

Université Lumière Lyon 2
Thèse pour obtenir le grade de Docteur en Informatique
présentée et soutenue publiquement par
Vasile-Marian SCUTURICI
le 5 décembre 2001

Utilisation efficace des serveurs Web en tant que serveurs vidéo pour des applications de vidéo à la demande

COMPOSITION DU JURY M. Andrezj DUDA : ENSIMAG rapporteur M. Serge FDIDA : LIP6 rapporteur M. Jean-Marie PINON : INSA co-directeur M. Serge MIGUET : ERIC co-directeur M. Grigor MOLDOVAN UBB : (Roumanie) examinateur M. Jean-Baptiste DUCATEZ : CSTI examinateur

Table des matières

Remerciements . .	1
Résumé .	3
Abstract . .	5
Chapitre 1. Introduction . .	7
1.1 Contexte de l'étude . .	7
1.2 Objectif de la thèse .	8
1.3 Principales contributions de cette thèse .	9
1.3.1 Stratégie d'utilisation efficace de HTTP pour la vidéo à la demande .	9
1.3.2 Méthodologie pour la mesure des performances d'un serveur VoD . .	9
1.3.3 Estimation des performances d'un système matériel VoD .	10
1.3.4 Conception d'un serveur Web optimisé pour le multimédia .	10
1.4 Organisation du document .	11
Chapitre 2. Serveurs Web .	13
2.1 Introduction .	13
2.1.1 World Wide Web . .	14
2.1.2 Le protocole HTTP .	16
2.1.3 HTTP 1.0 versus HTTP 1.1 .	18
2.2 Principes de base pour le développement des serveurs Web .	21
2.2.1 Stratégies de concurrence .	21
2.2.2 Architecture d'un serveur Web .	22
2.2.3 Serveurs Web modulaires (scalables) .	24
2.3 Evaluation des performances .	25
2.3.1 Métriques pour mesurer la performance d'un serveur Web .	26
2.3.2 Comment la performance est-elle mesurée? .	26
2.4 Conclusion .	28
Chapitre 3. Vidéo à la demande .	31

3.1 Introduction .	31
3.1.1 Multimédia: une définition . .	31
3.1.2 Types de données multimédias .	32
3.1.3 La compression des données multimédia .	35
3.1.4 Applications multimédia .	36
3.2 Systèmes VoD .	37
3.2.1 Introduction . .	37
3.2.2 La communication dans un système VoD .	38
3.2.3 Le serveur VoD . .	42
3.2.4 Le lecteur VoD .	48
3.3 Facteurs de performance pour un système VoD .	50
3.3.1 Les critères de performance d'un système VoD . .	50
3.3.2 Le système de stockage . .	51
3.3.3 Le réseau .	56
3.3.4 Autres facteurs .	57
3.4 Système Audio à la Demande. .	57
3.5 Conclusion .	58
Chapitre 4. Méthodologie pour la mesure des performances d'un serveur VoD . .	59
4.1 La mesure des performances d'un système VoD . .	59
4.1.1 La qualité de la perception (Quality of Perception - QoP) .	60
4.1.2 Les facteurs de performance pour un serveur VoD .	62
4.2 Méthodologie de tests . .	65
4.3 Résultats expérimentaux . .	67
4.3.1 Plate-forme des tests .	67
4.3.2 Les performances d'un serveur VoD .	69
4.4 Conclusion .	71
Chapitre 5. Une stratégie d'utilisation du protocole HTTP pour la VoD .	73
5.1 Introduction .	73
5.1.1 Pourquoi utiliser HTTP dans la VoD ? .	74

5.1.2 La stratégie de transfert HTTP actuelle .	75
5.1.3 Les défauts de la stratégie actuelle .	76
5.2 Présentation d'une nouvelle stratégie de transfert HTTP .	77
5.2.1 Avantages et inconvénients .	79
5.3 Analyse de la stratégie .	81
5.3.1 Analyse de la stratégie HTTP standard .	81
5.3.2 Analyse de la stratégie HTTP proposée . .	83
5.4 Implémentation .	89
5.4.1 Qualité des fonctionnalités VCR de l'implémentation . .	90
5.4.2 Performances par rapport aux autres implémentations .	90
5.5 Résultats expérimentaux - les performances de plusieurs serveurs Web dans la VoD ..	92
5.5.1 Plate-forme de tests .	92
5.5.2 Méthodologie de tests . .	92
5.5.3 Performances .	92
5.6 Conclusion .	93
Chapitre 6. Design et implémentation d'un serveur Web optimisé pour la VoD .	95
6.1 Introduction .	95
6.2 Les performances d'un système matériel VoD .	96
6.2.1 Architecture d'un disque dur . .	97
6.2.2 Estimation de performances d'un disque .	99
6.2.3 Estimation des performances d'un système de stockage RAID . .	100
6.3 L'architecture du serveur . .	105
6.4 La gestion de la concurrence .	107
6.4.1 Modèle "thread per request" . .	107
6.4.2 Le contrôle d'admission .	107
6.5 Implémentation .	109
6.6 Etude de la performance . .	113
6.6.1 Par rapport à d'autres serveurs Web .	113

6.6.2 Par rapport à d'autres serveurs VoD .	114
6.7 Conclusion .	114
Chapitre 7. Conclusions et Perspectives . .	117
7.1 Résultats . .	117
7.2 Limites . .	117
7.3 Positionnement par rapport à d'autres résultats .	118
7.3.1 Stratégie d'utilisation de HTTP .	118
7.3.2 Méthodologie pour la mesure des performances d'un serveur VoD . .	118
7.3.3 Estimation des performances d'un système matériel VoD .	118
7.3.4 Conception d'un serveur Web optimisé pour le multimédia .	118
7.4 Perspectives industrielles .	119
7.5 Résultats réutilisables . .	119
Bibliographie- personnelle .	121
Bibliographie . .	123
Annexe. Système VoD basé sur des serveurs Web - étude de cas . .	133
L'architecture . .	133
Composants .	135
Administration: Media Administrator (MA) . .	135
Equilibrage de charges (load-balancing) . .	137
Lecteur multimédia .	138
Avantages et inconvénients par rapport aux autres systèmes VoD .	140
Fiabilité et haute-disponibilité .	140
Intégration avec le Web . .	141
Performances .	141
Exemple d'architecture concrète de système VoD .	141

Remerciements

Je tiens à remercier :

Monsieur Andrezj DUDA, professeur à l'ENSIMAG à Grenoble, et Monsieur Serge FDIDA professeur à l'Université Pierre et Marie Curie à Paris, d'avoir bien accepté d'être rapporteurs de cette thèse.

Monsieur Grigor MOLDOVAN, professeur à l'Université Babes-Bolyai à Cluj-Napoca, qui à accepté à se déplacer depuis la Roumanie pour faire partie du jury.

Monsieur Serge MIGUET, professeur à l'Université Lumière Lyon2, et Monsieur Jean-Marie PINON professeur à l'INSA de Lyon qui ont encadré ce travail de thèse pour leur soutien et pour leurs conseils concernant la grande aventure qui à été pour moi la thèse.

Monsieur Jean-Baptiste DUCATEZ, mon responsable technique à CSTI, pour son accueil, ses conseils et tout l'aide qu'il a su me procurer.

Je remercie également ma femme Mihaela, mes parents, mes amis et mes professeurs en Roumanie.

Ces travaux ont été menés dans le cadre d'une convention entre le laboratoire ERIC et la société CSTI, filiale du groupe CS Communications & Systems.

Résumé

Avec les avancées dans le domaine des réseaux d'ordinateurs, de nouveaux services interactifs sont développés sur Internet ou les Intranets, parmi lesquels la vidéo à la demande (en anglais : Video on Demand, VoD). Actuellement, le nombre de solutions VoD croît de manière continue, ces systèmes deviennent de plus en plus complexes et coûteux, sans pour autant répondre totalement aux besoins des utilisateurs.

Un certain nombre d'utilisateurs ne peuvent accéder aux serveurs VoD que par l'intermédiaire du protocole HTTP (le protocole standard sur le Web), en raison des firewalls (pare-feux) mis en place dans les réseaux pour filtrer tous les autres paquets de données. L'installation d'un système VoD nécessiterait la modification des architectures réseau actuelles, en ajoutant des serveurs et des protocoles réseau supplémentaires. Hormis leur coût élevé, les serveurs VoD nécessitent souvent des plate-formes matérielles et logicielles propriétaires, dont le système de stockage est lié au serveur. Cependant, les serveurs Web sont présents dans presque tous les réseaux, fonctionnent sur toutes les plate-formes avec des systèmes de fichiers standards et utilisent HTTP. Par conséquent, l'utilisation d'un serveur HTTP comme serveur VoD présenterait de nombreux avantages. Mais l'utilisateur d'un système VoD s'attend à disposer de fonctionnalités similaires à celles qu'il trouve sur un système de vidéo traditionnelle (Vidéo Cassette Recorder, VCR), c'est à dire les fonctions usuelles d'un magnétoscope (avance et retour rapide, arrêt sur une image, ralentir) ou d'un vidéo disque (incluant le saut à une plage aléatoire). Malheureusement, pour l'instant, les serveurs VoD utilisant HTTP souffrent d'un manque d'interactivité en terme de fonctionnalités VCR (notamment "saut" et "pause"), en raison principalement de stratégies d'utilisation de HTTP inadéquates.

Dans cette thèse nous avons analysé et mis en place *une stratégie d'utilisation de HTTP* qui permet l'utilisation efficace de ce protocole pour la VoD. Cette idée permet le développement d'une solution simple et efficace d'un système VoD, articulé autour de serveurs Web existants, utilisés comme serveurs VoD.

Afin de mesurer les performances d'un système VoD (et du notre en particulier), nous avons développé une méthodologie de tests (*benchmarking*) pour un serveur VoD. Nous avons défini la *qualité de perception*, une mesure qui reflète le degré de satisfaction d'un utilisateur qui regarde un flux vidéo en utilisant un système VoD. À partir de cette mesure, nous avons déterminé des facteurs équivalents au niveau du serveur. En mesurant ces facteurs, nous proposons une stratégie d'évaluation de performances pour les serveurs VoD qui permet de trouver les performances réelles des différents serveurs VoD en termes d'utilisateurs satisfaits.

De plus, pour optimiser les performances d'un serveur Web utilisant notre stratégie en VoD tout en garantissant à l'utilisateur une certaine qualité de perception, nous avons mis en place une *politique de contrôle d'admission* et différentes *politiques de travail avec le système de stockage*. Ceci nous permet d'atteindre des performances proches des limites du système matériel VoD. La conception d'un serveur Web qui implémente ces politiques est également présentée dans le cadre de cette thèse.

Mots clefs: multimédia, vidéo à la demande (VoD), protocole HTTP, serveur vidéo, benchmarking pour VoD.

Abstract

Most existing VoD (Video on Demand) systems are relatively complex and expensive. They rely on specialized protocols and use proprietary hardware and software.

This thesis discusses how to use the HTTP protocol and a standard web server to build an efficient VoD system. This strategy offers three major advantages :

- It can allow some users to bypass firewalls set up to filter the data that are not sent using HTTP.

- It does not require any modification of existing network architectures since it uses standard protocols, software and files systems existing on all platforms.

- It allows an efficient implementation of extended video playback functions (play, fast forward, fast backward, pause, stop, seek).

To compare the efficiency of our VoD system with existing ones, we propose a performance evaluation strategy for VoD servers in terms of satisfied users. This strategy relies on a measure called "quality of perception" that encompasses users satisfaction regarding multimedia presentations.

An admission control policy and various data transfer policies from the storage system are also proposed to optimize the performances of our VoD system. These optimizations allow to reach performances that are close to the physical limits of the VoD hardware system. The design of a Web server implementing these policies is finally presented.

Keywords: multimedia, video on demand (VoD), HTTP, video server, VoD benchmarking.

Chapitre 1. Introduction

1.1 Contexte de l'étude

Les avancées dans le domaine des réseaux informatiques, ont permis le développement de nouveaux services interactifs sur Internet ou les Intranets pour répondre aux besoins toujours plus importants et diversifiés des utilisateurs. Si aux débuts de l'Internet (les années '80) les transferts de données dans les réseaux concernaient surtout des formats très simples, comme le texte ou l'image, les données multimédias occupent aujourd'hui une part importante de la bande passante. Ces données sont celles qui posent le plus de problèmes pour leur transport, compte tenu de leurs particularités : volume important, contraintes temporelles etc.

Fournir le support et la technique de communication pour tous ces services est un défi adressé aux sciences et technologies de l'information et de la communication. Des normes plus ou moins adéquates standardisent les formats des données à transférer selon des protocoles de communication variés, sans pour autant répondre totalement aux besoins des utilisateurs.

Cette thèse s'inscrit dans ce contexte, en essayant d'apporter des solutions simples aux problèmes de transport générés par les données multimédias, et spécialement le transport de séquences audiovisuelles dans le domaine de la vidéo à la demande.

Nous tenons à justifier en préambule l'utilisation de certains termes anglo-saxons par l'absence, à notre connaissance, de termes français équivalents et consacrés dans le domaine de la vidéo à la demande.

1.2 Objectif de la thèse

Le principal objectif de cette thèse est d'étudier la mise en place d'une plate-forme permettant l'utilisation efficace des serveurs Web en tant que serveurs dédiés à la vidéo à la demande (VoD).

En 1998-1999 (date du début de cette thèse), les solutions proposées par les deux principaux acteurs du domaine de la VoD¹ utilisaient HTTP seulement comme alternative de communication à d'autres protocoles spécialisés pour le multimédia. L'examen des arguments proposés pour justifier ce positionnement ([ms-http], [rn-http]) nous a montré que si certaines critiques formulées à l'encontre de l'utilisation de HTTP dans le domaine de la VoD étaient fondées (les performances de TCP par rapport à UDP), d'autres ne l'étaient pas (implémentation efficace des opérations VCR en utilisant HTTP, notamment *Seek*). De plus, on sait que d'autres protocoles que HTTP ne sont pas capables de passer certains pare-feu, privant une partie d'utilisateurs d'accéder aux serveurs VoD.

Plusieurs questions doivent être abordées pour définir une bonne stratégie d'utilisation des serveurs Web en tant que serveurs VoD :

· existe-t-il des critères de qualité pour un système VoD ? si oui, peut-on les modéliser pour mieux contrôler l'utilisation d'un système VoD ?

· l'inefficacité actuelle de HTTP en matière de VoD est-elle liée aux caractéristiques intrinsèques de ce protocole ou à une stratégie d'utilisation inadaptée ? peut-on définir une stratégie d'utilisation alternative compatible avec le protocole HTTP actuel ?

· une stratégie d'utilisation adaptée permet-elle aux serveurs Web actuels d'atteindre les performances des serveurs VoD dédiés ? peut-on améliorer les serveurs Web existants pour les rendre plus performants en matière de multimédia ?

Afin de mesurer les performances d'un système VoD, nous avons développé une méthodologie de tests (*benchmarking*) basée sur la notion de *qualité de perception*, notion subjective reflétant le degré de satisfaction d'un utilisateur qui regarde un flux vidéo. Nous avons déterminé des facteurs permettant de quantifier ces mesures d'un point de vue objectif au niveau du serveur. En mesurant ces facteurs, nous proposons une stratégie d'évaluation de performance qui détermine le nombre d'utilisateurs satisfaits d'un système VoD.

¹ Microsoft (avec [ms-tiger], devenu [ms-netshow], devenu [ms-wmp]) et RealNetworks ([rn-rs])

Nous avons ensuite analysé et mis en place *une stratégie d'utilisation de HTTP* permettant un recours efficace à ce protocole pour la VoD. Cette stratégie permet le développement de systèmes VoD simples et efficaces utilisant les serveurs Web existants comme serveurs VoD.

Pour optimiser les performances d'un serveur Web utilisant notre stratégie en VoD tout en garantissant à l'utilisateur une certaine qualité de perception, nous avons enfin mis en place une *politique de contrôle d'admission* et différentes *politiques de travail avec le système de stockage*. Ceci nous permet d'atteindre des performances proches des limites du système matériel VoD. La conception d'un serveur Web qui implémente ces politiques est aussi présentée dans le cadre de cette thèse.

1.3 Principales contributions de cette thèse

1.3.1 Stratégie d'utilisation efficace de HTTP pour la vidéo à la demande

Nous avons mis au point une stratégie d'utilisation de ce protocole autorisant des interactions client-serveur performantes dans un système multimédia, contrairement aux implémentations actuelles (voir [ms-wms], [rn-rs]).

La méthode actuelle utilisée pour le transfert d'un objet à l'aide du protocole HTTP consiste à envoyer une requête HTTP au serveur pour chaque objet et à attendre la réponse. Le serveur envoie ensuite le plus vite possible la réponse contenant l'objet demandé. Les principaux défauts de cette approche sont le temps d'interaction important pour la réalisation d'un *seek* et l'utilisation inefficace de la bande passante.

Nous avons proposé un mode de transfert des objets multimédia utilisant plusieurs blocs de données, transmis indépendamment les uns des autres à l'aide de l'option "Range" du protocole HTTP1.1. Les avantages d'une telle utilisation (optimisant la mise en oeuvre de l'opération "seek" et contrôlant mieux la bande passante) ont été présentés dans deux conférences: [coresa99] et [isimade2000].

1.3.2 Méthodologie pour la mesure des performances d'un serveur VoD

Il n'existe pas de méthodologie de tests (benchmarking) pour mesurer les performances d'un système vidéo à la demande. Nous avons apporté des contributions dans ce domaine, en essayant de mettre au point une méthodologie de tests permettant de déterminer les performances maximales d'un serveur VoD en termes de nombre de clients satisfaits et de débit maximal. Cette méthodologie s'inspire des outils de benchmarking de serveurs Web mais tient compte des caractéristiques particulières des flux multimédia.

La base de cette méthodologie de test est la définition d'un facteur (*qualité de la perception*, défini dans [ict01]) mesurant le degré de satisfaction d'un client qui reçoit un flux multimédia. L'utilisation de ce facteur dans notre méthodologie permet la réalisation de tests objectifs sans recours à des témoins humains pour mesurer la qualité d'un flux multimédia.

Cette méthodologie de tests a été appliquée à l'étude comparative des performances pour la VoD des deux principaux serveurs Web du marchés (Internet Information Server et Apache) [PROMS2000]. Certaines réflexions qui nous ont conduits à définir la qualité de perception sont présentées dans [studia98].

Elle a par ailleurs été fortement utilisée dans le sein de l'entreprise CSTI pour calibrer les performances des systèmes VoD avant leur livraison au client.

1.3.3 Estimation des performances d'un système matériel VoD

Nous avons étudié en détail les performances d'un système de stockage constitué de plusieurs disques associés par des cartes RAID. Nous avons construit un modèle décrivant, à partir des caractéristiques physiques de ses composants, les performances maximales d'un système VoD utilisant ce système de stockage.

Ce modèle nous a permis d'étudier le rendement d'un serveur VoD en fonction de ses caractéristiques physiques. Nous avons observé des écarts de performances sensibles entre des serveurs VoD utilisant un même système physique, écarts principalement imputables à la stratégie d'utilisation du système de stockage.

Le modèle construit a trouvé une utilisation dans la société CSTI, où il permet de calculer exactement les caractéristiques techniques d'un serveur VoD en fonction de la demande d'un client, sans avoir à acquérir préalablement le serveur pour faire les tests. En pratique, étant donné le nombre de clients simultanés, le débit moyen d'un flux et le nombre d'heures de stockage nécessaires, notre modèle évalue le nombre minimal de noeuds, cartes RAID et disques SCSI nécessaires pour satisfaire les besoins.

1.3.4 Conception d'un serveur Web optimisé pour le multimédia

Les serveurs Web actuels ne sont pas optimisés pour une utilisation intensive dans le domaine du multimédia, même si leurs performances, avec une utilisation adéquate de HTTP sont du même ordre que celles des serveurs VoD spécialisés. Ils ne gèrent pas les connexions en termes de qualité de perception et, dans le cas où le serveur serait surchargé, toutes les connexions existantes peuvent être affectées.

Pour résoudre ce problème, nous avons mis au point un serveur Web basé sur le modèle d'un serveur Apache, mais implémentant aussi une politique de contrôle d'admission et une optimisation de la politique de stockage.

Des tests effectués à CSTI ont montré qu'une telle solution est très performante dans un contexte VoD, donnant un meilleur rendement sur la même configuration physique que d'autres serveurs VoD (notamment celui de CSTI), tout en gardant une bonne qualité

d'interaction client-serveur. Une partie de ces résultats ont été présentés dans [PROMS2000].

Les travaux décrits dans cette thèse ont été à la base d'une solution commerciale "système vidéo à la demande", commercialisée par "Communications et Systèmes – Technologies Informatiques" (CSTI). CSTI a déplacé ses activités à Paris en 2000. L'essentiel des tests présentés dans cette thèse ont été réalisés avant septembre 2000. Néanmoins, les données technologiques n'ont pas évolué de manière significative depuis cette période.

1.4 Organisation du document

La thèse débute par un état de l'art dans les domaines des serveurs Web (Chapitre 2) et du multimédia, plus particulièrement centré sur les systèmes de vidéo à la demande (Chapitre 3). Nous proposons dans le Chapitre 4 une réponse à la question de la mesure des performances d'un serveur VoD. L'utilisation habituelle de HTTP n'étant pas souhaitable dans le domaine de la VoD, nous présentons une stratégie d'utilisation meilleure, sans modifier le protocole HTTP, dans le Chapitre 5. Le même chapitre présente les performances des serveurs Web actuels en tant que serveurs VoD (Section 5.5).

