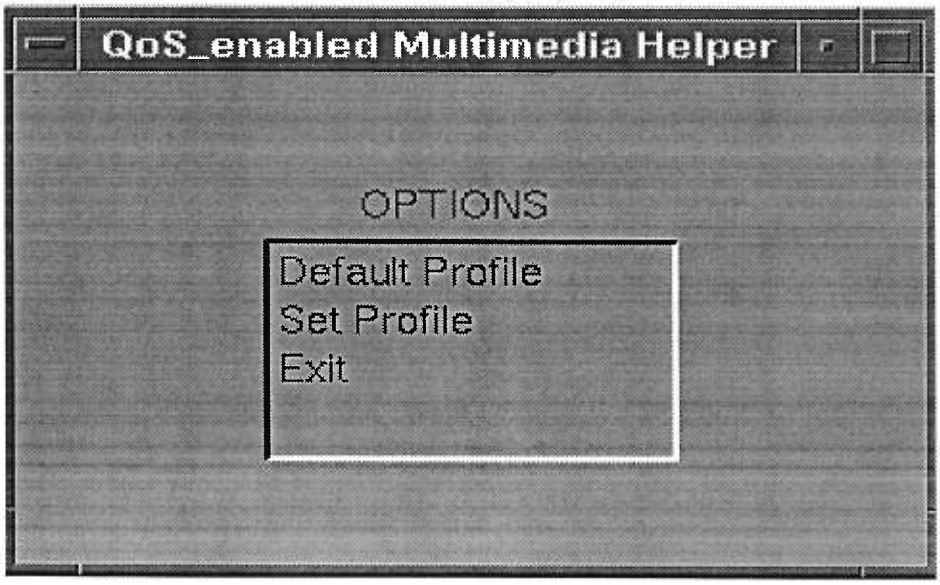

Figure 12 : Menu principal de l'application

L'option "Set Profile" donne accès à une deuxième fenêtre de l'interface (figure 13) qui comprend trois parties essentielles réparties en six blocs. La première partie à gauche intitulée "QoS parameters setting" comprend les blocs pour la spécification des valeurs (seuils) des paramètres. La deuxième partie concerne les priorités et la troisième partie comprend les boutons de commande.

Le premier bloc à gauche contient les paramètres vidéo. Le deuxième bloc à gauche contient les paramètres audio et le troisième bloc à gauche est assigné au coût. Le bloc à droite contient les boutons pour les priorités. Les boutons sont alignés en trois colonnes (correspondant aux trois niveaux de priorité) et six lignes correspondant aux six paramètres. Les boutons sont exclusifs horizontalement (Nous avons décidé qu'un même paramètre ne peut avoir plus d'un niveau de priorité) et verticalement (il n'est pas pratique que plus d'un paramètre soit assigné à la même priorité). Les différents boutons des parties 1 et 2 sont initialisés avec les valeurs du profile utilisateur. Lorsque l'utilisateur n'a pas de profile, le système lui assigne un profile par défaut constitué d'un seul sous-profile que l'utilisateur peut ensuite modifier. L'interface affiche les valeurs d'un sous-profile à la fois.

Les boutons de commande sont répartis en deux rangées : La première rangée correspond aux commandes de haut niveau et comprend les boutons EDIT PROFILE, NEGOTIATE, EXIT. Le bouton EDIT PROFILE correspond aux fonctions d'édition du profile dont les boutons sont sur la dernière rangée. Le bouton NEGOTIATE permet de lancer la négociation et EXIT permet de remonter au premier menu c'est-à-dire à la première fenêtre (figure 12).

Le bouton EDIT PROFILE donne accès aux fonctions suivantes : Parcourir la liste des sous-profiles contenus dans le profile, ajouter un nouveau sous-profile, supprimer un sous-profile, sauvegarder le profile. Ces fonctions sont initialement désactivées. Pour les activer, il suffit d'appuyer sur le bouton EDIT PROFILE. De même lorsqu'on est en mode d'édition du profile, la négociation est désactivée. Le premier sous-profile du profile est initialement affiché. Pour visualiser ou modifier un autre sous-profile, il faut parcourir la liste, choisir le sous-profile désiré et cliquer

deux fois sur le bouton gauche de la souris pour le sélectionner. Les valeurs correspondant à ce sous-profile sont alors affichées. De même pour supprimer un sous-profile, il faut d'abord le sélectionner dans la liste avant d'appuyer sur le bouton CUT PROFILE. Sinon, c'est le sous-profile affiché qui est supprimé. Après une suppression d'un sous-profile, le suivant dans la liste est affiché. La fonction CUT PROFILE est désactivée lorsque le profile ne contient qu'un seul élément. De même, la fonction ADD PROFILE est désactivé lorsque le profile contient déjà trois éléments.

Il n'est pas nécessaire d'entrer dans EDIT PROFILE pour modifier les valeurs affichées à l'écran. Lorsque l'utilisateur lance la négociation sans sauvegarder le profile modifié, la négociation tient compte des changements mais ils ne sont pas sauvegardés pour une utilisation ultérieure. La figure 13 qui suit est une image écran de l'interface.