Le serveur Web que nous avons conçu pour exploiter au maximum les performances de HTTP pour la VoD est décrit dans le Chapitre 6.

Après les conclusions tirées des nos travaux (Chapitre 7) nous proposons une solution plus générale comme serveur VoD distribué basé sur des serveurs Web (Annexe 1).

Chapitre 2. Serveurs Web

2.1 Introduction

Durant les dernières années l'Internet a connu un développement phénoménal. Plusieurs facteurs ont favorisé cette croissance :

- l'utilisation des navigateurs Web gratuits (NCSA Mosaic , Netscape Navigator, Microsoft Internet Explorer),
- la facilité d'utilisation,
- la tendance de partage de l'information entre les chercheurs, les institutions officielles et les entreprises commerciales,
- l'indépendance des protocoles et des langages utilisés pour la construction et le transfert des documents Web vis-à-vis des plate-formes matérielles.

Le trafic sur le réseau double chaque année, tandis que les coûts des équipements de

communication et de routage ne baissent que de 30% ([pc-mag]). Le trafic n'a pas seulement évolué en quantité, la nature des informations transportées a été également modifiée. Le passage du Web d'outil de recherche vers un "bien de consommation" a entraîné une transformation de contenu des serveurs, passant des documents austères mais informatifs à des documents faisant intervenir de nombreux média (son, vidéo) gourmands en bande passante. Il y a une demande toujours plus grande pour des flux vidéo et audio, ce qui pose un réel problème technique, l'infrastructure actuelle n'ayant pas été conçue pour répondre à ce type de demande.

Conséquence de cette croissance, l'Internet souffre des problèmes de congestion de trafic et des serveurs surchargés. Les travaux effectués dans cette thèse s'inscrivent dans les efforts faits pour la décongestion du trafic en optimisant le transfert des séquences audiovisuelles sur le Web. La croissance de performance des serveurs Web pour des requêtes multimédia a été aussi étudiée, elle est cruciale pour les performances du Web.

Ce chapitre présente un état de l'art sur le Web, le protocole HTTP et les serveurs Web.

2.1.1 World Wide Web

L'Internet est un réseau global, connectant des millions d'ordinateurs. En 1999 il y avait plus de 200 millions d'utilisateurs d'Internet dans le monde entier, et ce nombre est en croissance continue ([webopedia]). Au sein de l'Internet on a des serveurs qui savent travailler avec un protocole spécial (HTTP) pour transmettre des documents avec un format spécial (HTML). Ces serveurs constituent le Web (World Wide Web), un sous-système de l'Internet (Figure 1).



Figure 1 Le Web est un sous-système de l'Internet

Le Web est basé sur trois mécanismes pour rendre disponible ses ressources à une audience la plus large possible :

- une schéma uniforme pour adresser les serveurs (i.e. URL)
- des protocoles pour accéder aux ressources (e.g. HTTP)
-

un format pour naviguer entre les ressources (e.g. HTML)

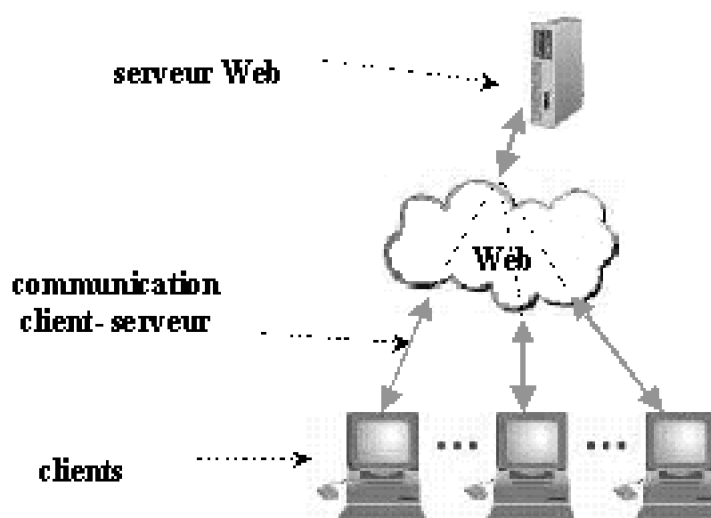


Figure 2 Modèle client-serveur pour le Web

Le modèle qui est à la base du Web est un modèle client-serveur. Le processus de communication utilise des paires requête-réponse, et le client est toujours celui qui l'initie. Un client utilise un navigateur Web pour accéder aux ressources Web. Quand l'utilisateur demande un document, le navigateur crée une nouvelle requête qui est envoyée au serveur Web correspondant. Chaque page Web peut contenir un ou plusieurs fichiers, et chaque fichier est demandé séparément au serveur.

Le serveur Web répond à chaque requête reçue des clients Web. La réponse du serveur contient le document demandé - si la requête a été validée - et un code pour informer le client sur l'état de la requête (valide, erreur etc.).

2.1.1.1 Définition d'un serveur Web

Un serveur Web (où serveur HTTP) est un serveur qui communique avec les clients en utilisant le protocole HTTP ([http]). Chaque serveur Web a une adresse IP (e.g. 192.168.116.165) et un nom de machine appartenant à un domaine (e.g. soft15.csti.fr). D'après une statistique réalisée par [netcraft], il existe maintenant (juillet 2000) plus de 18 millions de serveurs Web repartis dans le monde, la plus grande partie utilisant Apache (Figure 3).

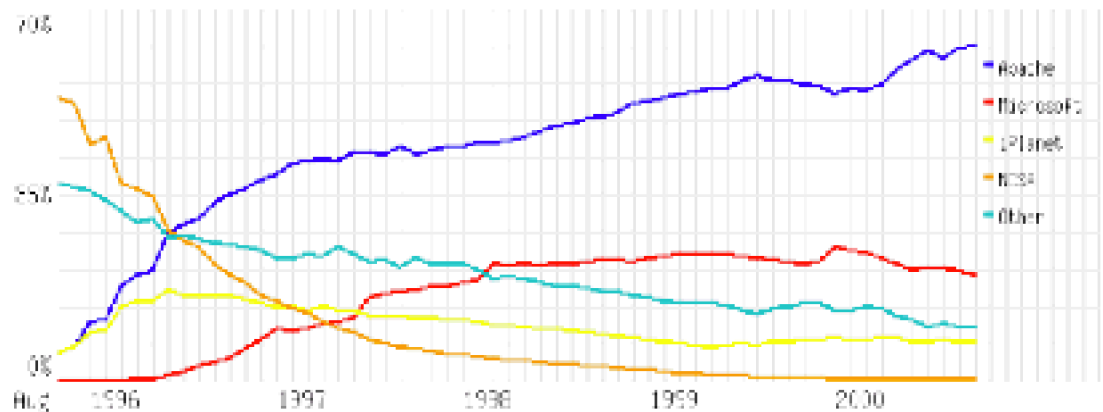


Figure 3 La distribution des serveurs Web sur Internet en pourcentage de marché ([netcraft] - juillet 2000)

2.1.2 Le protocole HTTP

Le protocole HyperText Transfer Protocol (HTTP) est le langage utilisé par les clients et les serveurs Web pour communiquer les uns avec les autres. L'histoire du standard HTTP a connu trois versions (0.9, 1.0 et 1.1), chacune d'entre elles apportant des améliorations aux versions antérieures, tout en gardant les standards déjà en place.

Toutes les transactions HTTP respectent le même format, détaillé en [http1.0], [http1.1]. Les requêtes et les réponses contiennent trois parties : la ligne avec la requête (ou avec la réponse), une section d'en-tête et le corps de la requête (la réponse), comme l'exemple présenté dans la Figure 4 .

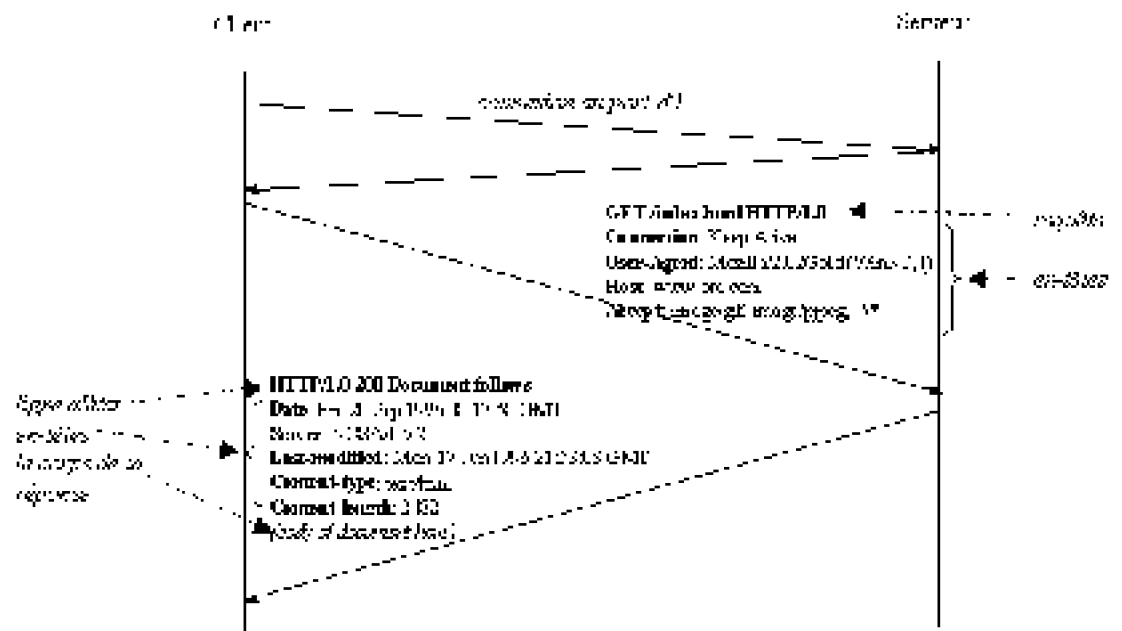


Figure 4 Un exemple de transaction HTTP

2.1.2.1 Fonctionnement du protocole HTTP

Connexion (initié par le client) : La connexion se fait, à travers une connexion TCP-IP, en mentionnant le nom du serveur ou son numéro IP, ainsi le numéro de port donné dans l'adresse. Dans le cas où ce dernier n'est pas spécifié, le port 80 est utilisé par défaut.

Si HTTP fonctionne le plus souvent sous TCP, il peut tout aussi bien être implémenté à l'aide de n'importe quel autre service en mode connecté.

Requêtes

Le client envoie une requête de document qui consiste en une ligne écrite en caractères ASCII se terminant par "CR LF" (Carriage Return, Line Feed). Cette requête n'est autre que le mot "GET", un espace et l'adresse du document, en omettant la partie de l'HTTP : l'hôte et le port.

L'adresse du document ne doit comporter qu'un seul mot, tout autre mot est ignoré. La fonctionnalité du processus de recherche repose sur la capacité à transformer une adresse donnée en une recherche d'index. Ex. :

GET /index.html HTTP /1.0 Le client peut envoyer des informations en-tête supplémentaires pour informer le serveur de sa configuration et les formats qu'il accepte. Toutes les informations en-tête sont données ligne par ligne, chaque ligne avec le nom d'en-tête et la valeur associée. Ex. :

Connection: Keep-Alive

User-Agent: Mozilla/2.02Gold (WinNT; I)

Host: www.ora.com

Accept: image/gif, image/pjpeg, */*

Après la requête et les en-têtes, le client peut envoyer des données additionnelles (le corps de la requête).

Réponse

Le format de réponse est tout aussi simple, il débute par une ligne d'état contenant la version du protocole utilisé par le serveur, ainsi qu'un code résultat et éventuellement un message. Ce premier flux d'informations est constitué de caractères ASCII. Ex. :

HTTP /1.0 200 Document follows Ceci, suivi par une série d'en-têtes dont les plus importants sont "type du contenu", qui décrit le type de l'objet qui est donné, et "longueur du contenu", qui indique la longueur du document. Les entêtes se terminent par des lignes vides. Ex. :

Date: Fri, 20 Sep 1996 08:17:58 GMT

Server: NCSA/1.5.2

Last-modified: Mon, 17 Jun 1996 21:53:08 GMT

Content-type: text/html

Content-length: 2482

Le serveur peut maintenant envoyer n'importe quelles données. Après l'envoi du document, le serveur coupe la connexion.

Déconnexion

La connexion TCP-IP est coupée par le serveur quand tout le document a été transmis. Le client peut avorter le transfert en coupant lui-même la connexion avant la fin du transfert, dans ce cas le serveur n'enregistrera aucune erreur.

2.1.3 HTTP 1.0 versus HTTP 1.1

Le grand nombre d'objets constituant une page Web représente un changement de l'environnement pour lequel le protocole HTTP1.0 a été conçu. Pour afficher la page Web, un navigateur doit faire plusieurs requêtes au serveur, une requête pour chaque objet intégré dans la page (image, script, etc.). En conséquence, le mécanisme de gestion de ces connexions est très important pour la performance d'un navigateur Web.

Avec le protocole HTTP/1.0, pour chaque nouvelle requête le client réalise une nouvelle connexion TCP avec le serveur. Mais pour établir une connexion TCP, un nombre de paquets supplémentaires est échangé, ce qui prend un temps conséquent. Ce problème est connu sous le nom de "démarrage lent HTTP" (*HTTP slow-start*). Si pour chaque requête on doit attendre que la connexion TCP se réalise, les performances de transfert seront réduites d'une façon importante.

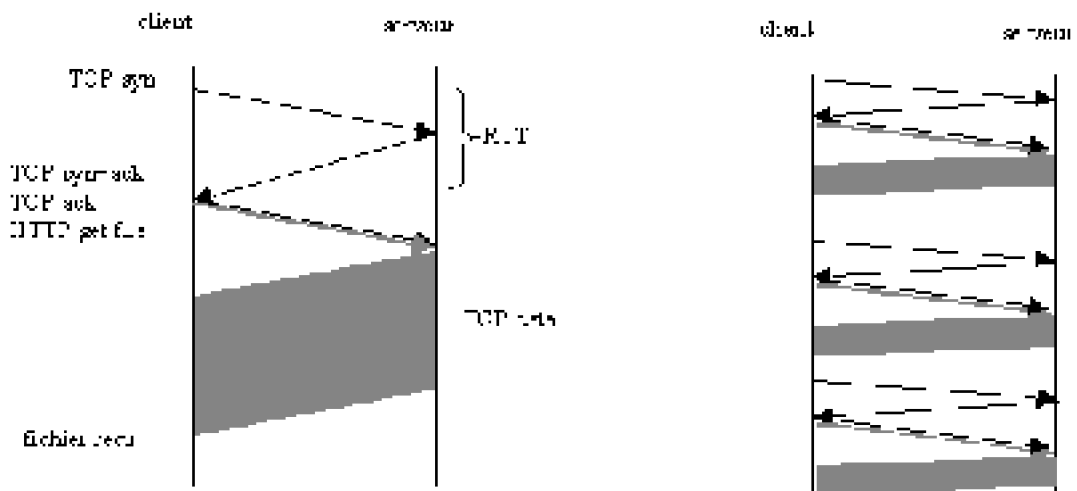


Figure 5 Une transaction HTTP1.0: pour demander un objet (gauche) ou plusieurs objets (droite)

L'analyse d'une transaction HTTP1.0 en termes de temps de transfert :

durée		action
1	RTT^2	ouverture d'une connexion TCP (l'équivalent de la commande TCP OPEN)
0.5	RTT	l'envoi de la requête
0.5	RTT	commencement du transfert du fichier réponse
F_{trans}		transmission du fichier
$2 RTT + F_{trans}$ = le temps nécessaire pour les transfert d'un fichier avec HTTP1.0		

Pour le transfert de plusieurs objets F^1, \dots, F^n avec le protocole HTTP1.0, le temps total de transfert est :

$$2n * RTT + F_{trans}^1 + \dots + F_{trans}^n \quad (1)$$

Comme RTT peut prendre des valeurs importantes, le protocole HTTP/1.0 gère d'une manière inadéquate les connexions Web client serveur en ouvrant chaque fois une nouvelle connexion HTTP.

La version 1.1 du protocole HTTP permet la réutilisation d'une connexion TCP déjà ouverte pour les autres requêtes adressées au même serveur (*connexion persistante*). L'utilisation des connexions persistantes a comme effet un important gain de performance, permettant le transfert beaucoup plus rapide de plusieurs objets sur la même connexion TCP ([Nielsen97]).

² Le Round Trip Time (RTT) est le temps que prend l'envoi d'un paquet d'un bout de la connexion à l'autre, plus le retour. Le RTT peut prendre 70 millisecondes ou plus.

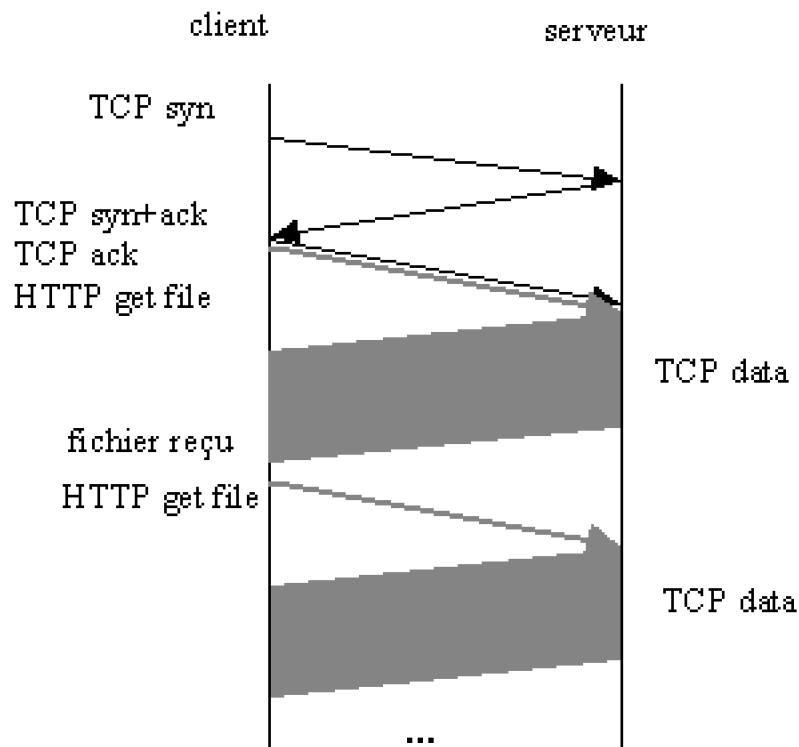


Figure 6 Une transaction HTTP1.1 pour le transfert de plusieurs objets (connexion persistante)

Pour le transfert de plusieurs objets F^1, \dots, F^n avec le protocole HTTP1.1 le temps de transfert est :

$$RTT = 0.5 * n * RTT + F_{trans}^1 - \dots - F_{trans}^n \quad (2)$$

Par rapport à HTTP1.0 on obtient un gain de $(1.5*n-1)*RTT$.

Une autre optimisation apportée par HTTP/1.1 au transfert des données sur le Web consiste en la possibilité d'utiliser un mécanisme de *pipelining* sur les connexions persistantes. Cela signifie qu'un client peut démarrer plusieurs requêtes sans attendre chaque réponse, afin de mieux utiliser une connexion TCP (un temps de latence global plus court). La seule restriction est que l'ordonnancement des réponses doit être le même que pour les requêtes envoyées.

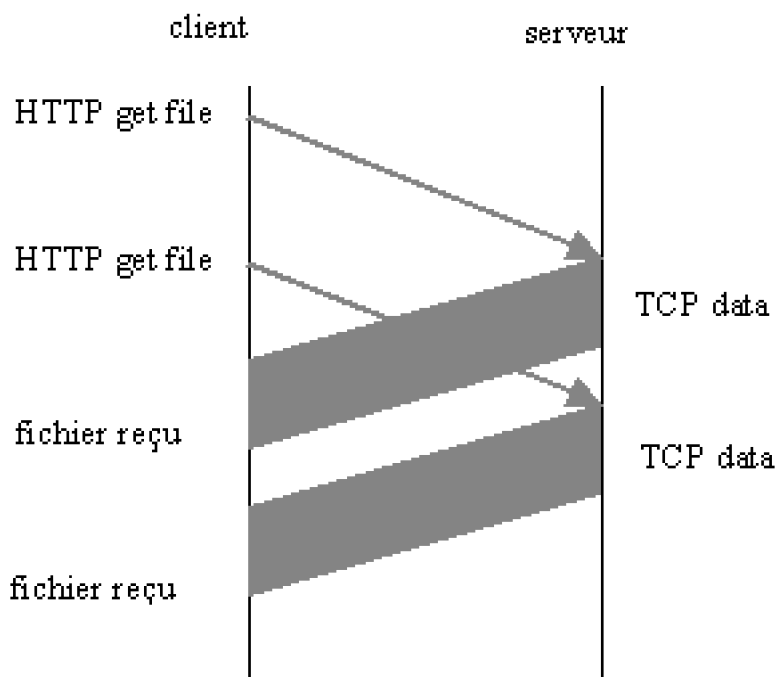


Figure 7 Le mécanisme "pipeline" autorisé dans HTTP/1.1

2.2 Principes de base pour le développement des serveurs Web

Les performances d'un serveur Web dépendent en bonne partie de son architecture, notamment la stratégie d'implémentation de la concurrence. Dans cette section nous présentons les principales stratégies de concurrence, ainsi qu'une autre fonctionnalité très importante, celle de réaliser des serveurs modulaires composés de plusieurs machines. Ces serveurs permettent d'obtenir plus de performance tout en gardant les caractéristiques d'un serveur unique, vu de l'extérieur (même URL).

2.2.1 Stratégies de concurrence

Pour mieux répondre aux besoins des clients qui accèdent simultanément à leurs ressources, les serveurs Web doivent traiter plusieurs requêtes à la fois. Pour cette raison ils doivent utiliser une *politique de concurrence* qui permette l'exécution simultanée de plusieurs tâches. Par rapport à cette politique de concurrence, on peut distinguer plusieurs types de serveurs Web ([Hu98], [McGrath96]) :

itératif : un serveur Web qui n'implémente aucune stratégie de concurrence au niveau applicatif; chaque tâche est traitée jusqu'à la fin.

concurrent avec un seul fil d'exécution (single-threaded concurrent) : un serveur Web avec un fil d'exécution de contrôle servant chaque tâche en attente jusqu'à ce qu'une quantité prédéterminée de travail soit réalisée ou qu'un *timer* ait expiré.

un fil d'exécution par requête (thread-per-request) : chaque tâche s'exécute avec son propre fil d'exécution de contrôle.

groupement pré-établi de fils d'exécution (thread-pool) : le nombre de fils d'exécution est établi d'avance, et les tâches sont réparties aux fils d'exécution disponibles.

un processus par requête (process-per-request) : pour chaque nouvelle requête un nouveau processus est créé (l'équivalent de la commande UNIX *fork*).

Les politiques de concurrence présentées ci-dessus ont des avantages et des inconvénients, et en fonction de leurs performances on a plusieurs générations de serveurs Web ([mcgrath96]):

1.
serveurs Web de la première génération : les serveurs qui démarrent un nouveau processus pour chaque nouvelle requête (20 requêtes par seconde – *rps* - sur un SPARCstation20 à 75 MHz – performances mesurées en [mcgrath96-2]).

2.
serveurs Web de la deuxième génération : les serveurs de type *thread-per-request* ou *thread-pool* (performances : 100 rps).

3.
serveurs Web de la troisième génération : les serveurs de type *thread-per-request* ou *thread-pool* qui utilisent les connexions persistantes du HTTP1.1 (performances : 300 rps).

2.2.2 Architecture d'un serveur Web

L'architecture d'un serveur Web travaillant avec une politique de concurrence *thread-per-request* est décrite dans la Figure 8. Un thread principal reçoit les connexions arrivant sur le port TCP 80 et pour chaque nouvelle connexion il crée un nouveau thread. Ce dernier sert les requêtes qui arrivent sur la connexion (une seule requête pour HTTP/1.0 ou plusieurs requêtes dans le cas de HTTP/1.1).

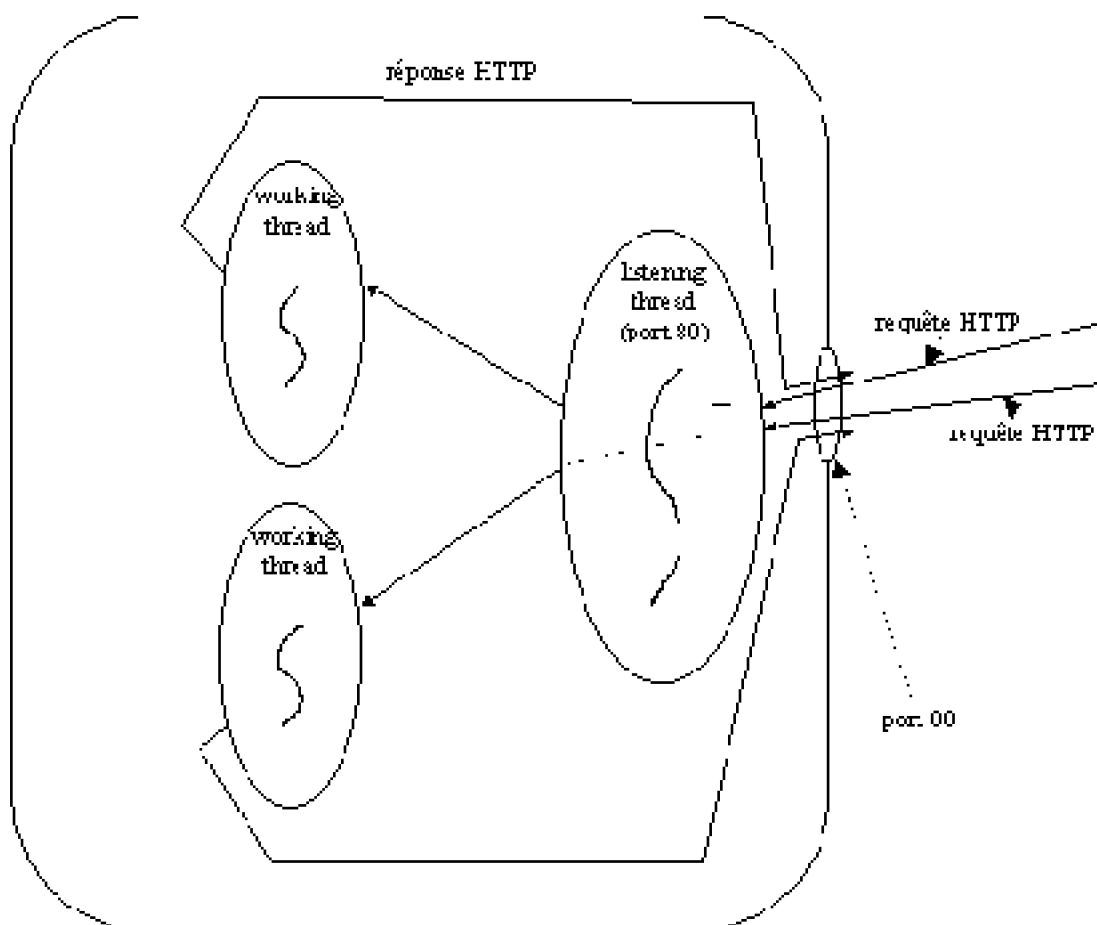


Figure 8 L'architecture d'un serveur Web (thread-per-request)

Le diagramme des états pour un fil d'exécution (thread) qui répond aux requêtes provenant d'un client HTTP est décrit dans la Figure 9. Le thread est dans un état *Ready* où il attend des requêtes de la part du client. Dès qu'il reçoit une requête, il passe dans l'état *Parse*. Si la commande n'est pas valide, le thread enregistre l'erreur dans un fichier texte avec un format spécial (fichier *log*) et il s'arrête (si HTTP/1.0) ou revient dans l'état *Ready* (si HTTP/1.1). Si la requête est valide, elle sera traitée dans l'état *Process*, où le serveur construit et envoie la réponse.

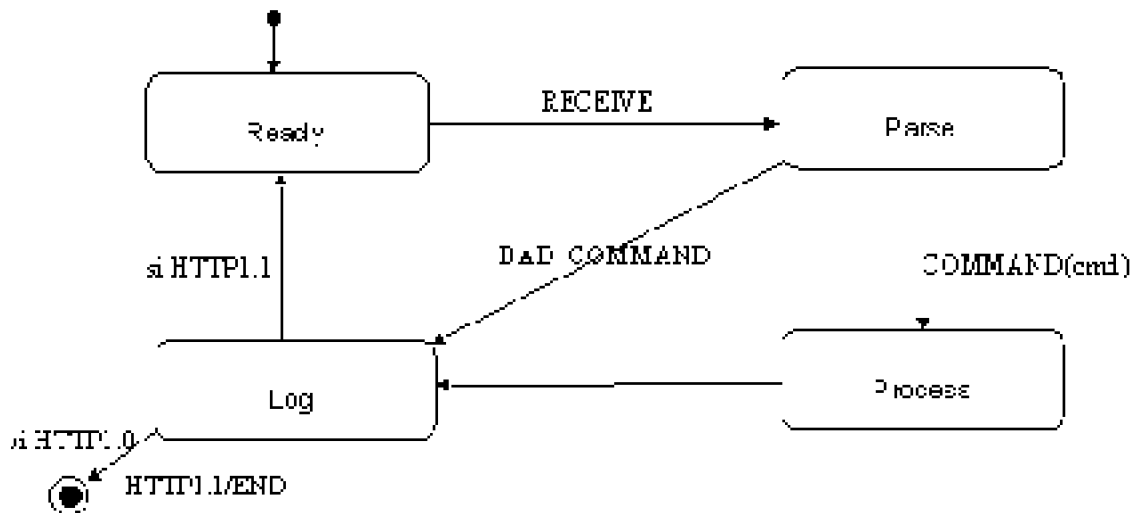


Figure 9 Un diagramme d'états schématisé pour un thread répondant à un client HTTP

Ce diagramme des états est important afin de déterminer les facteurs qui influencent les performances d'un serveur Web. D'après [almeida96], l'état *Log* apparaît pour 15% du temps d'exécution d'un serveur Web chargé. Le temps passé dans l'état *Parse* dépend principalement de la puissance de calcul du serveur, et généralement il est négligeable pour des requêtes statiques (pages Web, fichiers). La majeure partie du temps est passée dans l'état *Process*, où les performances du système de stockage et du réseau déterminent le temps de réponse à une requête statique.

2.2.3 Serveurs Web modulaires (scalables)

Avec l'explosion du Web, les sites Web les plus visités doivent être capable de répondre à un grand nombre de requêtes. Des sites comme *www.microsoft.com* reçoivent plus de 100 millions de requêtes chaque jour (plus de 1200 requêtes par seconde [ms- technet]) et la disponibilité d'un tel site doit être assurée quoi qu'il arrive (par exemple, si un serveur tombe en panne le site doit rester utilisable). Par conséquent, des solutions contenant plusieurs serveurs Web accédant aux mêmes données ont été mises en place.

Une grande partie de ces architectures ont comme modèle le prototype NCSA ([Kwan95]) et le prototype IBM construit sur un système IBM SP-2 ([Dias96]). Le principe consiste dans l'utilisation de plusieurs *noeuds* intégrés dans une architecture grappe autour d'un réseau haut-débit (Figure 10). Les noeuds sont de deux types :

des noeuds de stockage (back-end)

des noeuds de transfert – l'équivalent d'un serveur Web (front-end)

Le contenu du système peut être distribué sur tous les noeuds de stockage ou chaque noeud peut contenir une copie locale du site. Les meilleures performances sont obtenues

avec tout le contenu distribué sur tous les noeuds de stockage en utilisant un procédé de *stripping*.

Un module d'équilibrage des charges (*load balancing*) est utilisé pour répartir les requêtes arrivant de l'extérieur vers les noeuds qui contiennent les serveurs Web. Chaque nouvelle requête est répartie par le module de *load balancing* à un noeud serveur Web après un algorithme heuristique établi d'avance: *round robin*, *least-connections*, *maximum-connections* etc. (plus de détails sur les possibilités de load-balancing sont présentées dans [cisco], [radware], [resonate]) Ensuite le serveur Web qui reçoit la requête répond en utilisant les noeuds de stockage reliés à lui par un système de fichiers distribué.

Un exemple simplifié d'un serveur Web modulaire est présenté en Figure 10. Ce modèle a constitué le coeur du site Web des Jeux Olympiques d'Hiver Nagano 1998, site enregistré dans le "Guinness Book of World Records" comme l'événement Internet le plus populaire jamais enregistré, avec 634.7 millions de requêtes en 16 jours [iyengar00].

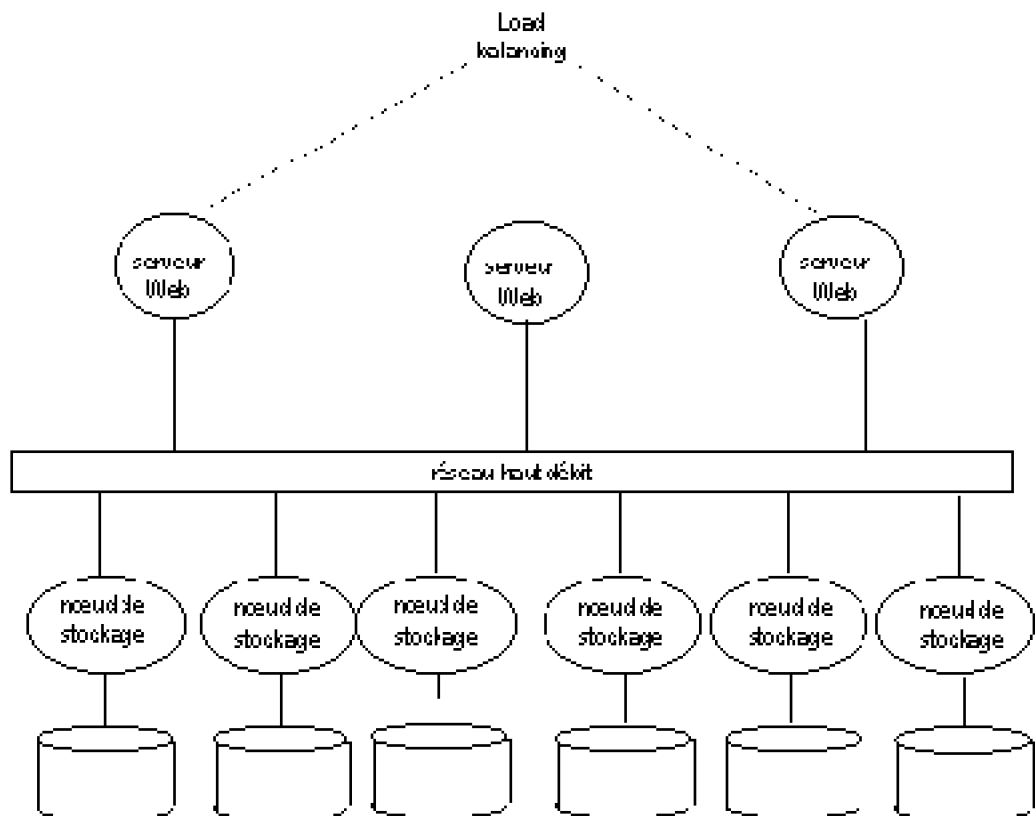


Figure 10 L'architecture d'un système modulaire de serveurs Web

2.3 Evaluation des performances

Les performances d'un serveur Web dépendent d'un grand nombre de variables : les

caractéristiques du système matériel, le système d'exploitation, le réseau, la conception de l'application. La perception de la performance côté client dépend aussi de la plate-forme client et de son navigateur Web. Plusieurs techniques sont utilisées par les navigateurs Web pour améliorer la perception de la performance côté client comme :

- l'affichage du cadre d'une image tant que l'image est en cours de transfert

- l'utilisation de plusieurs connexions TCP en parallèle,

mais le plus important est que le serveur Web puisse servir le maximum de clients dans un délai raisonnable.

2.3.1 Métriques pour mesurer la performance d'un serveur Web

Quatre métriques sont utilisées pour mesurer la capacité d'un serveur Web ([McGrath96], [Rubarth96]):

- *le nombre des requêtes traités par seconde* (unité de mesure: *rps* ou *HTTPops/sec*) : est une mesure du nombre de requêtes gérées par un serveur Web dans une période de temps déterminée; les requêtes peuvent être adressées pour des fichiers statiques de différentes tailles, mais aussi pour des fichiers HTML dynamiques, des commandes CGI ou accès aux bases de données à travers différentes API. Si les requêtes arrivent avec une fréquence plus grande que la capacité du serveur, elles ne sont pas traitées.

- *le débit* (unité de mesure: octets par seconde) : la quantité maximale de données que le serveur Web peut transmettre sur toutes les connexions HTTP ouvertes pour une durée de temps prédéterminée (le débit dépend fortement de la bande passante du réseau).

- *la latence* d'une requête (*round-trip time* en anglais, unité de mesure: RTT): le temps nécessaire pour commencer à répondre à une requête après l'établissement de la connexion.

- *le nombre d'erreurs* : les requêtes non-traitées par le serveur en raison d'un très grand nombre de requêtes reçues.

2.3.2 Comment la performance est-elle mesurée?

Pour évaluer les performances d'un serveur Web nous avons à notre disposition deux méthodes : *l'étude de données opérationnelles* et *les procédures de tests artificielles*. *L'analyse des données opérationnelles* concerne l'utilisation des diverses traces laissées dans le fichier de *log* ou d'autres outils surveillant le trafic sur le réseau, le système

d'exploitation ou le serveur Web ([Arlitt97]). Les résultats reflètent la pure réalité, mais elles dépendent de trop de variables pour permettre de faire des comparaisons entre les différentes configurations. Par conséquent des *techniques artificielles (benchmarking)* sont utilisées afin de reproduire les mêmes tests sur des configurations différentes, mais restant proche des scénarios réels ([McGrath96], [Almeida96], [Mogul95]).

2.3.2.1 Techniques d'évaluation de performances (*benchmarking*)

Une procédure d'évaluation de performances utilise un jeu de données prédéfini et mesure les résultats retournés par le système. Le but est de permettre une comparaison facile entre plusieurs systèmes exposés aux mêmes tests. Pour que les résultats prédisent la performance réelle il faut que le jeu des données utilisées soit une image fidèle du comportement des utilisateurs réels.

La procédure d'évaluation de performances pour un serveur Web consiste à simuler l'activité du serveur en utilisant plusieurs processus clients qui envoient des requêtes Web au serveur d'après une loi prédéfinie, imitant l'activité des navigateurs Web (Figure 11). Les clients varient les types des requêtes HTTP (par exemple GET, POST, PUT), la taille et le nombre des fichiers demandés, la fréquence d'envoi des requêtes, avec le but de trouver les performances maximales du serveur. Les réponses du serveur sont mesurés habituellement avec les normes déjà présentées (*nombre des requêtes traités par seconde, temps de réponse, débit, nombre d'erreurs*).

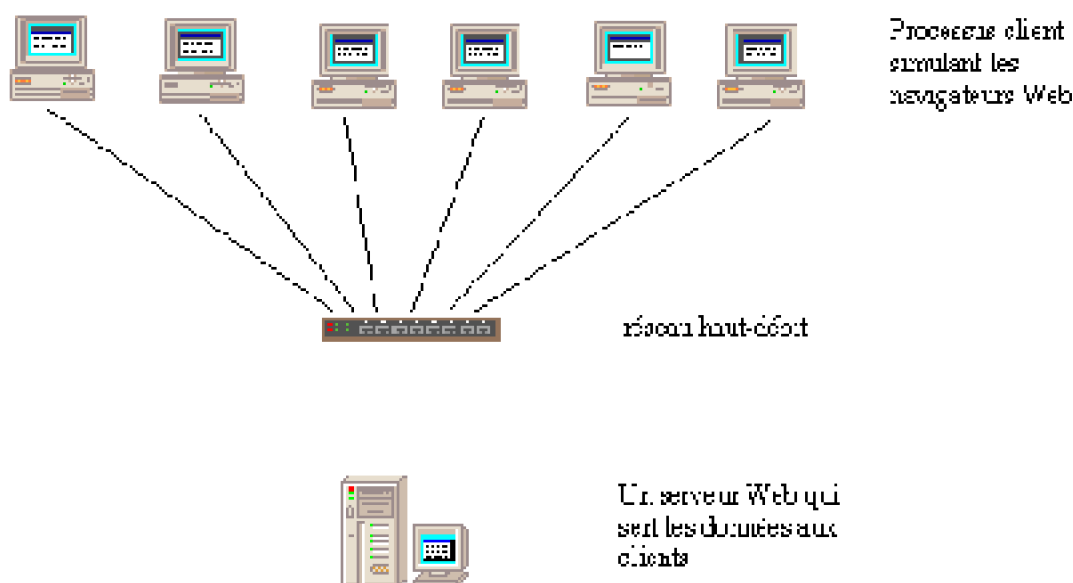


Figure 11 Procédure d'évaluation de performances pour un serveur Web

Les processus client essaient de trouver la sollicitation maximale que le serveur admet. Quand les tests s'arrêtent l'outil d'évaluation de performances calcule un score général pour la configuration testée. Les outils de benchmarking les plus répandues pour les serveurs Web sont SPECWeb96 (avec la version plus récente SPECWeb99, [specweb99]), WebBench ([webBench]) et WebStone ([trent95]). Elles calculent le nombre

maximal de requêtes par seconde et le débit maximal en octets par seconde.

Le jeu des données utilisé par serveur (nommée *workload*) est construit par l'instrument d'évaluation de performances en utilisant une répartition statistique de la taille des fichiers qui représente le mieux possible les scénarios réels. Par exemple, le jeu des données de WebSpec99 utilise quatre classes de fichiers avec des fréquences d'accès différentes (Figure 12).

Figure 12 Description du jeu des données utilisé dans WebSpec99

Classe de fichiers	Taille des fichiers	Fréquence d'accès
Class 0	0 -- 1KB	35%
Class 1	1KB -- 10KB	50%
Class 2	10KB -- 100KB	14%
Class 3	100KB -- 1MB	1%

Les performances des configurations récentes (septembre 2000), testées avec WebBench se situent autour de 2500 rps³ (un processeur) ou 7000 rps (quatre processeurs).