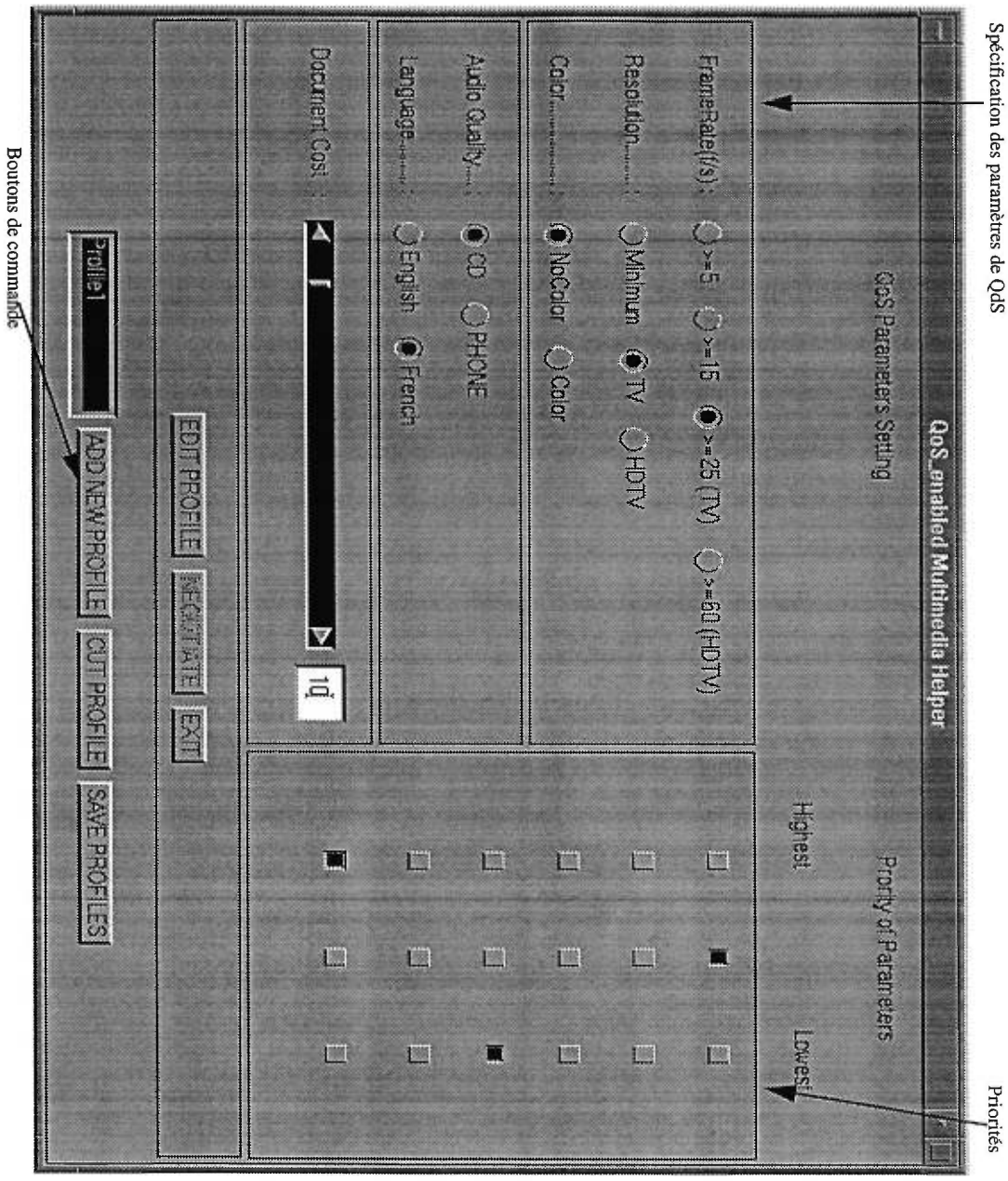

Figure 13 : Interface pour la spécification de profile

V.5 Architecture du nouveau système

Nous avons utilisé une architecture client serveur comme dans la première approche. Les parties client et serveur sont respectivement une intégration de

composantes logicielles client et serveur. La figure 14 qui suit donne une vue de l'architecture du système. Les textes, les images et les méta-données sont stockés sur le serveur Web. Les données continues proviennent, comme nous l'avons mentionné plus haut, du serveur CMFS du projet CITR. Cependant le système a été élargi à d'autres serveurs de données continues qui existent dans l'environnement du Web de sorte que les données vidéo et audio peuvent provenir de n'importe quel d'entre ces serveurs.