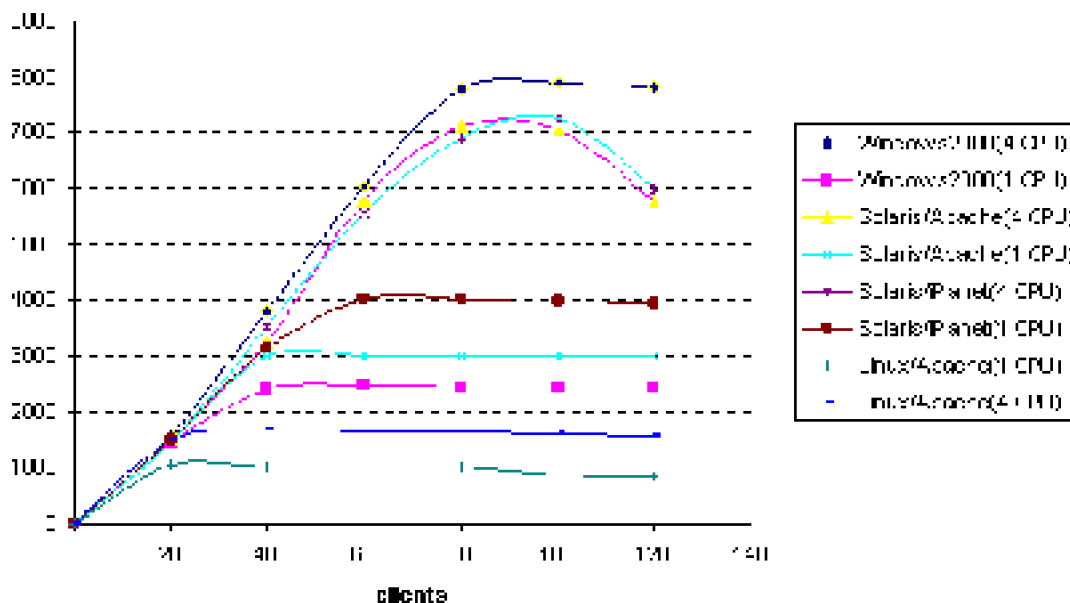


Figure 13 Les performances des plusieurs plate-formes Web testées avec WebBench ([pcmag])

2.4 Conclusion

³ rps: requêtes par seconde; unité de mesure pour la performance d'un serveur Web; représente le nombre de requêtes gérées par un serveur Web dans une période de temps déterminée

Les instruments d'évaluation de performances actuels ne sont pas adéquats pour la mesure d'un serveur Web dans un contexte particulier, comme celui du multimédia. Ils s'intéressent à la mesure des deux facteurs : *le nombre de requêtes par seconde* servies par le serveur et *le débit* du serveur. Contrairement aux clients Web, un client qui reçoit un flux multimédia réagit à plusieurs variables : la qualité de l'image, le nombre d'images par seconde, la qualité du son, la synchronisation audio-vidéo etc. Comme conséquence, si on souhaite mesurer la capacité d'un serveur Web en termes des flux multimédias on doit mettre au point une stratégie de d'évaluation de performances appropriée. Dans notre thèse nous présentons une telle stratégie et ses applications (Chapitre 4).

Les serveurs Web ont été conçus pour une utilisation générale: partage de l'information trouvée sur n'importe quelle forme (fichiers HTML, images, bases de données etc.). Dans le contexte particulier de multimédia est communément admis que les modèles de serveur Web actuels ne sont pas performants, et que de plus ils ne peuvent pas fournir les fonctionnalités de base d'un magnétoscope ou vidéodisque ([ms-http], [rn-http]). Nous montrons qu'une utilisation adéquate de HTTP nous permet de mettre en valeur l'architecture d'un serveur Web, en fournissant toutes les fonctionnalités que nous attendons d'un serveur vidéo à la demande, tout en gardant sa simplicité.

Chapitre 3. Vidéo à la demande

Dans ce chapitre nous présentons un état de l'art introductif du domaine de la vidéo à la demande. Premièrement, seront passés en revue les concepts fondamentaux du domaine multimédia : types de données multimédias, concept de flux, application multimédia, compression de données multimédias, etc. Ensuite, la notion de système Vidéo à la Demande (VoD) sera définie avec ses composants : le serveur et les clients, les protocoles et les stratégies de communication. Finalement nous mettons en évidence les principaux facteurs déterminants pour les performances d'un système VoD, parmi lesquels : le système de stockage, le réseau, la puissance de calcul.

3.1 Introduction

3.1.1 Multimédia: une définition

Pour commencer, nous cherchons une définition acceptable pour le terme *multimédia*. Les dictionnaires définissent *média* comme "support de diffusion massive de l'information" [petit-robert]. Par extrapolation, en informatique le mot *média* dénote les différents types de données utilisées comme support pour l'information : textes, sons, animations, vidéo. À partir de cette définition, certaines sources définissent *multimédia* comme étant la

présentation de l'information utilisant la combinaison de texte, sons, images, animations et vidéo ([britannica], [encarta]⁴).

Une autre définition présente le multimédia comme toute forme de représentation de l'information où intervient la synchronisation par rapport au temps ([nahrstedt95]). Nous utilisons ici cette définition, en considérant principalement comme information multimédia les données audio et vidéo, pour lesquelles la synchronisation (en temps) joue un rôle majeur. Pour certains auteurs, les médias qui dépendent du temps s'appellent *des media continus*. D'habitude, les médias continus sont constitués d'une séquence de petits échantillons (en anglais : samples) qui ont des dépendances temporelles strictes. Cette séquence d'échantillons constituant un média continu s'appelle un *flux* (en anglais : *stream*).

3.1.2 Types de données multimédias

En utilisant les définitions précédentes, nous distinguons deux grandes catégories de données multimédias élémentaires : *audio* et *vidéo*. Ces deux types des données sont décrits dans les deux prochaines sections.

3.1.2.1 Audio

Les sons (audio) peuvent être définis comme des oscillations de la pression de l'air qui stimulent les nerfs auditifs ([burger93]). Par leur nature, les sons ont une forme analogique, mais en informatique ils sont représentés sous une forme numérique (digitale). Par un procédé d'échantillonnage (Figure 14), à partir de la forme originale de l'onde et avec une certaine fréquence, des échantillons sont enregistrés pour garder la trace d'une séquence audio (séquence audio digitale). En effet, on peut reconstruire une onde sonore à partir des échantillons.

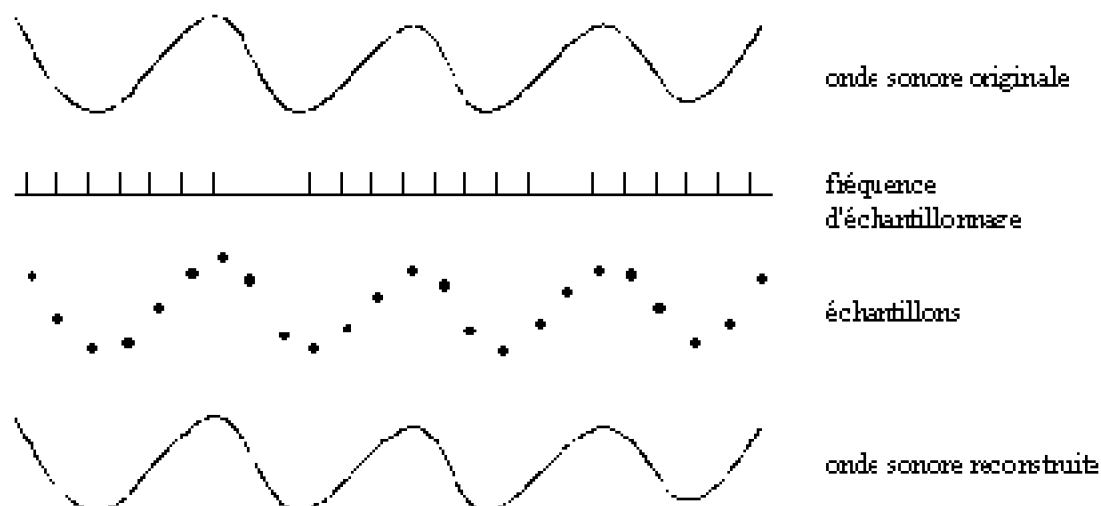


Figure 14 La digitalisation d'une onde sonore analogique

⁴ Multimédia = présentation de l'information utilisant toute combinaison de textes, images, sons, animation et vidéo.

En rapport avec la fréquence d'échantillonnage et avec la taille de chaque échantillon, nous trouvons plusieurs types de service audio, parmi lesquels (plus de détails dans [nahrstedt95]) :

Figure 15 Les paramètres de différentes ondes sonores digitalisées

Service	Taille de l'échantillon	Fréquence d'échantillonnage	Débit moyen
ligne téléphonique	8 bits	8 KHz	64 Kbps
son qualité mono	8 bits	22.01 KHz	176.08 Kbps
son qualité stéréo	16 bits	44.1 KHz	705 Kbps
son qualité CD – audio	32 bits	44.1 KHz	1.4 Mbps

3.1.2.2 Vidéo

Une séquence vidéo représente une succession d'images en mouvement bien définie dans le temps ([benoit97]) ou une séquence de cadres (en anglais : *frames*) où le mouvement dans la scène se reflète dans des petits changements dans les cadres affichés d'une manière séquentielle ([fdida97]).

Le concept de vidéo est très lié au concept de télévision. La télévision est un système capable d'envoyer et de recevoir images et sons par l'intermédiaire de signaux électroniques. Il existe actuellement plusieurs standards de télévision analogique, les plus importants étant NTSC, PAL et SECAM (pour une description détaillée, voir [benoit97]). D'une même façon que pour les données audio, la succession d'images analogiques peut être représentée sous une forme numérique.

Les professionnels de la vidéo utilisent dans leur studio des divers formats numériques pour l'enregistrement, la manipulation, le montage et la copie des signaux vidéo. Afin de faciliter l'interopérabilité des matériels et l'échange de programmes, plusieurs formats numériques ont été mis en place par les différents organismes de standardisation pour la télévision (les formats 4:2:2, 4:2:0, SIF, CIF). Mais tous ces formats génèrent un débit très important, qui les rendent pratiquement inutilisables pour la transmission vers les téléspectateurs sur un réseau (Figure 16).

Figure 16 Les principaux formats numériques pour la télévision

Normes TV	Format numérique	Format de compressé dérivé	Taille de l'image	Octets par pixel	Cadres par seconde	Débit non compressé (Mbps)	Débit compressé (Mbps)
	QCIF	MPEG1, 4	160*120	1.5	7.5	1.7	0.384
	CIF	MPEG1	360*288	1.5	30	37.3	1.5
NTSC LD	SIF	MPEG1	352*240	1.5	30	30.4	1.5
PAL LD	SIF	MPEG1	352*288	1.5	25	30.4	1.5
NTSC	Half D1	MPEG2	352*480	1.5	30	60.8	4
PAL	Half D1	MPEG2	352*576	1.5	25	60.8	4
	VGA		640*480	2	60	294.9	
	4:2:2		640x480		29.97	216	
	4:2:0	MPEG-2	640x480		29.97	162	15
NTSC	Full D1	MPEG2	704*480	2	30	162.2	15
PAL	Full D1	MPEG2	704*576	2	25	162.2	15
NTSC	CCIR 601	MPEG2	720*480	2	30	165.9	15
PAL	CCIR 601	MPEG2	720*576	2	25	165.9	15
	S-VGA		800*600	2	60	460.8	
NTSC HD		MPEG2	1440*1080	2	30	746.5	60
PAL HD		MPEG2	1440*1152	2	25	663.6	60
NTSC HD 16/9		MPEG2	1920*1080	2	30	995.3	80
PAL HD 16/9		MPEG2	1920*1152	2	25	884.7	80

Comme le présente la figure ci-dessus, il existe une palette assez large de formats TV numériques. Il est considéré qu'une bonne qualité de séquence visuelle est obtenue avec un échantillonnage de 1.5 octet par pixel, 25 ou 30 images par seconde, et une image de minimum 352 lignes et 480 colonnes [yes-tv]. Il en résulte un débit de plus de 60 Mbps non-compressé. Les demandes d'utilisateurs en termes de qualité font que les prochains postes TV vont utiliser la télévision haute-définition en format 16/9. Ces standards demandent des débits situés autour de 1 Gbps non-compressé, impensables pour les réseaux informatiques actuels.

3.1.2.3 Présentations multimédia

Généralement, les objets multimédias ne sont pas uniquement constitués de flux élémentaires (audio ou vidéo), mais contiennent habituellement plusieurs flux, synchronisés entre eux. Par exemple, un film peut contenir un flux vidéo et un flux audio. Sur les DVD nous pouvons avoir un flux vidéo avec plusieurs canaux audio en parallèle, représentant la traduction des dialogues en différentes langues. Un objet multimédia composé de plusieurs flux multimédias élémentaires est appelé objet multimédia agrégé ou présentation multimédia (plus de détails sur les présentations multimédia en [duda94], [Gibbs94]).

3.1.3 La compression des données multimédia

Les données multimédias numérisées occupent un volume important. En conséquence, l'une de leurs caractéristiques les plus importantes est le débit (mesuré en bits par seconde – *bps*, avec ses multiples⁵ *Kbps*, *Mbps* etc.). Les débits de quelques services multimédias ont été présentés dans la Figure 15 et dans la Figure 16. Ils montrent la difficulté de la transmission d'un objet multimédia sur un réseau (les bandes passantes de principaux types de réseaux actuels sont détaillé dans la Figure 39).

Pour économiser de l'espace et de la bande passante, les séquences audiovisuelles sont donc compressées. Il existe plusieurs formats de compression, mais le plus utilisé pour la vidéo grand public est le standard MPEG. MPEG utilise des techniques de prédiction avec compensation de mouvement pour déduire avec un minimum d'informations additionnelles la plupart des images de celles qui les précèdent ([benoit97-2]). Les différentes normes MPEG⁶ sont :

MPEG1 : une compression qui offre une qualité d'image similaire à celle d'un magnétoscope grand public et un son stéréophonique haute-fidélité - débit total 1.5 Mbps (voir [MPEG1]),

MPEG2 : une compression pour des sources DVB⁷; comme résultat on obtient de la vidéo haute qualité, avec un débit entre 3 Mbps et 15 Mbps (voir [MPEG2]),

MPEG4 : un très bon taux de compression pour des objets à bas débit - 10 Kbps, 20 Kbps, ..., 2Mbps (voir [MPEG4], [divx]).

La compression réduit beaucoup l'espace de stockage nécessaire pour stocker les objets multimédias, mais les objets ont encore de grandes dimensions (voir Figure 17).

Figure 17 Des débits d'objets multimédias compressés

⁵ 1 Kbps = 1000 bps, 1 KBps = 1024 bps, 1 Mbps = 1000 Kbps, 1 MBps = 1024 KBps (voir [tanenbaum])

⁶ Les standards MPEG sont décrites dans [mpeg1], [mpeg2], [mpeg4]

⁷ **DVB** : acronyme pour Digital Video Broadcast, la norme européenne de télévision numérique pour diffusion par satellite, câble ou réseau terrestre [dvb]

Type d'objet compressé	Débit	Taille de stockage nécessaire pour une heure de vidéo
MPEG1 (son qualité CD + vidéo qualité moyenne)	1.5 Mbps	644 MB
MPEG2 (son qualité CD + vidéo haute qualité)	6 Mbps	2575 MB
MPEG4 (son qualité CD + vidéo qualité faible)	500 Kbps	214.5 MB
MP3 (son qualité CD)	64 Kbps	28.8 MB

3.1.4 Applications multimédia

Les applications qui utilisent l'information multimédia s'appellent *des applications multimédia*. Nous nous intéressons aux applications multimédias réparties, impliquant des transferts de données multimédia entre plusieurs ordinateurs. Les principales applications multimédias distribuées sont présentées dans le tableau suivant, classées par rapport aux techniques de transmission des données :

Figure 18 Les mécanismes de transmission d'applications multimédia

Mécanisme de transmission	Définition	Applications
Broadcast	une seule source diffuse à tous les clients connectés	la télévision, le radio
Multicast	une seule source diffuse à tous les clients enregistrés	le radio et la télévision sur Internet (Webcasting), téléconférences
Unicast	une seule source diffuse à un seul client	vidéo a la demande

Une classe importante d'applications multimédias réparties sont les systèmes de distribution de services vidéo. Les principaux services vidéo (d'après [nwsu96, pp. 2-4]) sont :

CATV (Community Antenna Television) ou Simply Cable Television : les services sont envoyés aux clients à des instants fixes à partir d'une source vidéo; les consommateurs peuvent sélectionner un programme particulier pendant sa transmission, mais ils ne peuvent pas visualiser le programme après sa transmission ; que le consommateur regarde ou non un programme, il paie tout de même pour ce service ;

PPV (Pay-Per-View) : ce service est similaire au CATV, mais les programmes sont envoyés plus d'une fois dans une journée ; de cette manière les clients ont une certaine flexibilité dans leur emploi du temps pour regarder un programme ; de plus, le client ne

paie que pour les programmes auxquels il accède ;

VRC (Video Repository Center) : l'utilisateur peut obtenir une cassette vidéo d'un centre vidéo ; il a besoin d'un magnétoscope connecté à sa TV pour visualiser le film ;

VoD (Vidéo à la Demande) : dans un service VoD, le client a tout le contrôle de quand, comment et quel programme vidéo il regarde ; l'utilisateur dispose de tous les services de la CATV, PPV ou VRC avec plus de flexibilité ; comme sur un magnétoscope, il peut réaliser toutes les opérations VCR avec un délai minime.

La caractéristique principale des applications multimédia est le débit élevé de transmission des données. Il est dû à l'immense quantité de données nécessaire pour représenter les informations multimédias. Même si la quantité de données peut être réduite en utilisant différentes techniques de compression, les débits restent élevés. Différents mécanismes de communication sont utilisés, notamment le multicast pour éviter la duplication de l'information. Le revers du multicast est que l'utilisateur ne peut pas avoir des interactions de type VCR avec le contenu multimédia.

Nous nous intéressons dans cette thèse aux applications qui utilisent un mécanisme de communications unicast, particulièrement utilisé à la vidéo à la demande.

3.2 Systèmes VoD

3.2.1 Introduction

Les applications VoD sont des applications permettant aux clients de sélectionner des séquences audiovisuelles à partir d'un serveur central pour être visualisées sur un ordinateur, un téléviseur ou un set-top box (STB⁸). Cette technologie a d'importantes applications, notamment dans les domaines de l'hôtellerie, de l'enseignement, des mass-media etc.

On peut définir un *système VoD* comme étant un système intégrant un serveur et des clients interconnectés par un réseau ainsi que leur logiciels, conçu pour permettre aux clients d'accéder à un ou plusieurs flux multimédias depuis le serveur. Les flux sont servis d'une façon concurrente et indépendante, en temps réel et à la demande des utilisateurs.

L'architecture très générale d'un tel système est la suivante :

⁸ STB (acronyme pour Set Top Box) : terminal numérique connecté à un téléviseur, permettant la réception et le décodage des signaux câble, satellite ou numériques.

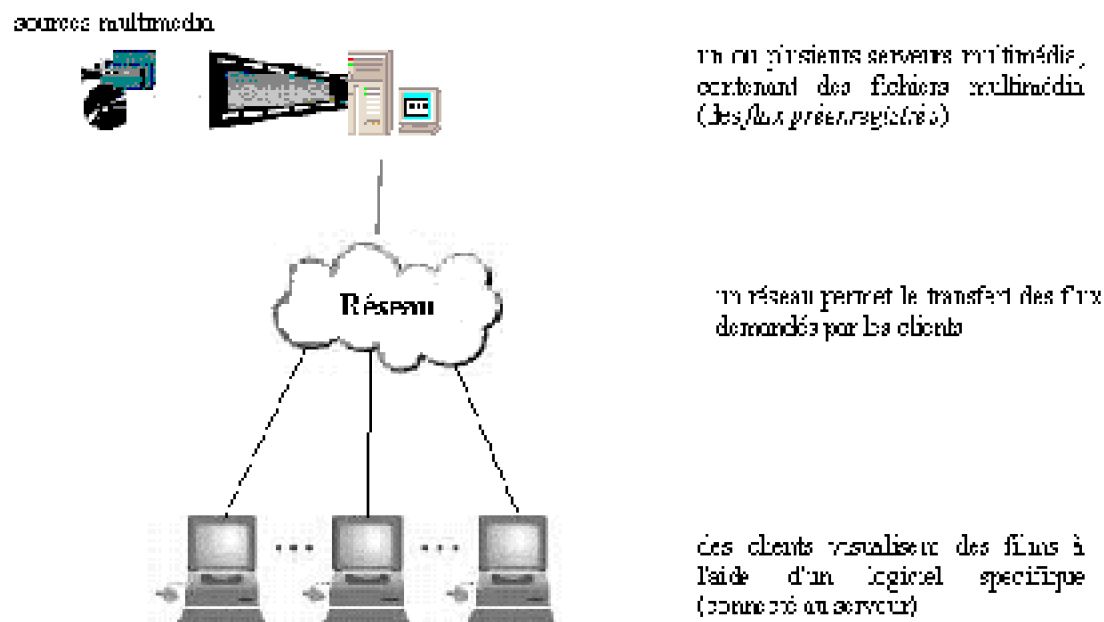


Figure 19 L'architecture très simplifiée d'un système VoD

Dans un scénario normal, chaque client potentiel d'un serveur VoD a la possibilité de naviguer dans un catalogue d'objets qui se trouve sur le serveur et de les visualiser ("jouer"). Pour jouer les objets, on utilise des logiciels spécialisés qui permettent la réalisation des opérations VCR étendues, telles que : *play*, *pause*, *stop*, *fast-forward*, *fast-backward*, *seek*. Les logiciels de lecture multimédia doivent réaliser toutes ces opérations en utilisant le réseau de manière optimale tout en gardant des temps de réaction minimaux aux commandes des utilisateurs.

Le contenu du serveur est fourni par des caméras, des magnétoscopes, des disques laser, des transmissions satellites ou d'autres sources multimédias numériques (ou analogiques via numérisation). Chaque objet peut être visualisé par plusieurs clients simultanément sans perte de qualité. Les clients peuvent visualiser les objets du serveur en utilisant un ordinateur (en plein écran ou dans une fenêtre) ou une TV connectée à une STB.

3.2.2 La communication dans un système VoD

3.2.2.1 La communication sur un ou plusieurs canaux

En détaillant l'architecture d'un système VoD, les principaux acteurs sont *le serveur* (serveur VoD) et *les clients*. Chaque client utilise un *lecteur multimédia* (en anglais : *media player*) qui permet de visualiser les séquences audiovisuelles (présentations multimédias) trouvées sur le serveur. Une présentation multimédia peut être constituée d'un ou plusieurs flux. Pour réaliser le transfert de ces flux entre le serveur et le client, on utilise une ou plusieurs connexions (canaux, voir la Figure 20).

Lors d'un transfert sur un seul canal (l'approche *uni-canal*) les relations temporelles

entre les plusieurs flux qui composent l'objet sont préservées et le nombre de paquets de contrôle est relativement petit. Mais si le débit ne peut pas être assuré, le son et l'image seront détériorés. Par exemple, un flux audio-vidéo encodé à 1.5 Mbps ne pourra pas être joué correctement sur une connexion qui ne permet qu'un débit de 1 Mbps. Même si la partie audio n'occupe que 64 Kbps de la bande passante totale, l'utilisateur va recevoir un son de mauvaise qualité.



Figure 20 Les deux approches de communication entre un serveur VoD et ses clients : uni-canal et multi-canaux

Pour éliminer ce problème, l'approche *multi-canal* permet la séparation des données audio et vidéo, en transférant un flux par canal. Dans ce cas, si le réseau ne peut pas fournir le débit souhaité, le *lecteur multimédia* peut choisir une compression plus forte (débit plus faible) pour le flux vidéo, tout en gardant un son de bonne qualité. Cette technique de transfert de flux d'une manière intelligente (en anglais : *intelligent streaming*) est utilisée dans [ms-wmp] et [rn-rp]. Dans cet exemple, la partie audio arrivera au client sur un canal (à 64 Kbps) et la partie vidéo sur un autre canal. Si la connexion ne peut fournir que 1 Mbps alors que 1.5 Mbps sont demandés, le canal audio sera prioritaire (l'oreille humaine est plus sensible que l'oeil [steinmetz96]), ayant comme effet une réduction de la bande passante sur le canal vidéo. Côté utilisateur on réceptionnera une succession d'images saccadées (des images manquantes), mais le son restera de bonne qualité.

3.2.2.2 Le streaming

En ce qui concerne la stratégie de transfert d'un objet, il existe deux stratégies de communication entre le lecteur multimédia client et le serveur multimédia :

"*download and play*" : mode dans lequel le fichier est entièrement transféré avant que le client ne puisse le visualiser (stratégie appelée également «full file transfer»)

"*media streaming*" : le mode dans lequel le lecteur multimédia client commence à jouer le fichier pendant le transfert, quelques secondes seulement après le début du téléchargement.

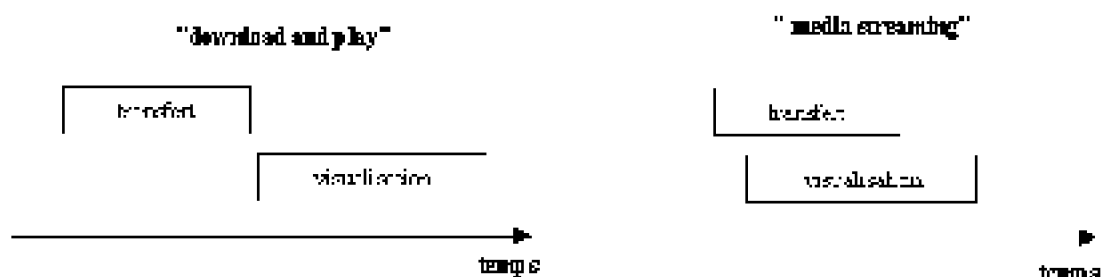


Figure 21 Les deux stratégies de transfert des données utilisées en VoD : "download and play" (le fichier est entièrement transféré avant que le client ne le visualise) et "media streaming" (le lecteur client commence à jouer le fichier pendant le transfert)

Les avantages et les inconvénients de ces deux méthodes sont présentés dans la Figure 22. En raison de ses défauts, la méthode "download and play" est de moins en moins utilisée (historiquement, elle a été, et reste encore aujourd'hui, la première méthode de transfert utilisée sur Web). Nous nous intéressons de plus près à la méthode "media streaming".

Figure 22 Etude comparative sur les deux méthodes de transfert dans un système VoD. Les facteurs étudiés sont l'utilisation du réseau et le temps de réaction aux commandes de l'utilisateur.

	utilisation du réseau	temps de réaction aux commandes d'utilisateur
download and play	mauvaise ; le fichier est complètement transféré, que l'utilisateur veuille regarder tout ou une partie du film ;	important ; pour réaliser le commandes VCR il faut que le fichier soit entièrement arrivé
media streaming	meilleure que "download and play" ; si c'est le client qui demande les données, sur le réseau n'est transférée que la quantité de données nécessaire au lecteur client (cela inclues aussi un certain tampon de données ⁹ utilisé par le lecteur avant de commencer à jouer)	faible ; le temps de réaction dépend du temps de réponse du serveur aux messages du lecteur client

3.2.2.3 Les caractéristiques du communication dans un système VoD

Le transfert des données multimédia impose une série de contraintes particulières sur les applications multimédias. On trouve ainsi (une étude complète se trouve en [hafid98]) :

9

un débit élevé de transmission des données ; le débit élevé est dû à l'immense quantité de données nécessaire pour représenter les informations multimédias. Même si la quantité de données peut être réduite en utilisant différentes techniques de compression, les débits restent élevés (500Kbps – 10Mbps par flux, donc un grand défi pour les réseaux de nos jours),

des contraintes temporelles : le système doit être capable de respecter les relations temporelles dans un flux (communication uni-canal) ou entre plusieurs flux (communication multi-canaux), pour préserver la continuité d'une présentation multimédia,

des garanties pour les services : les réseaux et les composants du système doivent être capable de garantir la bande passante et les contraintes temporelles demandées.

3.2.2.4 Protocoles de communication utilisés pour la VoD

Les principaux protocoles de domaine public, les plus utilisés en VoD, sont TCP et UDP (niveau transport [tanenbaum97]), HTTP et RTSP (niveau application, eux-mêmes implémenté sur TCP et UDP). Chaque protocole configure les données en paquets, en ajoutant pour chaque paquet un en-tête identifiant son contenu.

TCP ([tcp]) est un protocole conçu pour le transfert fiable des données alphanumériques. Il arrête le transfert quand des données sont perdues pour les corriger, afin de garantir une transmission séquentielle sans erreur. Ses inconvénients sont les retards qu'il peut accumuler en essayant de corriger les données manquantes, mais un grand avantage réside dans la fiabilité de la communication offerte.

Le protocole UDP ([udp]) est un protocole qui ne garantit pas la qualité du transfert, les données étant de préférence perdues que retransmises pour s'adapter à la bande passante utilisable.

Le protocole de transfert hypertexte (HTTP- Hyper Text Transfer Protocol [http]) est le protocole standard sur le Web. Il s'agit d'un protocole du niveau "application" (modèle OSI, [tanenbaum97]), au dessus de TCP, qui offre un standard de communication entre les serveurs et les clients Web. Il n'est pas adapté au streaming audio - vidéo (il possède les mêmes défauts que TCP).

Le protocole de streaming en temps réel (RTSP - Real Time Streaming Protocol – [RTSP]) a été conçu comme une extension de HTTP, pour la transmission de données ayant des contraintes temps réel. Il est maintenant l'un des protocoles les plus utilisés en VoD (voir [rn-rs]). Cependant les infrastructures réseau actuelles ne permettent pas toutes son utilisation (certains pare-feu ne laissent pas passer les données RTSP).

D'autres protocoles sont utilisés en solutions VoD propriétaires (comme le protocole mms dans [ms-netshow]) mais leur description n'est pas disponible et ils ont les mêmes difficultés que RTSP pour passer à travers certains réseaux.

3.2.3 Le serveur VoD

3.2.3.1 Les caractéristiques d'un serveur VoD

Le serveur VoD est le coeur d'un système vidéo à la demande. Ce lui qui partage les objets multimédias aux clients. Tenant compte de la taille de ces objets, le système de stockage du serveur doit faire face à des challenges importants. Chaque nouveau client désirant accéder aux objets disponibles est accepté ou non par un *ordonnanceur* (en anglais: *scheduler*). L'ordonnanceur du système vérifie si le serveur peut satisfaire la demande d'un nouveau client sans pénaliser les autres clients déjà connectés. La décision d'accepter un nouveau client est prise en utilisant une *politique de contrôle d'admission*. Plusieurs politiques sont utilisées par les serveurs VoD actuels. Nous en proposons une qui est adaptée au protocole HTTP (Chapitre 6).

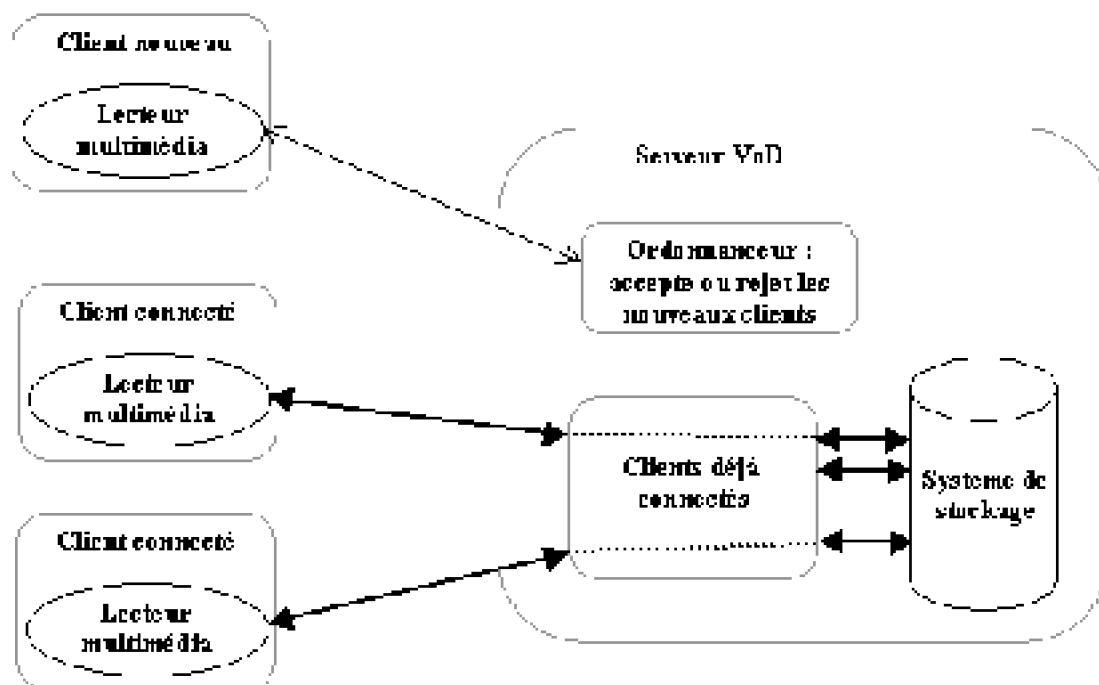


Figure 23 Vision d'ensemble sur les composants d'un serveur VoD

3.2.3.2 Classification par rapport aux interactions client-serveur : serveurs "push" et serveurs "pull"

Par rapport aux interactions client-serveur VoD, on trouve deux grandes catégories de serveurs VoD : des serveurs "push" et des serveurs "pull" ([rao96]).

Dans le paradigme "push", la politique de transfert des données multimédias est laissée à la charge du serveur VoD. Le client ouvre une connexion avec le serveur, envoie une requête pour un objet et le serveur envoie cet objet de façon continue, en gardant les contraintes temporelles (i.e. le débit de décompression) de cet objet (Figure 24). Pour chaque objet le serveur est obligé de connaître toutes ces caractéristiques

(format de compression, débit etc.) pour envoyer chaque paquet à temps.

Dans le cas des objets multimédias codés à débit variable (VBR¹⁰), le serveur doit réaliser un semi-décodage avant le transfert, pour connaître la quantité de données à transférer à chaque instant¹¹. Cette décompression (la décomposition "push") prend beaucoup de ressources, en limitant les performances d'un tel serveur. Une solution réside dans l'utilisation de *formats conteneur*. Les formats conteneur contiennent des informations supplémentaires pour chaque bloc des données. C'est le cas des formats AVI, ASF ([ms-wm]) ou QuickTime ([apple]), qui contiennent les flux compressés avec MPEG ou un autre standard, et avec un en-tête pour chaque bloc. L'utilisation de cet en-tête évite au serveur de décompresser chaque fois l'objet à transmettre.

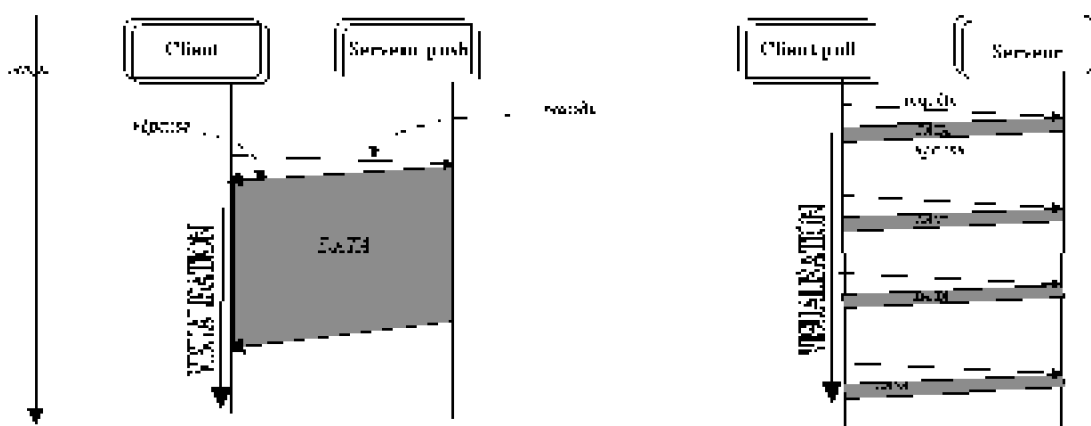


Figure 24 Les deux paradigmes d'interaction client - serveur en VoD: "push" et "pull"

Le paradigme "pull" laisse le client gérer lui-même le transfert des données (Figure 24). Ainsi, chaque fois que le client a besoin de nouvelles données il demande un bloc au serveur. Cela rend plus difficile la gestion des ressources du côté du serveur, mais il existe un grand avantage pour le streaming des objets encodés à débit variable. En effet, le client réalise le décodage et sait parfaitement combien de données il doit demander à chaque instant. Le serveur est donc déchargé de cette planification (à quel instant il faut envoyer un certain bloc). En revanche, un inconvénient de ce paradigme réside dans l'augmentation du trafic sur le réseau (chaque sous-requête pour un bloc ajoute des informations à transférer).

Les deux approches (*pull* ou *push*) ont leurs avantages et inconvénients, mais actuellement la quasi-totalité des serveurs VoD travaille en mode push (parmi d'autres NetShow et RealServer, les plus répandus). Dans cette thèse nous montrerons, entre autres choses, qu'un serveur *pull* peut être utilisé avec les mêmes performances qu'un serveur *push*, et offrir plus de satisfaction aux utilisateurs en ce qui concerne le temps de réponse aux commandes.

¹⁰ VBR : acronyme pour Variable Bit Rate ; qualité de service à débit variable ; un des types de débit proposé par l'ATM Forum qui s'adapte à la demande de l'utilisateur.

¹¹ Les films sur DVD sont codés à un débit très variable (des séquences codées à 4 Mbps peuvent alterner avec des séquences à 12 Mbps)

3.2.3.3 Classification par rapport aux protocoles réseau utilisés

Par rapport aux protocoles réseau utilisés, il existe deux grandes catégories de serveurs qui peuvent fournir de la VoD :

serveurs Web : ce sont des serveurs qui travaillent avec le protocole HTTP,

serveurs VoD spécialisés : des serveurs qui travaillent avec d'autres protocoles (basés sur UDP, TCP ou IPMulticast), comme par exemple RTSP.

L'utilisation du protocole HTTP présente l'avantage de permettre à une grande variété de serveurs Web existants d'être utilisés. De plus, parce que les "firewalls" Internet filtrent les données en utilisant le numéro du port TCP/IP, de nombreux utilisateurs ne peuvent pas recevoir les données envoyées par d'autres logiciels que les serveurs Web (utilisant le protocole HTTP). Comme mécanisme de travail, un serveur Web, en réponse à une requête sur un objet, essaie d'envoyer l'objet le plus vite possible (stratégie *push*, Figure 25). Cette stratégie a un inconvénient : une utilisation inefficace de la bande passante. Cependant, plusieurs lecteurs multimédia utilisent le protocole HTTP, malgré ses défauts, comme alternative à d'autres protocoles ([ms-wmp], [rs-rp], [xing], [vivo], [mpegvtv]), mais ils ne donnent pas accès à toutes les opérations VCR (*seek*, par exemple).

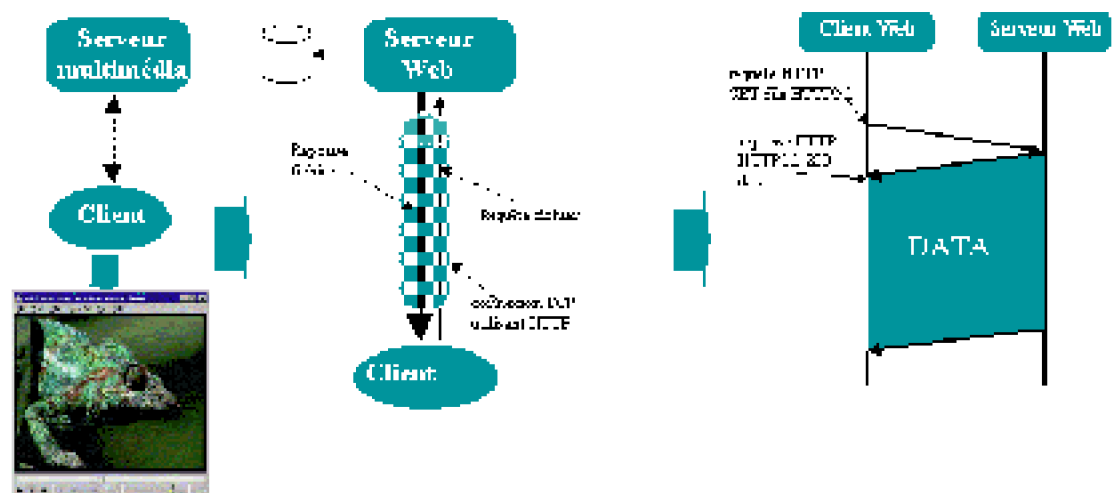


Figure 25 "Streaming" avec un serveur Web (HTTP)

Les serveurs spécialisés utilisent quant à eux de nouveaux protocoles pour délivrer le contenu multimédia, en améliorant la performance du «streaming». Un exemple d'un tel protocole est RTSP ([rtsp]) qui contrôle le transfert d'un ou plusieurs flux synchronisés grâce à une connexion TCP séparée de la connexion des données (Figure 26). Un inconvénient de cette approche réside dans le fait que les nouveaux protocoles ne sont pas reconnus par tous les lecteurs multimédias et les pare-feu existants. De plus, les serveurs spécialisés demandent beaucoup plus de ressources que les serveurs Web, sont chers ([approach99]), et peuvent nécessiter la modification de l'infrastructure réseau

existante.

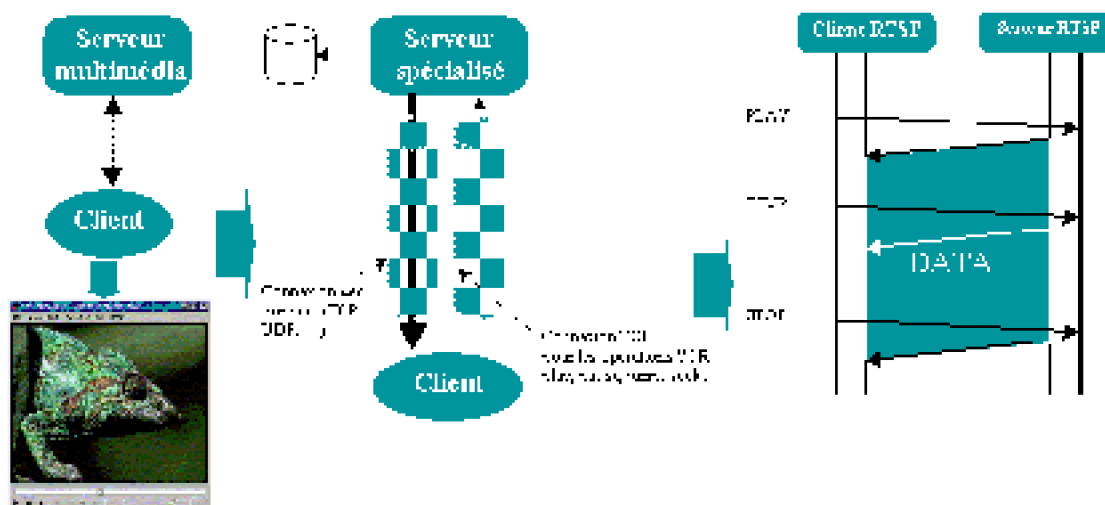


Figure 26 "Streaming" avec un serveur spécialisé

Parce que les deux catégories de serveurs ont des inconvénients et des désavantages, nous essayons de trouver les bonnes stratégies HTTP pour utiliser un serveur Web avec des performances comparables à celles des serveurs VoD spécialisés. Plus exactement, nous utilisons le serveur Web avec une stratégie *pull* et nous montrons qu'une telle solution arrive à atteindre les performances maximales d'un système physique (Chapitre 5).