Pour accéder à ces différents serveurs, nous avons choisi d'utiliser les clients qui sont déjà disponibles pour chacun d'eux et qui connaissent déjà une certaine popularité chez les navigateurs du Web. Comme la plupart de ces serveurs sont conçus pour fonctionner dans l'environnement du Web, les clients sont en général implantés sous forme de helper ou de plug-in. Il s'agit alors de définir une interface entre ces différents clients et notre agent de QdS. En effet, après la négociation, lorsqu'une version a été sélectionnée pour être présentée à l'utilisateur, l'agent de QdS doit activer le client approprié pour aller chercher la version sur le serveur où elle réside. Nous allons détailler dans la section 6.3, comment nous avons implanté une telle interface.

La partie client du système est donc constituée du navigateur Web, de l'agent de QdS, et du logiciel client correspondant à chacun des serveurs utilisés dans le système. Initialement, nous avons choisi d'inclure les serveurs Vosaic et RealAudio/Vidéo. Cependant, le mécanisme utilisé pour réaliser l'interface avec les serveurs permet théoriquement d'inclure dans le système n'importe quel serveur pour lequel un helper ou un plug-in est disponible.

Dans la section qui suit nous allons présenter nos choix d'implantation. L'implantation porte essentiellement sur l'agent de QdS qui est le but central de ce travail de maîtrise.

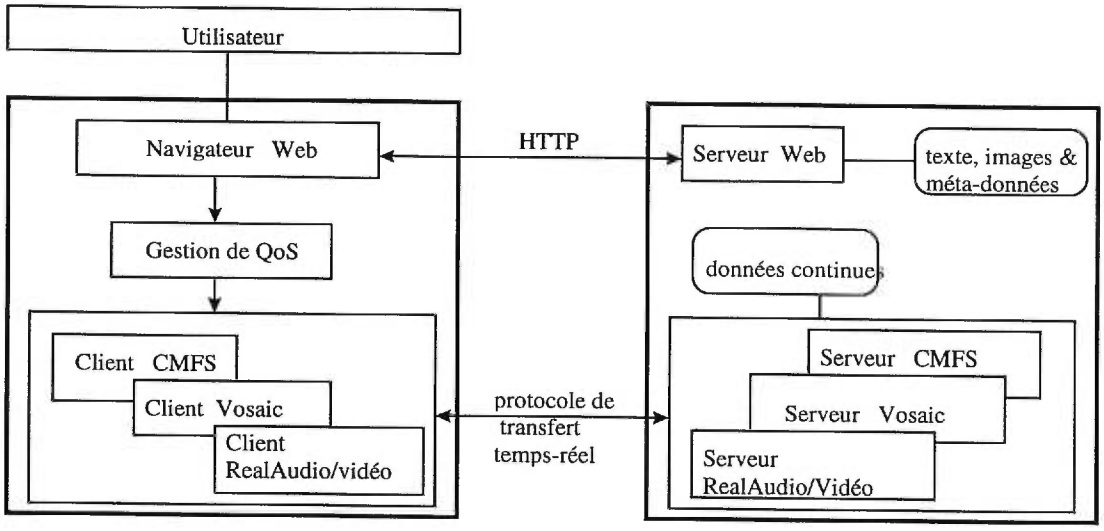

Figure 14 : Architecture conceptuelle du nouveau système

V.6 L'implantation de l'agent de QdS

V.6.1 Commentaires généraux

Basés sur notre expérience du projet CITR, nous avons conçu et implanté en Java un agent de gestion de QdS. Le choix du langage Java est dans le but d'avoir une application portable. En effet, c'est le manque de portabilité qui, entre autre, limitait la première solution qui est par contre assez complète par ses fonctions. L'agent de gestion de QdS fournit les principales fonctions suivantes :

1) Lire et interpréter les méta-données codés en HTML afin de construire la structure du document sur lequel porte la négociation
2) Interface graphique pour la spécification du profile utilisateur
3) Négocier la QdS en tenant compte des offres disponibles, du profile utilisateur et des contraintes de la machine en vue de choisir la meilleure offre parmi celles disponibles.
4) Interface d'accès aux différents serveurs

Le langage Java fournit les classes nécessaires pour implanter de telles fonctions sous formes d'applet. Les avantages que nous associons aux applets Java sont :

- L'utilisateur n'aura pas besoin d'installer le logiciel.
- L'applet s'intègre dans une page Web. L'écran n'est pas encombré avec des fenêtres multiples. L'utilisateur a l'impression de n'utiliser que le navigateur Web.

Malheureusement, les applets ont la sévère restriction (du moins jusqu'à très récemment), de ne pouvoir accéder aux ressources locales du système où ils sont exécutés. Ceci inclut les restrictions de lecture et d'écriture sur la machine cliente. Or l'application a besoin de ces fonctions pour la sauvegarde du profile utilisateur (pour une utilisation ultérieure) et pour la lecture des caractéristiques de la machine.

Contraints par ces restrictions des applets, nous avons choisi d'implanter notre application sous forme de helper comme dans la première approche. Cette restriction n'existe pas pour les helpers du fait qu'ils résident localement sur la machine qui les exécute.