Avant cela, nous étudierons dans la section suivante différents serveurs VoD existants, afin d'en déduire les facteurs de performance.

3.2.3.4 Serveurs VoD existants

3.2.3.4.1 RealSystem G2

Le serveur G2 de Real Networks ([rn-rs]) permet la VoD sur Internet ou Intranet en utilisant RTSP comme protocole de communication et HTTP comme alternative pour passer les pare-feu. Il fonctionne sur différents systèmes d'exploitation, en utilisant des systèmes de fichiers standards, mais le format de stockage des données est un format propriétaire. Par rapport aux autres serveurs VoD, G2 a des bonnes performances pour Internet avec des flux bas-débit, ceux-ci étant les plus utilisés sur le Web. Cependant il n'est pas utilisable sur des réseaux Intranet pour faire de la VoD haut-débit (faibles performances du point de vue visualisation et temps de réaction aux commandes VCR).

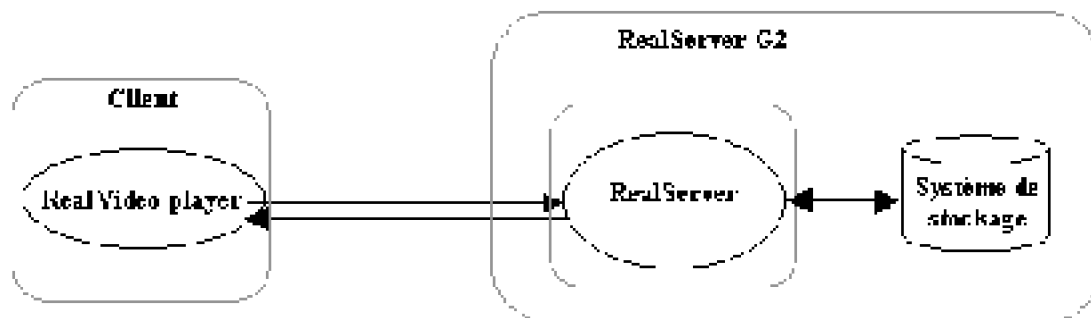


Figure 27 L'architecture de G2

3.2.3.4.2 NetShow

Le serveur VoD de Microsoft (après l'acquisition de Vxtreme en août 1997) est basé sur une architecture distribuée, des systèmes de stockage haute performance et un système de fichiers propriétaire. La communication est réalisée avec un protocole propriétaire Microsoft (MMS), mais HTTP et RTSP sont utilisés comme alternatives.

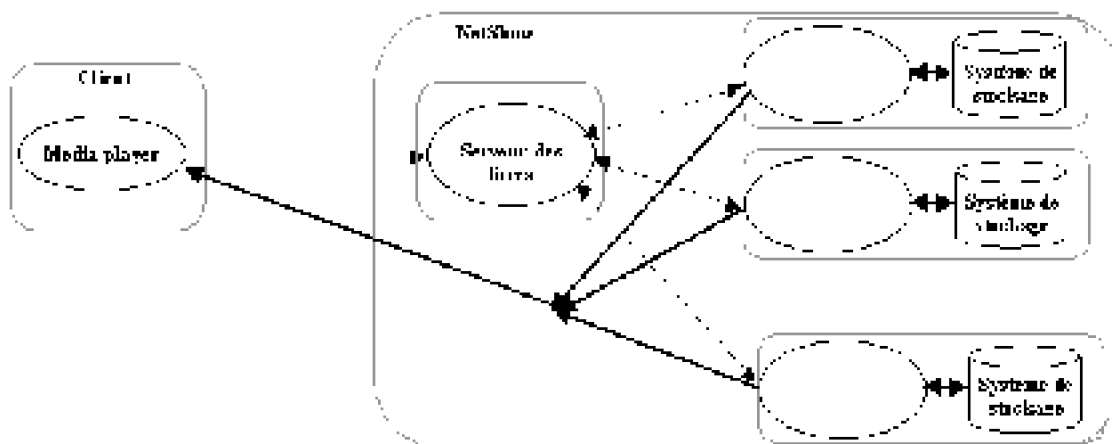


Figure 28 L'architecture de NetShow

Un système NetShow contient un serveur de titres connecté à plusieurs serveurs de contenu. Chaque serveur de contenu a son propre système de stockage, un objet multimédia étant distribué sur tous les serveurs de contenu. Un client va recevoir le contenu d'un objet avec un protocole *multipoint-to-point*¹². L'avantage d'une telle architecture est le suivant : il n'y a pas de serveurs de contenu plus chargés que d'autres, toutes les opérations sont uniformément distribuées sur les serveurs de contenu.

Les systèmes matériels compatibles avec NetShow sont très coûteux, et les performances en VoD haute-qualité les indiquent comme une solution Intranet. Des versions plus récentes de NetShow, moins coûteuses, avec de bonnes performances sur

¹² *multipoint-to-point* : stratégie de communication selon laquelle plusieurs hôtes envoient des blocs des données appartenant à un seul fichier vers un seul destinataire avec le but de reconstituer le fichier à la destination ; d'autres sémantiques multipoint et multicast inclues les connexions *IP-multicast* et *point-to-multipoint*.

Internet pour des flux bas-débit, sont intégrées dans Windows Media Services ([ms-wms]).

Au niveau performance, une solution NetShow capable d'envoyer 667 flux à 2 Mbps sur des réseaux ATM a besoin de 1 serveur de titres, 14 serveurs de contenu avec 6 disques SCSI par serveur, 10 flux par disque ([netshow- price]). La solution serveur Web comme serveur VoD que nous proposons sera comparée à ces performances (Chapitre6).

3.2.3.4.3 AMS

La société CSTI, au sein de laquelle nous avons réalisé cette thèse, a conçu un système distribué à hautes performances apte à supporter des applications réparties temps réel critiques, appelé ANTARA¹³ ([antara]). Avec cette technologie comme base, on a réalisé une version plus spécifique nommée AMS (Antara Multimedia Server) et destinée aux applications multimédia ([ams]). Le serveur AMS est conçu pour les applications vidéo à la demande (VoD), diffusion différée à la demande (en anglais : *delayed TV*) et serveur de stockage pour les chaînes de télévision numériques.

L'architecture de ce système est totalement distribuée. Le serveur est schématiquement composé d'un processus de contrôle (AMSD) qui dirige plusieurs processus responsables du transfert d'objets multimédia (MOT – acronyme pour Multimedia Object Transfer). Tous ces processus peuvent se trouver sur le même ordinateur ou sur des ordinateurs différents.

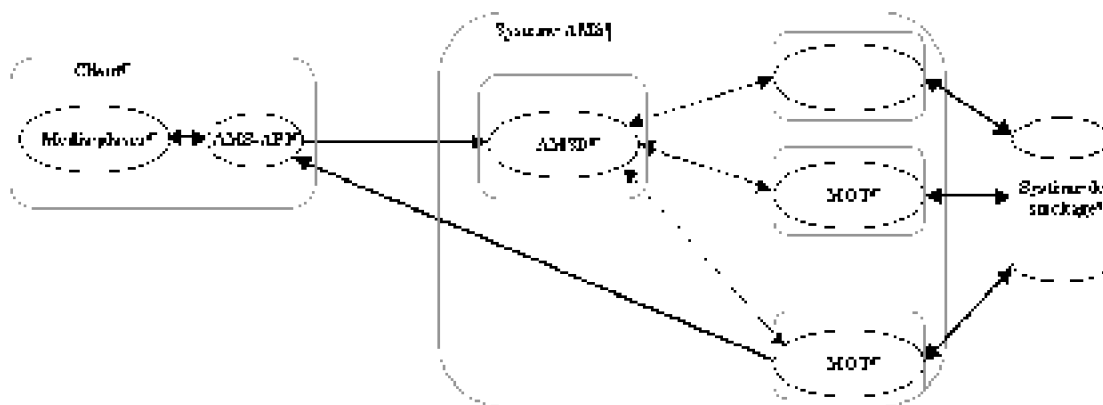


Figure 29 Architecture d'AMS

Un logiciel client VoD se connecte au AMSD à travers une API¹⁴ spécifique (AMS-API). L'AMSD dirige la requête vers le MOT le moins utilisé, qui s'occupera du transfert des données vers le client. Le protocole de communication est basé sur TCP, UDP et IPMulticast, mais ses spécifications sont propriétaire CSTI. L'AMS est une solution optimisée pour des flux haut-débit (1.5 – 8 Mbps) sur des réseaux locaux

¹³ ANTARA – acronyme pour Advanced Network Technology Applied to Real-time Applications

¹⁴ API: acronyme pour Application Programming Interface

haute-performance (Gigabit-Ethernet, Fast-Ethernet, ATM).

Au niveau performance, les tests que nous avons faits avec AMS sont présentés dans le Chapitre 4.

3.2.4 Le lecteur VoD

Dans un système VoD, les utilisateurs ont un contact direct avec les logiciels de lecture multimédia (en anglais: *media players*). Un lecteur multimédia (ou lecteur VoD) est un logiciel qui s'occupe de tous les aspects concernant la connexion avec le serveur VoD, le transfert et le décodage des données, l'affichage de ces données sur l'écran du client.

3.2.4.1 L'architecture logicielle d'un lecteur VoD

Actuellement il existe un grand nombre de lecteurs permettant la VoD en local ou en utilisant des serveurs VoD : [rn-rv], [quicktime], [ms-wmp], [siren], [winamp], [sonique], [midisoft], [mps], [iqms] etc. Leurs interfaces utilisateur ont changé assez souvent (généralement pour des raisons commerciales), mais les principes de fonctionnement sont relativement stables en ce qui concerne le flux des données. L'architecture qui se retrouve sur la grande majorité de lecteurs est celle basée sur des filtres (ou plug-ins en anglais) : le lecteur contient différents composants logiciels (filtres qui composent un graphe, voir [activex] et [directx]) avec des interfaces communes, utilisées pour le transfert, le décodage et même l'affichage des données (voir la Figure 30).

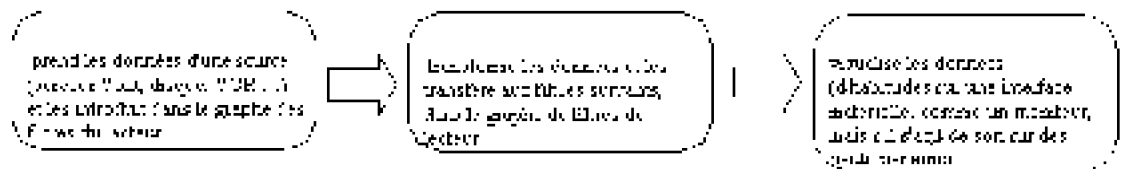


Figure 30 Le graphe des filtres d'un lecteur multimédia

Chaque lecteur possède un certain nombre de filtres dans sa distribution standard, mais il est possible d'en ajouter pour accroître ses fonctionnalités. Par exemple, on peut remplacer le filtre source qui implémente le protocole HTTP pour la communication avec le serveur par une autre implémentation. Le mécanisme de travail d'un lecteur avec cette stratégie est décrit par l'algorithme suivant :

1. l'utilisateur ouvre un objet multimédia avec son adresse (par ex. : `rtsp://server/object.mpg`)
2. le lecteur cherche le filtre qui implémente le protocole `rtsp` et le démarre
3. le lecteur commence à transférer les données par l'intermédiaire du filtre `rtsp` ; il construit le graphe des filtres de transformation et de visualisation nécessaires en fonction du format de l'objet

4.

le lecteur commence à jouer l'objet en utilisant le graphe des filtres construit

Un exemple de graphe utilisé par [ms-wmp] pour jouer un objet MPEG-1 est présenté dans la Figure 31 (l'objet vidéo se trouve sur un serveur RTSP).

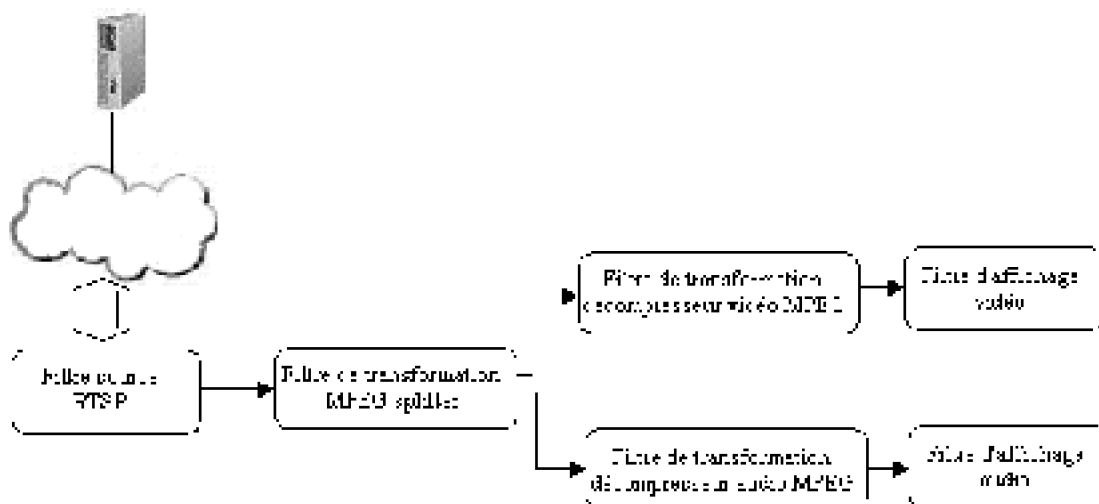


Figure 31 Graphe des filtres pour jouer un objet MPEG-1 avec le protocole RTSP

La possibilité d'ajouter des filtres dans un lecteur multimédia est très importante pour les applications pratiques de cette thèse. Cela nous a permis le remplacement du protocole HTTP existant dans un lecteur avec une autre implémentation de HTTP, pour obtenir plus de performance en utilisant un serveur Web comme serveur VoD (Chapitre 5).

3.2.4.2 Le flux de données

Les données arrivent au client en paquets, chaque paquet contenant un certain nombre de trames (en anglais: frames) à décompresser et visualiser. Dans certains cas, pour décompresser une trame on a besoin de plusieurs paquets voisins (ceci est une caractéristique des formats de compression MPEG, avec des trames P, I, B¹⁵). Pour cela, et pour compenser des fluctuations dans le débit du réseau, un lecteur utilise une mémoire tampon (en anglais : *buffer*) pour recevoir en avance les données. La taille de ce buffer est mesurée en temps de visualisation des ces données (généralement, la taille du buffer est de moins de 5 secondes de contenu).

Le lecteur commence à jouer l'objet uniquement lorsque le buffer est rempli. Si le serveur fonctionne en mode push, le client va attendre un certain temps avant que l'objet ne soit joué. Le même phénomène se reproduit pour chaque opération Seek (le buffer est vidé et re-rempli). Pour qu'un client soit satisfait de son service VoD, le temps de réponse a ses commandes (play, seek) doit être le plus petit possible ([nwsu96]). En conséquence, pour garantir une bonne satisfaction du client, le serveur doit augmenter le

¹⁵ frames P, I, B : types de cadres utilisés dans le format de compression MPEG

trafic lorsque ces événements surviennent. Idéalement, le débit des données reçues côté client pour un flux multimédia encodé à débit constant va avoir un profil comme celui de la Figure 32.

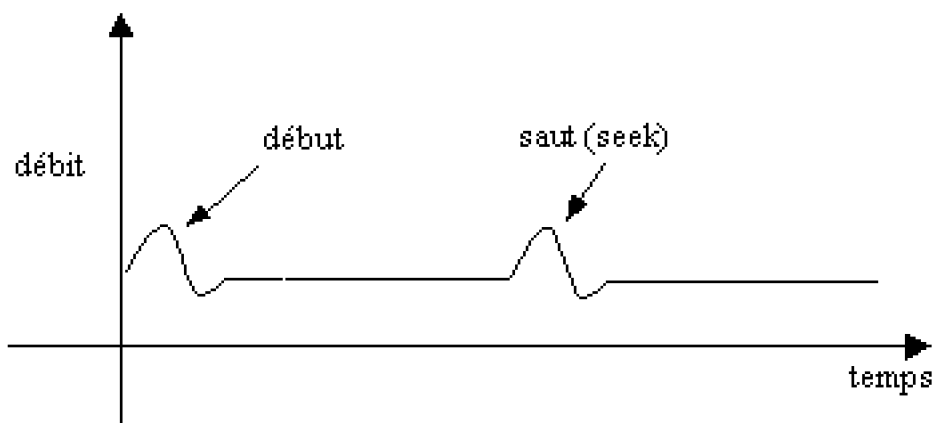


Figure 32 Le débit sur réseau d'un flux multimédia encodé à débit constant.

Les opérations VCR (seek, jump) nécessitent un nouveau remplissage du buffer. Ce transfert se fait le plus vite possible afin d'offrir une bonne réactivité à ces commandes et engendre donc une fluctuation importante dans le débit.

3.3 Facteurs de performance pour un système VoD

Les clients d'un système VoD sont influencés par deux facteurs concernant la qualité de ce service : la qualité de l'image et le temps de réponse aux commandes VCR entendues. L'image doit avoir les mêmes caractéristiques que lors de l'encodage (même nombre d'images par seconde, même nombre de couleurs, bonne synchronisation avec la partie audio etc.). Le temps de réponse représente la période entre la demande d'une fonctionnalité (par exemple Seek) et son exécution effective. Il ne doit pas prendre des valeurs importantes afin de satisfaire l'utilisateur (des bons temps de réponse se situent autour de 0.5s).

Pour arriver à réaliser ces contraintes, un grand nombre de facteurs doivent être pris en compte. Après une présentation des critères de performance d'un système VoD, nous présenterons les facteurs de performance les plus importants : le système de stockage, le réseau et la puissance de calcul.

3.3.1 Les critères de performance d'un système VoD

Dans le cas des serveurs Web nous avons à notre disposition une grande variété des méthodes de tests (benchmarks) qui nous permettent de calibrer un serveur en fonction

des différents facteurs (généralement le nombre de requêtes par seconde et le débit, voir le chapitre précédent). Dans le domaine des serveurs VoD nous ne disposons pas d'une telle stratégie de mesure, mais nous supposons que les critères de performance pour un système VoD sont le nombre de clients satisfaits du service fourni et le débit total du serveur vers les clients.

Par la suite nous détaillerons une méthode de mesure des performances pour un système VoD que nous avons proposée (Chapitre 4). Pour l'instant, nous supposons que le système de stockage, le réseau et la puissance de calcul doivent être les plus performants possibles.

3.3.2 Le système de stockage

Le système de stockage est généralement constitué de plusieurs disques durs. Les taux de transfert d'un bon disque dur se situent actuellement autour de 80 Mbps ([storagereview])¹⁶. Pour éviter qu'un seul disque ne devienne un goulot d'étranglement (ce qui est le cas lorsqu'un nombre important de clients accèdent au même fichier), les disques sont organisés en tableau de disques (RAID¹⁷) de manière à ce que les requêtes soit réparties uniformément entre tous les disques qui composent ce tableau. Il existe plusieurs types des systèmes d'organisation des disques RAID de base (nommées *niveaux*) : RAID0, RAID1, RAID5 et des systèmes RAID hybrides : RAID10 (RAID1+RAID0), RAID50 (RAID5+RAID0).

3.3.2.1 RAID0

Les données sont organisées en blocs et ceux-ci sont répartis uniformément, dans un ordre fixe, entre les disques du tableau (Figure 33) ; les opérations lecture/écriture sont réalisées de manière indépendante et simultanée entre plusieurs disques à la fois : on obtient ainsi des performances plus élevées. Un tel système de stockage n'est pas tolérant aux pannes : si un disque tombe en panne, tous les informations sont inutilisables.

Si on note :

$bw(\text{disque})$ = la bande passante d'un disque ($bw = \text{bandwidth}$)

$bw(\text{RAID}x)$ = la bande passante d'un système de stockage RAID (qui dépend des caractéristiques de ses disques, qui n'ont pas obligatoirement les mêmes performances)

les performances de transfert d'un système RAID0 sont alors :

¹⁶ Les facteurs les plus déterminants pour la performance d'un disque dur sont le débit soutenu et le nombre de rotation par minute.

¹⁷ RAID : acronyme pour Redundant Array of Independent Disks

$$bw(RAID0) = n * \min_{disque \in RAID0} (bw(disque))$$

où $n = card(RAID0)$ c'est à dire le nombre de disques disponibles sur le système RAID0

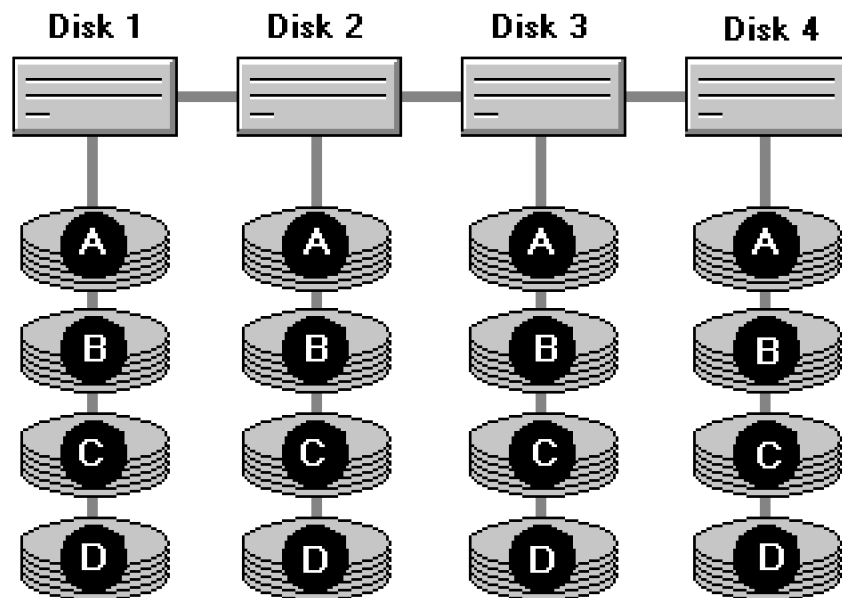


Figure 33 Répartition des blocs sur des disques organisés en RAID0 (les blocs proviennent de quatre fichiers : A, B, C et D)

3.3.2.2 RAID1

Ce niveau est aussi connu sous le nom de "disque miroir" ; pour chaque disque on a un deuxième disque qui est la copie fidèle du premier. Toutes les données écrites sur le premier disque se retrouvent sur le deuxième également. La performance pour la *lecture* est améliorée (la bande passante des deux disques additionnés) et les disques bénéficient de plus d'une tolérance aux pannes (si un disque tombe en panne l'autre contient les mêmes informations, mais dans ce cas nous n'avons plus les mêmes performances en lecture).

$$bw_{read}(RAID1) = 2 * \min_{disque \in RAID1} (bw_{read}(disque))$$

$$bw_{write}(RAID1) = \min_{disque \in RAID1} (bw_{write}(disque))$$

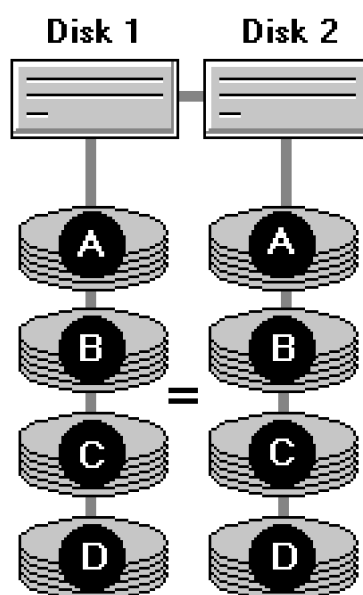


Figure 34 La répartition des blocs sur des disques organisés en RAID1 (les blocs de quatre fichiers : A, B, C et D)

3.3.2.3 RAID5

RAID5 est similaire à RAID0, mais il fournit un autre type de tolérance aux pannes (en utilisant un disque de parité) et permet l'utilisation plus efficace de l'espace de stockage (avec RAID1 on n'utilise que 50% de l'espace disponible). L'information redondante pour permettre la reconstruction d'un disque tombé en panne est distribuée comme indiqué en Figure 35, et le pourcentage de l'espace utile perdu est $1/n$ (où n est le nombre des disques qui composent le système RAID5).

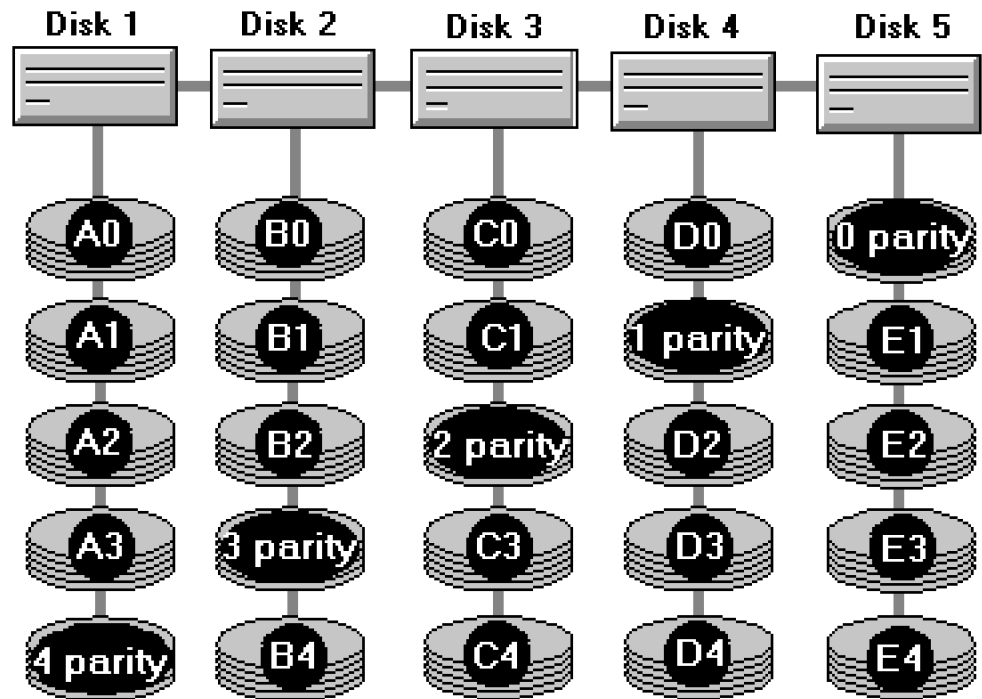


Figure 35 La répartition des blocs sur des disques organisés en RAID5 (les blocs de quatre fichiers : 0 à 3)

$$bw_{write}(RAID5) = (n - 1) * \min_{disque \in RAID5} (bw_{write}(disque))$$

$$bw_{read}(RAID5) = n * \min_{disque \in RAID5} (bw_{read}(disque))$$

En utilisant un système de stockage RAID on arrive à des performances importantes en augmentant le nombre des disques durs du système. Cependant l'utilisation de ces performances n'est pas toujours facile. Les disques sont liés à un contrôleur, qui possède également certaines limitations (Figure 36).

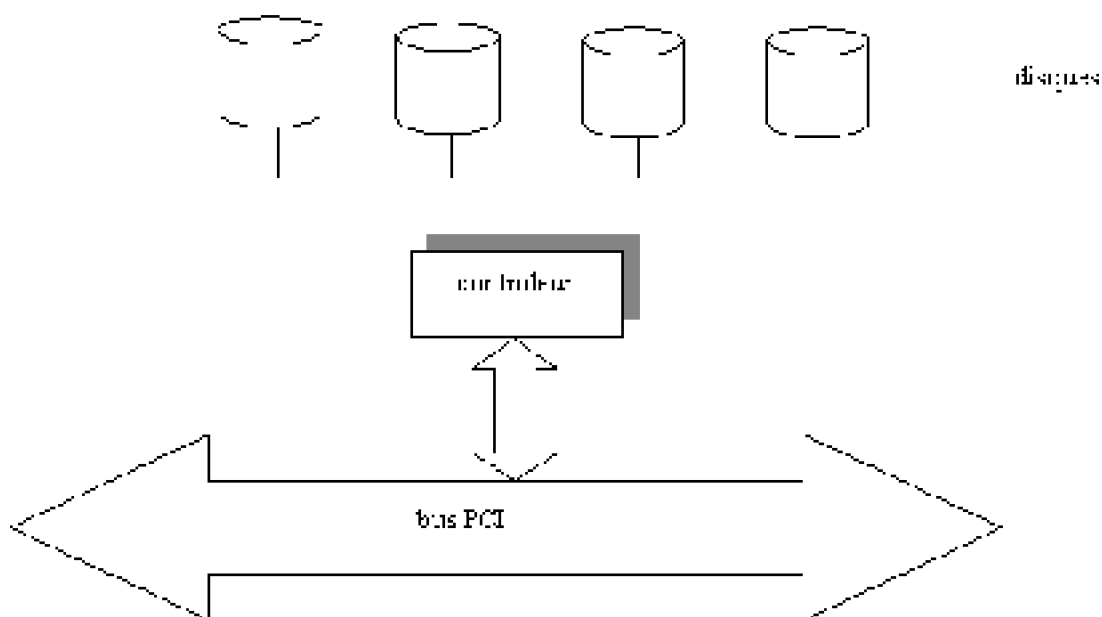


Figure 36 L'architecture d'un système de stockage : les disques sont liés à un contrôleur, qui se trouve sur la carte mère du serveur

Les performances théoriques actuelles d'un contrôleur de disques sont présentées dans la Figure 37, mais elles ne sont en réalité jamais atteintes. Le débit permis sur le bus est généralement supérieur aux performances du contrôleur de disques (un bus 64bits à une fréquence de 33 Mhz permet des taux de transfert de $33 \text{ Mhz} * 64 \text{ bits} = 2.1 \text{ Gbps}$), mais il doit également faire fonctionner d'autres périphériques (comme par exemple une carte réseau). Pour ces raisons, le bus peut limiter lui-même les performances d'un système de stockage.

Figure 37 Les interfaces et leur performances de transfert pour les systèmes de stockage actuelles

Interfaces pour le système de stockage	Débit
IDE/ATA (Integrated Drive Electronics / AT Attachment – [ata])	33MB = 264 Mbps
SCSI-1 (Small Computer Systems Interface – [scsi], [storagereview])	40 MB = 320 Mbps
SCSI-2 ([scsi], [storagereview])	80 MB = 640 Mbps
SCSI-3 ([scsi], [storagereview])	160 MB = 1280 Mbps
SSA ([ibm-storage])	100 MB = 800 Mbps
SAN (Storage Area Network - [ibm-storage])	200 MB = 1.6 Gbps

Les disques durs faisant partie de système de stockage avancent d'une manière exponentielle, nous permettant d'exploiter au maximum les interfaces du système de stockage (Figure 38, [ibm-storage]).

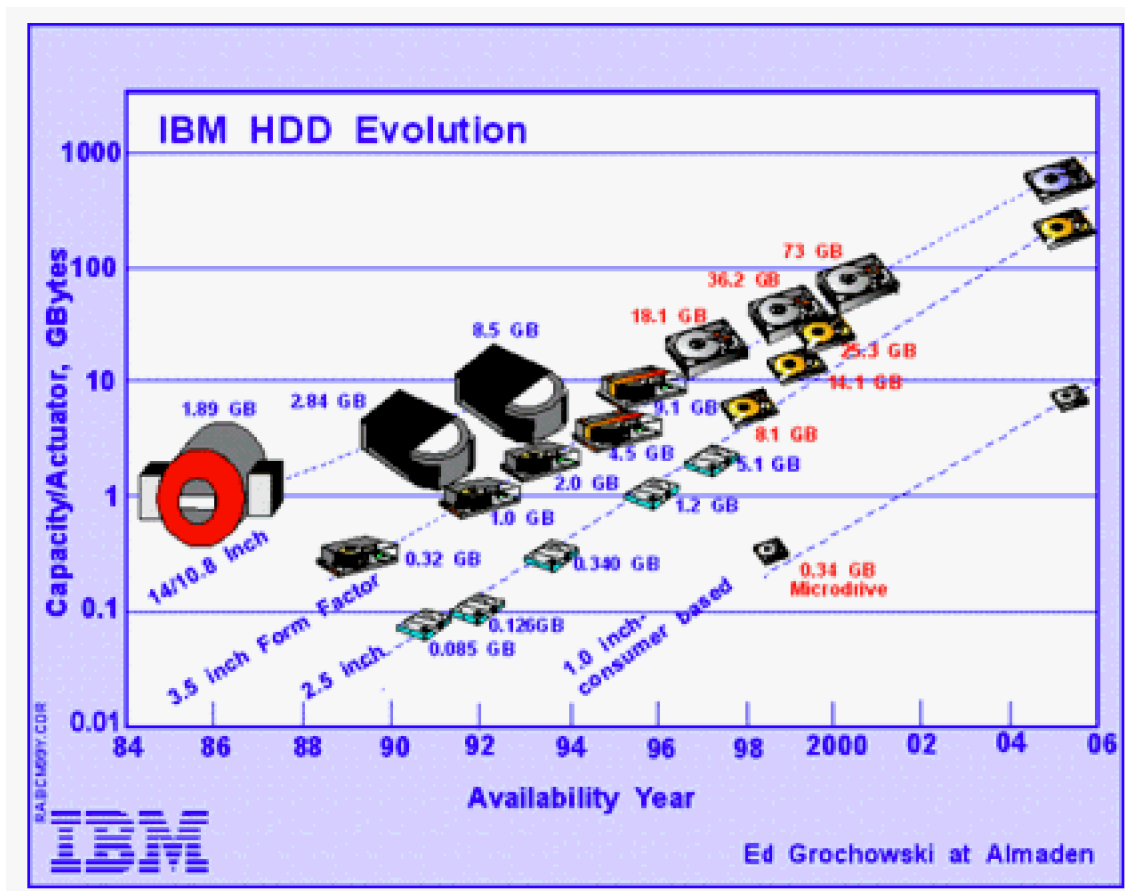


Figure 38 L'évolution des performances des disques durs IBM

3.3.3 Le réseau

En supposant que le système de stockage a des performances satisfaisantes, il faut que le réseau permette la distribution de ces informations aux clients dans de bonnes conditions. En ce qui concerne le débit il y a deux scénarios :

les clients se connectent au serveur via l'Internet : dans ce cas on ne peut pas avoir beaucoup d'information sur une connexion avec un client ; le débit est généralement faible, irrégulier et sans garanties particulières pour la qualité de service ; les objets envoyés aux clients sont généralement compressés avec le standard MPEG-4 à faible débit, et un serveur peut gérer un grand nombre de connexions (si les disques et la connexion réseau le permettent)

les clients se connectent au serveur via Intranet (réseau local) : dans ce cas on dispose d'un réseau haut débit, qui dans certains cas peut assurer la qualité du service (exemple: garantir le débit sur une connexion ATM [atm-forum] ou IPV6 [ipv6]) ; le nombre de clients est généralement plus petit et chaque client peut visualiser des objets à haut débit (MPEG1, MPEG2 etc.) ;

Les cartes réseau actuelles permettent d'obtenir des performances de même ordre de grandeur que les systèmes de stockage RAID (Figure 39). Tous les tests faits dans le cadre de cette thèse ont comme base des réseaux locaux de type Ethernet, Fast-Ethernet, GigabitEthernet ou ATM. ADSL¹⁸ est le premier type de connexion Internet haut-débit auquel nous avons eu accès à CSTI, mais à un débit de 500 kbps, peu adapté aux objets multimédias de débit élevé.

Figure 39 Les débits des plusieurs types des réseaux hauts - débit

Type de réseau	Débit
ADSL	0.5-9 Mbps selon la distance et l'opérateur et dans un seul sens (l'autre sens: 16-640 Kbps) ([adsl])
Ethernet	10 Mbps ([ethernet])
Fast-Ethernet	100 Mbps ([ethernet])
GigabitEthernet	1000 Mbps ([ethernet])
ATM	34Mbps, 155 Mbps, 622 Mbps ([atm-forum])

3.3.4 Autres facteurs

Le système de stockage et le réseau ne sont pas les seuls facteurs déterminants pour les performances d'un système VoD. D'autres facteurs peuvent avoir une influence importante, comme:

- la puissance de calcul (fréquence processeur, type processeur, mémoire cache etc.)

- la mémoire vive

- le bus de transfert des données (PCI sur 32 ou 64 bits)

Chacun de ces facteurs est important, mais si les composants physiques du système sont bien choisis, les goulots d'étranglement se situent autour du système de stockage, réseau et de la puissance de calcul.

3.4 Système Audio à la Demande.

¹⁸ ADSL : acronyme pour Asymmetric Digital Subscriber Line, une nouvelle technologie permettant d'envoyer plus de données sur les lignes téléphoniques existantes. ADSL reçoit les données à un débit situé entre 0.5 et 9 Mbps (*downstream rate*), et envoie à un débit 16 à 640 Kbps (*upstream rate*).

Un cas particulier de système multimédia à la demande sont les systèmes où les objets stockés sur le serveur sont des objets audio, donnant lieu à des systèmes appelées "audio à la demande" (AoD). Même si les débits audio sont moins importantes que les débits d'autres objets contenant de la vidéo (voir Figure 17), nous considérons que ces systèmes seront assimilés à des systèmes de vidéo à la demande.

3.5 Conclusion

Dans ce chapitre introductif nous avons présenté les composants et l'architecture générale d'un système vidéo à la demande. D'abord ont été passées en revue les caractéristiques des objets multimédias : format, taille importante, débit élevé, méthodes de compression, synchronisation en temps. Les serveurs et les clients ont été décrits, avec les diverses stratégies de communication entre eux.

Enfin, on a détaillé les principaux facteurs physiques influençant les performances d'un serveur VoD, tels que le système de stockage, l'architecture interne et le réseau.

Ce chapitre nous permet d'avoir une vision d'ensemble sur le contexte multimédia où se situent les travaux de cette thèse. Dans les prochains chapitres nous présentons des travaux originaux concernant un système VoD construit autour d'un serveur Web, mais aussi une méthodologie plus générale de mesure des performances d'un système VoD.

Chapitre 4. Méthodologie pour la mesure des performances d'un serveur VoD

4.1 La mesure des performances d'un système VoD

Quand un client demande une page Web, il souhaite que la page arrive vite et sans erreur. En conséquence les outils de mesure de performance (*benchmarking*) pour les serveurs Web s'intéressent à la mesure de deux facteurs : *le nombre de requêtes susceptible d'être servis chaque seconde* et *le débit* du serveur (voir le chapitre2). Contrairement aux clients Web, un client qui reçoit un flux multimédia réagit à plusieurs variables : la qualité de l'image, le nombre d'images par seconde, la qualité du son, la synchronisation audio-vidéo etc. Un outil mesurant la performance d'un serveur VoD devra donc calculer *le nombre de clients satisfaits simultanément* servis par le serveur et *le débit* maximal de ce serveur.

Dans cette section nous définissons la *qualité de la perception*, un facteur qui mesure le degré de satisfaction d'un client recevant un flux multimédia. L'utilisation de ce facteur pour la mesure des performances d'un système VoD sera mise en évidence, avec une méthodologie d'utilisation spécifique.

4.1.1 La qualité de la perception (Quality of Perception - QoP)

Dans un environnement multimédia distribué la qualité d'une présentation multimédia est déterminée par la perception de l'utilisateur. Il réagit à la qualité de l'image, au nombre d'images par seconde, à la qualité du son etc. La perception du flux multimédia par l'utilisateur a été intensivement étudiée en psychologie de l'éducation et dans le domaine de l'interaction homme-ordinateur (HCI¹⁹). L'évaluation de la qualité de perception (QoP) est principalement empirique et elle dépend de la satisfaction de l'utilisateur recevant le flux multimédia. Un exemple d'évaluation de la QoP est décrit dans [ghinea98], où un certain nombre d'utilisateurs donnent des notes à plusieurs présentations multimédias qu'ils regardent. Nous définissons la QoP comme :

QoP = mesure du degré de satisfaction de l'utilisateur qui reçoit (regarde) un flux multimédia.

Le but d'un système VoD est d'offrir une bonne QoP aux utilisateurs. En supposant que le matériel côté client soit assez performant pour jouer (visualiser) les flux multimédias demandés, les performances du serveur multimédia et du réseau sont les facteurs qui déterminent la QoP. Le problème que l'on se pose est d'exprimer la QoP en termes de performances de l'ensemble réseau - serveur multimédia.

On sait que les clients interagissent avec un système VoD à travers des lecteurs multimédias (Figure 40).

¹⁹ HCI : acronyme pour Human Computer Interface (Interface Homme-Machine)

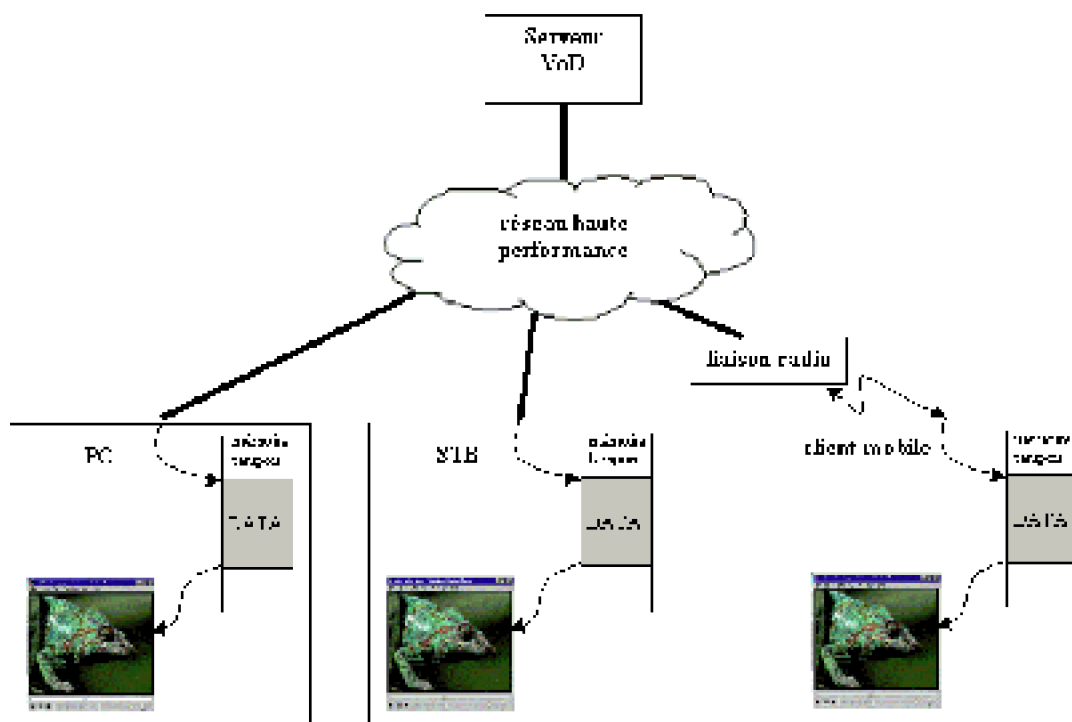


Figure 40 Clients possibles pour un serveur VoD. Les logiciels qui visualisent les présentations multimédias provenant du serveur utilisent les données déjà arrivées et emmagasinées dans une mémoire tampon (buffer) locale.