Du point de vu de l'utilisateur, dans le cas de notre application, la différence apparaît de la manière suivante : Le helper est activé en cliquant sur un lien hypertexte (qui pointe sur un fichier configuré pour ce helper). Le helper affiche une fenêtre autre que celle du navigateur pour la négociation. L'applet par contre se matérialise par un objet graphique qui s'insère dans une page Web.

Du point de vue de la programmation, le helper a été implanté sous forme d'un *stand alone* application ou application indépendante qui peut s'exécuter directement à partir d'un shell. Un stand alone application est une classe Java qui fournit une méthode main(). Celle-ci constitue le point d'entrée dans l'application. Par opposition, un applet est un programme conçu pour être intégré dans une autre application. Les applets Java en particulier sont conçus pour être intégrés dans des pages Web. Un applet Java est une classe Java qui fournit les méthodes nécessaires pour permettre au navigateur de l'initialiser, de l'exécuter etc. De plus la programmation d'interface graphique pour les deux types d'application diffère légèrement. Cependant, nous avons conçu notre interface graphique de manière à ce qu'elle puisse s'adapter au helper aussi bien qu'à la version applet. En effet, comme

on le verra dans la section suivante, l'interface graphique a été implantée sous forme d'une classe Java. Cette classe est dotée entre autre, d'un attribut "contenant" qui peut être de type **Helper** ou **Applet**. Une version applet peut donc utiliser la même classe pour implanter l'interface graphique. Notre helper joue le rôle d'agent de gestion de QdS dont nous allons décrire la structure dans la section qui suit.

V.6.2 La structure de l'agent de gestion de QdS

Notre application se répartit en quatre modules principaux qui sont :

1) Un module principal qui sert de contrôleur d'application.
2) Un module d'interprétation des méta-données codées en HTML.
3) Une interface graphique pour la spécification du profile de l'utilisateur.
4) Un module de négociation de QdS.

Dans cette section nous allons donner une brève description de ces différents modules. Chaque module constitue essentiellement une classe Java faisant appel ou ayant pour attribut des classes intermédiaires qui ne sont pas toutes décrites. Nous allons nous limiter dans cette description aux classes importantes. La figure 15 qui suit donne un aperçu des interactions entre les différents modules. Les flèches indiquent des appels de fonctions sur les classes. Les numéros indiquent l'ordre des appels.

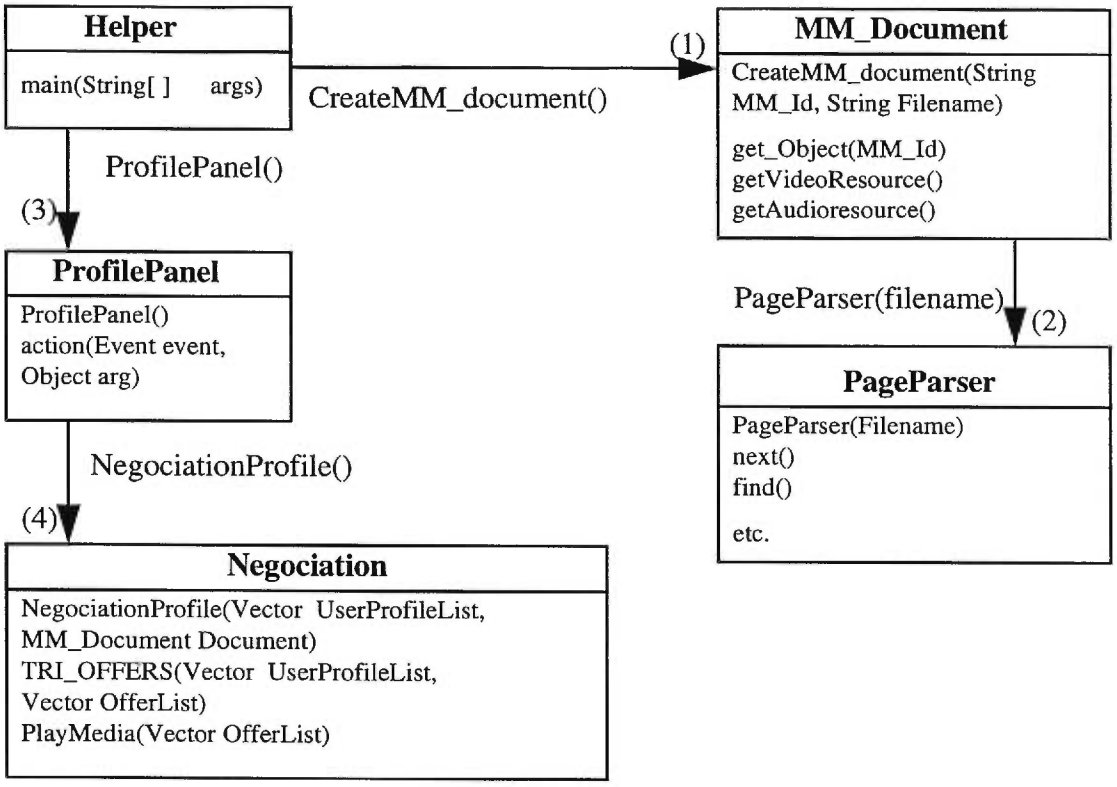

Figure 15 : Interaction entre les différents modules

1) Le module principal

La différence entre une version Helper et une version Applet se rattache essentiellement à ce module qui constitue le point d'entrée dans l'application. Nous avons implanté la version helper sous forme d'une classe que nous avons appelé **Helper**. Il est activé lorsque l'utilisateur clique sur un lien hypertexte qui pointe sur un type de fichier configuré pour ce helper. La méthode importante de la classe Helper est la méthode main(). Elle prend pour argument le nom du fichier faisant l'objet du lien hypertexte pour lequel il est activé. Ce fichier contient deux chaînes de caractères : La première indique le nom du fichier contenant les méta-données et la deuxième indique l'identificateur du document. Lorsque ce module est activé, il crée un objet de type MM_Document (la classe MM_Document est décrite ci-dessus) puis appelle la méthode MM_Create_Document sur cet objet. Cette méthode a pour fonction de récupérer les méta-données. Si cette étape se déroule avec succès,

l'interface graphique est alors activée pour lire le profile de l'utilisateur.

2) Le module d'interprétation des méta-données

Ce module est implanté par la classe **MM_Document**. Il a pour fonction de lire le code HTML contenant les méta-données et de "construire" le document multimédia. Un des attributs importants de cette classe est l'objet parser de la classe **PageParser**. La classe PageParser fournit des méthodes élémentaires pour analyser une page Web.

La méthode principale de la classe MM_Document est la méthode MM_Create_Document(). Cette méthode prend deux arguments qui sont : le nom du fichier HTML contenant les méta-données et l'identificateur du document. Il est utile d'attribuer un identificateur au document parce qu'une page Web peut contenir plus d'un objet multimédia. Dans ce cas l'identificateur sert à se référer à un objet multimédia particulier. La méthode MM_Create_Document() fait appel à la méthode get_Object() qui va rechercher dans le code HTML les paramètres de l'objet dont l'identificateur lui est passé en paramètre. Cette méthode retourne comme résultat des pointeurs vers le code HTML (RESOURCE) qui décrit les composantes de l'objet. Ces pointeurs permettent aux méthodes getAudioResource() et getVideoResource() de récupérer respectivement les méta-données de la composante audio et vidéo du document.

La méthode MM_Create_Document() a pour résultat, deux vecteurs, AudioComponent et VideoComponent qui contiennent respectivement la liste des versions de la composante audio et la liste des versions de la composante vidéo.

3) L'interface graphique

Ce module qui tient lieu de gestionnaire de profile offre les fonctions graphiques de l'interface utilisateur destinée à la spécification du profile de l'utilisateur. Il est implanté par la classe **ProfilePanel.** Il donne comme résultat le

profile de l'utilisateur. L'interface est décrite à la figure 13.

3) Le module de négociation de QdS

Ce module est activé lorsque l'utilisateur appuie sur le bouton NEGOCTIATE de l'interface graphique. Il est implanté par la classe **Negociation** dont la méthode principlale est la méthode NegociationProfile. Il prend pour argument le profile de l'utilisateur et l'objet Document crée par le module principal. Une fois la négociation terminée, il fait appel à la méthode PlayMédia() qui implante les interfaces avec les autres logiciels de la partie cliente. Nous allons décrire dans la section qui suit notre implantation de cette interface.

V.6.3 L'interface de l'agent de QdS avec les autres modules de la partie cliente.

Comme le montre l'architecture de la figure 14, l'agent de QdS n'a pas d'interface directe avec les serveurs mais passe par les clients pour s'adresser aux serveurs. Il a par contre a une interface avec le client Web et avec les autres clients. L'interface avec le client Web est comme celle implantée dans la première solution. Il s'agit de définir un type MIME spécifique à cette application et un fichier correspondant à ce MIME qui lui passe les paramètres nécessaires à son exécution. Les paramètres sont respectivement le nom du fichier contenant les méta-données et l'identificateur du document sur lequel porte la négociation.

Une fois que la négociation est terminée, il faut présenter à l'utilisateur la version d'audio et/ou de vidéo déterminée(s) par la négociation. Étant donné que ces versions peuvent provenir de n'importe quel serveur, il faut s'adresser au serveur approprié. Autrement dit, il faut activer le client approprié. Notons que ce client, doit être installé sur la machine cliente. Dans de telles conditions, le navigateur peut jouer la version en question.

Pour activer le logiciel client, nous avons utilisé le mécanisme de Remote Control de Netscape [Zaw94]. Ce mécanisme permet de contrôler Netscape à partir d'un shell Unix et par conséquent à partir d'un programme. Différentes fonctions,

dont *open(Url)* sont fournies par ce mécanisme. Elles permettent d'activer certaines commandes de Netscape à distance. Par exemple la commande open(Url) à partir du shell a le même effet que lorsqu'on demande une connexion à une adresse Url à partir de l'interface graphique de Netscape.

Pour présenter une version de média, nous proposons la commande open(Url) avec l'adresse Url de la version comme paramètre. Netscape, sur la base de cette adresse Url, déduit le helper ou plug-in approprié qu'il faut activer. C'est ce dernier qui en fin de compte présente le média. Notons également que Java dispose de la classe Runtime qui permet à une application Java d'exécuter un autre programme (créer un autre processus). C'est cette classe que nous avons utilisé pour exécuter la commande open(Url) à partir de notre programme Java.

Dans le cas particulier du CMFS, le client est un programme indépendant qui s'exécute directement à partir du shell. Pour l'activer, nous avons utilisé la classe Runtime. Il aurait été possible de configurer le client CMFS sous forme de helper pour avoir une solution uniforme mais nous n'avons pas jugé utile un tel détour. De plus, dans le système CMFS, la localisation des versions est consignée dans un fichier spécifique et est utilisée au niveau du serveur pour accéder aux données. C'est le nom de ces fichiers qui tient lieu d'adresse Url dans le cas des versions provenant du serveur CMFS. Ce sont également ces adresses Url qui constituent les arguments du client CMFS.

V.6.4 Conclusion

Dans cette solution, nous avons pu utiliser une extension du langage HTML qui nous permet de décrire les méta-données. Nous avons également implanté un agent de gestion de QdS portable. Le mécanisme utilisé pour réaliser l'interface avec les serveurs de données continues permet théoriquement d'utiliser n'importe quel helper ou plug-in existant pour jouer le média une fois la négociation réalisée.

Le logiciel a été testé avec RealAudio sous Unix. Cependant, une démonstration complète requière qu'un serveur RealAudio/Vidéo soit installé. Ceci permettra d'encoder différentes versions des média.

Une version utilisant un client du CMFS sur des stations IBM RS/6000 a été également mise en œuvre. Cette version permet de synchroniser la présentation de l'audio et de la vidéo car elle utilise le module de synchronisation développé par l'Université d'Ottawa. Pour des raisons de compatibilité entre les versions du système client/serveur CMFS, celle-ci n'a pu fonctionner à l'Université de Montréal, mais plutôt à l'Université de Waterloo (responsable de l'intégration des différents sous-projets du CITR). Pour l'Université de Montréal, une version incorporant un client du CMFS sous forme de plug-in est en cours d'installation. Celle-ci sera disponible sur PC et utilisera les serveurs CMFS de l'Université de Colombie Britannique et de l'Université de Montréal.

Il est à noter que l'implantation actuelle n'assure pas encore l'intégration dans une page Web comme nous l'aurions voulu du fait que le navigateur de Netscape ne supporte pas encore les extensions de HTML que nous avons utilisées. L'affichage des media se fait encore dans une fenêtre autre que celle du navigateur.

Remarquons d'autre part que l'implantation actuelle pourrait bien fonctionner si les méta-données étaient stockées dans un simple fichier. Cela ne demanderait qu'une adaptation de la méthode MM_Create_Document() de la classe MM_Document.

Le mécanisme de contrôle à distance du navigateur n'est offert que par le navigateur Netscape. Cela revient à dire que pour l'implantation actuelle, l'intégration des divers helpers et plug-ins ne fonctionne que si on utilise Netscape comme navigateur.

VI. Conclusion générale

Notre application se classe dans la vague des applications multimédia sur le Web et doit être située dans le contexte particulier du projet CITR où un prototype de Nouvelles-sur-Demande a été développé. Le prototype permet aux usagers d'accéder, via un réseau à haut débit, à des documents multimédia provenant de divers sites. Face à l'utilisation de plus en plus répandue des protocoles du Web pour accéder aux informations sur l'Internet, il a semblé intéressant d'étendre les principes utilisés dans le prototype dans l'environnement du Web.

En effet, ces dernières années ont vu un développement rapide des applications multimédia sur le Web. Celles-ci sont habituellement disponibles sous forme de helper ou de plug-in. L'évolution de ces applications est telle que, la situation a beaucoup changé depuis le debut de notre projet. C'est dans cette dynamique que nous avons mené notre recherche, notre but étant de porter le prototype initial du CITR dans l'environnement du Web. Dans ce chapitre, nous allons résumer les réalisations qui ont pu être faites dans notre travail ainsi que les perspectives qui en découlent.

VI.1 Réalisations

Le but initial de ce projet de maîtrise est de porter le mécanisme de négociation de QdS réalisé pour le projet CITR dans l'environnement du Web. La définition du projet prévoit deux versions du système, la première version utilisant le protocole de négociation existant et la deuxième version devant offrir une implantation révisée et portable. C'est ce à quoi notre travail a abouti. Les points essentiels à retenir de notre réalisation sont :

- **L'Adaptation du prototype initial du CITR à l'environnement du Web**.

Le helper résultant permet de jouer des média audio et vidéo synchronisés accessibles à partir du Web. Cette version de notre système montre que le prototype initial du CITR est réutilisable. Cependant sa complexité et sa lourdeur ne la rendent pas assez pratique comme helper. Du fait également de sa complexité, l'adaptation du logicel au niveau de l'implantation a été coûteuse en temps.

- **Application d'extension HTML pour coder les méta-données**

Nous avons montré l'usage d'une extension HTML qui permet d'encoder pour un document multimédia donné, les versions existantes, leurs adresses et d'autres paramètres de QdS. Un auteur d'une page Web pourra aisément spécifier les versions disponibles pour un document multimédia donné. Notre proposition d'utilisation des éléments OBJECT et RESOURCE a été soumise au W3C pour validation. Pour l'instant, nous n'en avons pas encore reçu de suite. Un article portant sur ces travaux a été soumis et accepté à Notère'97[5].

- **Implantation d'un protocole de négociation**

L'application utilise les méta-données codés en HTML pour réaliser la négociation. Vu que les extensions HTML proposées ne sont pas encore supportées par les navigateurs existants, notre application se charge elle-même d'interpréter le code HTML contenant les méta-données.

Une interface graphique a été implantée pour gérer les profiles utilisateurs. La spécification de profile a été simplifiée du point de vue de l'utilisateur. La notion de facteur d'importance a été par exemple remplacée par le système de priorité. Le protocole de négociation est simple et portable sur n'importe quelle machine. Le système a été testé sur des stations Sun, IBM RS/6000 et sur PC.

L'application ne constitue pas en elle même un client Web pour jouer la vidéo

---

[5] Notere'97 (Nouvelles TEchnologies de la Repartition), Pau , France, 4-6 novembre 1997, http://www.univ-pau.fr/NOTERE

mais elle offre des fonctions de négociation de QdS puis utilise des clients multimédia existants pour jouer les média. Dans le cas particulier où un client CMFS est utilisé, la vidéo et l'audio sont présentés de manière synchronisée. Les fonctionnalités du système global sont les suivantes : L'utilisateur sélectionne le document multimédia à consulter à partir du Web. Une négociation est ensuite conduite pour déterminer la version du document qui répond le mieux aux préférences de l'utilisateur. Selon la provenance de la version de document choisie (c'est-à-dire le serveur où est stocké la version), le client approprié est sélectionné pour présenter le document à l'utilisateur. Le protocole de négociation a été intégré à la dernière version du système CMFS et a été démontré à la dernière conférence annuelle du CITR en août 97 à Toronto. Cependant, notre application n'implante pas encore toutes les fonctionnalités du projet initial du CITR. D'autre part, l'intégration dans le Web telle que nous l'aurions voulue n'a pu être réalisée, du fait que les navigateurs n'implantent pas encore les éléments HTML utilisés.

VI.2 Perspectives

- **Intégration des fonctions d'adaptation**

Il serait intéressant d'augmenter les fonctionnalités du système en ajoutant les mécanismes de surveillance du réseau comme cela est le cas dans le projet initial. Ceci permettra de réaliser une adaptation automatique au cours de la présentation lorsqu'un problème surgit affectant ainsi la QdS initialement convenue avec l'utilisateur. C'est la fonction d'adaptation qui permet de rendre transparent à l'utilisateur, l'instabilité du réseau. Cette fonction est d'autant plus importante dans le Web que le réseau sous-jacent est caractérisé par un service meilleur effort sans aucune garantie de QdS.

- **Extension du protocole de négociation**

Une extension intéressante pour la négociation de QdS est également envisageable. Les versions définissant les différentes qualités de présentation pour un document étaient considérées pour chaque media constituant le document. Une autre

approche pourrait être de considérer deux niveaux de négociation. Le premier niveau de négociation porterait sur des versions du document global. Des exemples de versions de document peuvent être : texte seulement, texte et image, texte et son, texte, son et vidéo etc. Pour chaque composante de la version de document choisi, le deuxième niveau de négociation serait celui réalisé dans le système actuel.

- **Normalisation dans le Web**

Dans le contexte général du Web, une meilleure intégration des données multimédia est souhaitable. La normalisation et le déploiement de l'élément OBJECT joueraient un grand rôle dans l'état actuel des choses pour réaliser cette intégration. En effet, d'après la définition de OBJECT, l'utilisation de cet élément devrait permettre à un programme externe d'utiliser la zone d'affichage du navigateur pour afficher les media.

En ce qui concerne la négociation, une normalisation des paramètres de QdS est également souhaitable. La spécification du profile de l'utilisateur pourrait se faire dans ces conditions au niveau du navigateur une fois pour toute. L'implantation de OBJECT devrait prévoir dans le même sens, des mécanismes pour permettre aux programmes externes de récupérer le profile de l'utilisateur et en utiliser la partie qui leurs sera utile. Il en ressort que les paramètres de QdS devraient être considérés dans un cadre général et de manière exhaustive pour que ces paramètres soient utiles à la communauté du Web. D'autre part, cela pourrait rendre beaucoup plus transparent à l'utilisateur le processus de négociation.

Références Bibliographiques

[Apache] *Content Negotiation* http://www.apache.org/docs/content-negotiation.html.

[Apache97] R. Fielding , *About the Apache HTTP Server Project*, http://www.apache.org/ABOUT_APACHE.html, June 1997.

[Ber95] T. Berners-Lee , D. Raggett, *Giving Information About Other Resources in HTML*, http://www.w3.org/pub/WWW/MarkUp/Resource/Specification.

[Ber97] T. Berners-Lee , *About The World Wide Web Consortium* , http://www.w3.org/Consortium/, 1997.

[Bol94] J. Bolot , T. Turletti, *A rate control for packet video in the Internet*, Proc. IEEE INFOCOM '94, pp. 1216-1223, Toronto, June 1994.

[Bol96] J. Bolot and P. Hoschka, *Sound and Video on the Web* , In : 5th International World Wide Web Conference Tutorial Notes, pp.153-175, O'Reilly, Mai 1996.

[Bor93] RFC1521, *MIME (Multipurpose Internet Mail Extensions) Part One:*, http://ds.internic.net/rfc/rfc1521.ps.

[Bri96] B. Behlendorf, Organic Online, *What Is Content Negotiation?* , http://www.organic.com/staff/brian/cn/.

[Cgi96] *The Common Gateway Interface*, http://hoohoo.ncsa.uiuc.edu/cgi/overview.html.

[Con97] D. Connolly, *Overview of SGML Resources*, http://www.w3.org/MarkUp/SGML/.

[Dec94] J. December, N. Randell, *The World Wide Web unleashed*, SAMS Publishing, 1994.

[Fry96] M. Fry, A. Seneviratne, A. Vogel and V. Witana , *Delivering QoS Controlled Continuous Media on the World Wide Web*, In Proc of IWQoS96, pp. 115-124, Paris, March 1996.

[Grail] Grail R 0.3 Home Page, http://monty.cnri.reston.va.us/grail/.

[Haf96] A. Hafid, *Gestion de la Qualité de Service dans les Applications Multimédia*, thèse de doctorat, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Août 1996.

[Hen97] H. Schulzrinne, *About RTP and the Audio-Video Transport Working Group*, http://www.cs.columbia.edu/~hgs/rtp/.

[Hos96] P. Hoschka, *Synchronized Multimedia*, http://www.w3.org/AudioVideo/Activity.html.

[IETF] *The Internet Engineering Task Force*", http://www.ietf.cnri.reston.va.us/.

[Kin96] C. Kindel, L. Montulli, E. Sink, J. Hirschman, T. Berners-Lee, D. Connolly, *Inserting objects into HTML*, http://www.w3.org/pub/WWW/TR/WD-object.html.

[Lak96] K. Lakshman, M. Manoharan, R. Yavatkar, *Adding RealTime Applets and Quality of Service to Support the World Wide Web*, To appear in the proceedings of the Seventh ACM SIGOPS European Workshop on Systems Support for the WorldWide Applications, Connemara, Ireland , September 1996.

[Lam91] C. Lamb, G. Landis, J. Orenstein and D. Weinreb, *The ObjectStore Database System*, Communications of the ACM, Vol 24 #10, pp. 50-63, October 1991.

[Lam94] L. Lamont and N.D. Georganas, *Synchronization Architecture and Protocols for a Multimedia news Service Application*, In Proceedings of the IEEE International Conference on Multimedia Computing System, pp. 3-8, Boston, May 1994.

[Mec96] R. Mecler, *CMFS Data Stream Protocol,* Department of Computer science, University of British Colombia, Canada, August 1997.

[Net96] Netscape Communications Corporation, *Plug-in Developer's Guide*, http://home.netscape.com/eng/mozilla/3.0/handbook/plugins/.

[Neu96] G. Neufeld, D. Makaroff and N. Hutchison, *Server-Based Flow Control in a Continous Media File System*, 6th International Workshop on Network and Operating Systems Support for Digital Audio and Video, pp. 29-35, Zushi, Japan, April 1996.

[Pro95] Progressive Networks, *RealAudio Server*, http://www.realaudio.com/products/server.html

[Rag97] D. Raggett, *HTML 3.2 Reference Specification*, W3C Recommendation 14-Jan-1997, http://www.w3.org/TR/REC-html32.html.

[Ram91] J. Rambaugh & al. *Object Oriented modeling and Design*, Prentice Hall, 1991.

[Vit94] C. Vittal, M. Oszu, D. Szafron, G. Medani, *The logical Design of a multimedia Database for a News-on-Demand Application*, Technical Report #94-16, University of Alberta, Canada, 1994.

[Zap96] D. Zappala, *RSVP Protocol Overview*, http://www.isi.edu/div7/rsvp/overview.html , September 10 1996.

[Zaw94] J. Zawinski, *Remote Control of Unix Netscape*, http://home.netscape.com/newsref/std/x-remote.html.

[Zhi95] Z. Chen, See-Mong Tan, Roy H. Campbell, Yongcheng Li, *Real Time Video and Audio in the World Wide Web*, http://choices.cs.uiuc.edu/Vosaic/Vosaic.html.