Les lecteurs multimédias trouvés sur les clients d'un système VoD utilisent une mémoire tampon (buffer) pour recevoir les données provenant du serveur. Dès que le lecteur a besoin des données pour la visualisation il les cherche dans son buffer. S'il trouve ces données la visualisation se poursuit sans problème ; si les données ne sont pas là on commence à avoir des images incomplètes ou manquantes ou des problèmes avec le son.

Nous nous situons dans un contexte où le matériel est assez performant pour jouer les flux reçus (cette supposition n'est pas très contraignante, étant donnée l'évolution des prix et des performances des ordinateurs de nos jours). Dans ce cas on peut considérer que le lecteur va fournir une bonne QoP si (et seulement si) il a toutes les données nécessaires en temps utile. Si les données prennent du retard par rapport aux instants où elles sont requises, la QoP ne sera pas très bonne.

Il est difficile de quantifier la QoP pour mesurer le degré de satisfaction d'un utilisateur, et ce n'est pas le but de cette thèse (des travaux dans ce domaine sont présentés en [ghinea98]). Ce qui nous importe est de savoir si un client est satisfait ou non, et combien de clients satisfaits un système VoD est capable de servir.

On peut définir alors la qualité de perception d'un client lorsqu'il regarde un flux multimédia O comme :

$QoP(O) = \begin{cases} 1, & \text{client satisfait} \\ 0, & \text{client insatisfait} \end{cases}$	(3)
$QoP(O) = \begin{cases} 1, & \text{client satisfait} \\ 0, & \text{client insatisfait} \end{cases}$	(3)

ou, en terme réseau – serveur :

$QoP(O) = \begin{cases} 1, & \text{données arrivées à temps} \\ 0, & \text{données arrivées en retard} \end{cases}$	(4)
$QoP(O) = \begin{cases} 1, & \text{données arrivées à temps} \\ 0, & \text{données arrivées en retard} \end{cases}$	(4)

4.1.2 Les facteurs de performance pour un serveur VoD

Un système VoD contient un ou plusieurs serveurs et plusieurs clients (comme décrit dans le Chapitre 3). Chaque client peut jouer un ou plusieurs flux multimédia. On note VoD l'ensemble du système, S l'ensemble de serveurs, C l'ensemble de clients, F l'ensemble de flux.

$VoD = (S, C, F)$	(5)
$VoD = (S, C, F)$	(5)

Un objet multimédia O est transféré en une séquence des blocs (B_1, \dots, B_n) . La visualisation d'images d'un bloc B_i ne peut pas commencer avant que le bloc B_i ne soit

complètement transféré. Pour éviter certains goulots d'étranglements (congestions) du réseau, le lecteur accumule une quantité de données dans une mémoire tampon avant de commencer à jouer un flux (Figure 41). Généralement la taille de cette mémoire tampon est égale à la quantité de données nécessaires pour t_0 secondes de visualisation (5 secondes pour [ms-wmp]).

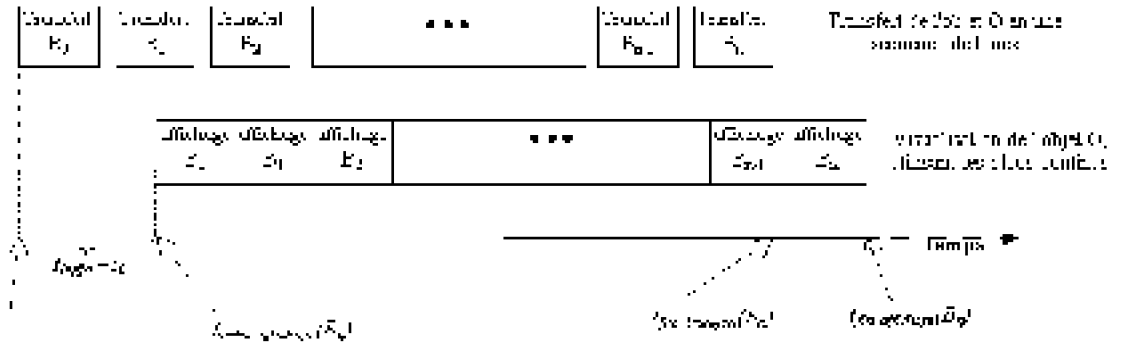


Figure 41 Mécanisme de transfert-visualisation d'un lecteur multimédia

Si on note :

$t_{\text{début_affichage}}(B_i)$ = l'instant où le bloc B_i va commencer à être joué

$t_{\text{fin_affichage}}(B_i)$ = l'instant où la visualisation du bloc B_i arrive à sa fin

$t_{\text{début_transfert}}(B_i)$ = l'instant où le transfert du bloc B_i commence

$t_{\text{fin_transfert}}(B_i)$ = l'instant où le bloc B_i est complètement transféré

alors la condition (4) se transforme dans :

$t_{\text{début_affichage}}(B_i) > t_{\text{fin_transfert}}(B_i), \forall i = \overline{0, n}$	(6)
$t_{\text{début_affichage}}(B_i) > t_{\text{fin_transfert}}(B_i), \forall i = \overline{0, n}$	(6)

Nous définissons la *qualité de perception partielle jusqu'à l'instant t* d'un flux multimédia O comme le rapport entre la quantité de données arrivées à temps et la quantité des données qui devrait arriver :

$QoP(O, t) = \frac{\sum_{i \in \{j t_{\text{debut_affichage}}(B_j) > t_{\text{fin_transfer}}(B_j), t_{\text{debut_affichage}}(B_j) < t\}} B_i }{\sum_{i \in \{j t_{\text{debut_affichage}}(B_j) < t\}} B_i }$	(7)
$QoP(O, t) = \frac{\sum_{i \in \{j t_{\text{debut_affichage}}(B_j) > t_{\text{fin_transfer}}(B_j), t_{\text{debut_affichage}}(B_j) < t\}} B_i }{\sum_{i \in \{j t_{\text{debut_affichage}}(B_j) < t\}} B_i }$	(7)

où $|B_i|$ dénote la taille du bloc B_i en octets. Ce rapport à une valeur entre 0 et 1, et il prend la valeur 1 si toutes les données sont arrivées à temps et d'autant plus faible qu'un grand nombre de données arrivent en retard. L'allure graphique de $QoP(O, t)$ est présentée dans la Figure 42.

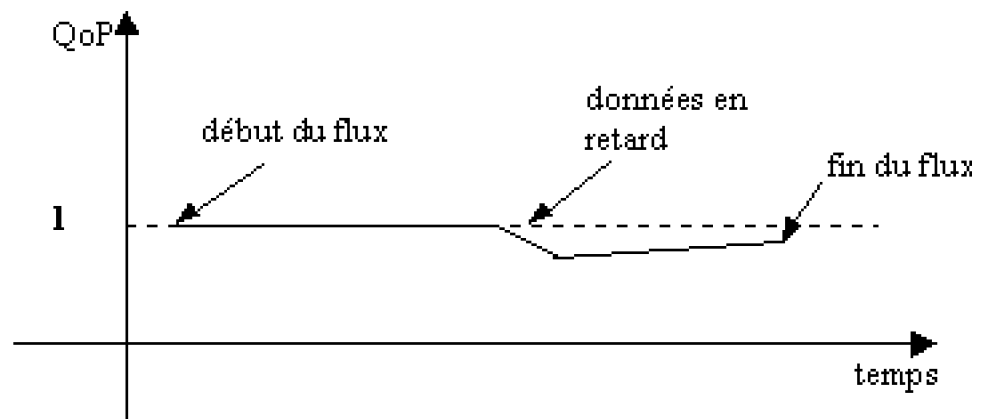


Figure 42 Représentation graphique de $QoP(O, t)$

QoP ne peut pas être nul, puisque le "en retard" se mesure par rapport au temps de début de la lecture. Le premier bloc n'est donc jamais en retard. Par convention on peut poser $QoP=0$ si la connexion ne peut pas s'établir.

La qualité de perception pour un flux O sera définie comme la qualité de perception partielle jusqu'à la fin du flux :

$QoP(O) = \frac{\sum_{i \in \{j t_{\text{debut_affichage}}(B_j) > t_{\text{fin_transfer}}(B_j)\}} B_i }{ O }$	(8)
	(8)

$QoP(O) = \frac{\sum_{i \in \{j t_{\text{debut_affichage}}(B_j) > t_{\text{fin_transfert}}(B_j)\}} B_i }{ O }$	(8)
$QoP(O) = \frac{\sum_{i \in \{j t_{\text{debut_affichage}}(B_j) > t_{\text{fin_transfert}}(B_j)\}} B_i }{ O }$	

Les propriétés de QoP :

1. $QoP(O) = QoP(O, +\infty)$
2. $QoP(O) \in [0, 1], \in O$
3. $QoP(O) = 1 \iff (QoP(O, t) = 1, \forall t), \in O$
4. $QoP(O) = 1 \iff$ tous les données sont arrivées à temps,
5. $QoP(O) = 0 \iff$ la connexion ne peut pas s'établir

En conclusion, nous pouvons dire que pour un système VoD les clients sont satisfaits si la condition $QoP(O, t)=1$ est satisfaite simultanément pour tous les flux à chaque instant t . Dans la prochaine section nous essayons de décrire une procédure d'évaluation de performances pour les serveurs VoD. Cette procédure utilisera le facteur QoP défini ci-dessus.

4.2 Méthodologie de tests

Pour les serveurs Web, les logiciels d'évaluation existants s'intéressent au nombre maximal de requêtes traitées par seconde et au débit total maximal du serveur (voir le Chapitre 2). Dans le domaine de la VoD, il est normal que nous nous intéressions au nombre maximal de flux de bonne qualité délivrés par le serveur (les flux pour lesquels $QoP=1$) et au débit total maximal du serveur. Les deux facteurs pris en compte sont :

$utilisateurs_satisfaits(VoD) = \text{card}\{\text{flux} \text{flux} \in F, QoP(\text{flux}) = 1\}$	(9)
$utilisateurs_satisfaits(VoD) = \text{card}\{\text{flux} \text{flux} \in F, QoP(\text{flux}) = 1\}$	(9)

et

$\text{débit}(VoD) = \sum_{\{flux flux \in F, QoP(flux)=1\}} \text{débit}(flux)$	(10)
$\text{débit}(VoD) = \sum_{\{flux flux \in F, QoP(flux)=1\}} \text{débit}(flux)$	(10)

La procédure d'évaluation de performances doit trouver les limites de ces deux facteurs. En principe, on augmente le nombre des clients d'un serveur VoD jusqu'à dégradation du QoP. Le scénario est le suivant :

- 1.
- sur le serveur VoD on installe des objets multimédias différents d'une durée assez longue (quelques heures chacun) au débit souhaité ; cela est l'équivalent du *workset* décrit dans le Chapitre 2 pour les outils d'évaluation de serveurs Web.
- 2.
- on connecte plusieurs ordinateurs au serveur en utilisant un réseau rapide ; ils vont simuler les lecteurs réels avec un logiciel qui calcule la QoP pour chaque flux ;
- 3.
- $n = 0$ (le nombre de flux démarrés simultanément)
- 4.
- on démarre un nouveau flux sur un ordinateur client; le logiciel de test redémarrera le flux quand il arrive à la fin (boucle infinie) ; $n = n+1$
- 5.
- on répète le pas 4 jusqu'à la saturation du serveur; dès qu'un client ne vérifie pas $QoP=1$ on arrête les tests ; le nombre maximal de clients satisfaits simultanément est n .

Si on note la qualité de perception pour tout le système VoD (\overline{QoP}) comme la

moyenne du QoP pour tous les flux simultanés :

$\overline{QoP}(VoD) = \left(\sum_{flux \in F} QoP(flux) \right) / \text{card}(F)$	(11)
$\overline{QoP}(VoD) = \left(\sum_{flux \in F} QoP(flux) \right) / \text{card}(F)$	(11)

les résultats de cette procédure de tests ont une forme graphique comme celle présentée dans la Figure 43. Nous supposons que le système VoD à tester serve les clients d'une manière équilibré, et que le réseau est configuré d'une telle manière que tous les clients peuvent obtenir les mêmes performances. Donc, si un client a un QoP inférieur à 1, il va généré aussi une dégradation du QoP des autres clients.

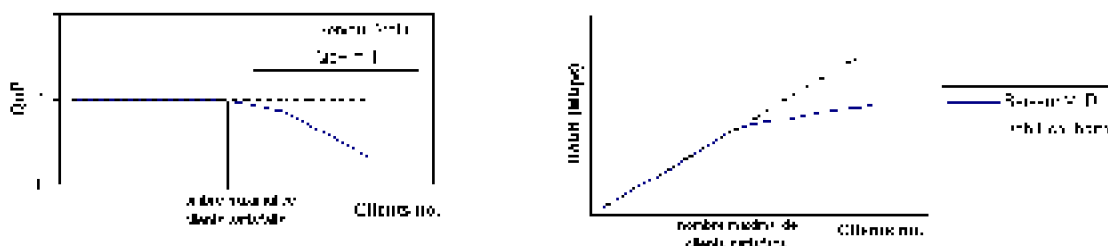


Figure 43 Résultats d'une procédure d'évaluation de performances : le nombre maximal de clients satisfaits simultanément et le débit maximal du serveur

Dès que le \overline{QoP} du système descend au-dessous de 1, les clients ne

bénéficieront pas tous d'une bonne qualité du service VoD. Même si le débit total peut croître encore, le résultat n'entre pas en compte pour les performances du système.

4.3 Résultats expérimentaux

Pour valider cette stratégie de tests nous avons mesuré les performances d'un serveur VoD (AMS de CSTI). Comme il y avait déjà plusieurs systèmes testés d'une manière empirique, mais sûre, avec des clients réels, nous avons pu vérifier les résultats de notre procédure.

4.3.1 Plate-forme des tests

La plate-forme de test utilisée est basée sur le système d'exploitation WindowsNT (Figure 44). Elle est composée d'un serveur Dell, avec un processeur PentiumIII à 533 MHz, 256 MB RAM et un bus PCI sur 64 bits. Le système de stockage utilise une carte RAID5 avec un cache de 128 MB, sur laquelle nous avons connecté quatre disques SCSI de 18 GB à 7200 RPM.

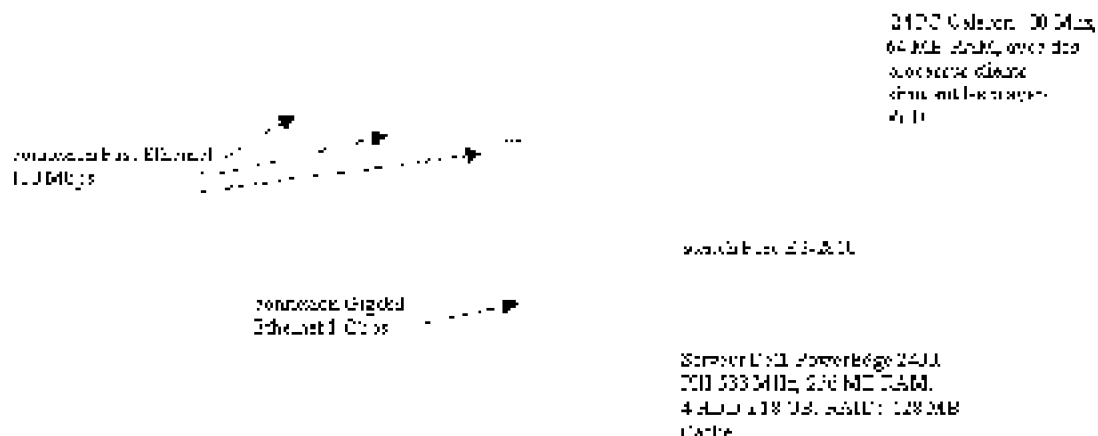


Figure 44 Plate-forme de tests pour un serveur VoD

Les clients sont des PC Celeron à 400 MHz avec 64 MB RAM. Pour pouvoir simuler plusieurs clients sur un même ordinateur, nous avons réalisé un logiciel qui simule un lecteur multimédia, mais qui ne visualise pas les flux (la visualisation d'un flux consomme beaucoup des ressources). Le logiciel simule la lecture concurrente d'un ou plusieurs flux multimédias et il affiche différentes caractéristiques numériques du transfert, dont la QoP (Figure 45). Les protocoles de communication avec le serveur VoD utilisés alternativement dans nos expériences sont HTTP (transfert par blocs) et le protocole de communication d'AMS ([ams]).

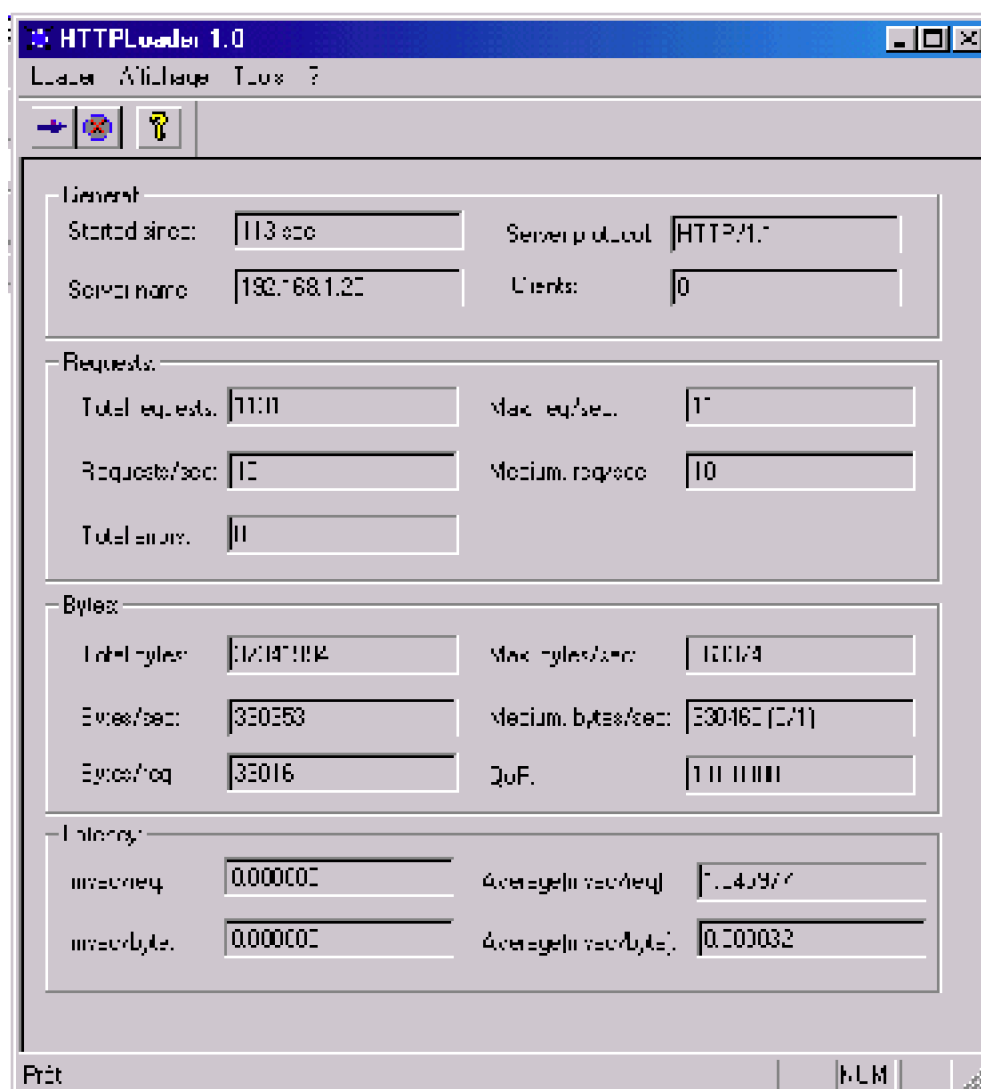


Figure 45 Logiciel qui simule la lecture concurrente de un ou plusieurs flux multimédias

4.3.2 Les performances d'un serveur VoD

Le serveur VoD que nous avons testé sur cette configuration est Antara Multimedia Server de CSTI ([ams]) en raison de sa disponibilité pour l'auteur (accès au source du protocole de communication). La version WinNT de ce serveur à été installée sur le serveur Dell, et son catalogue d'objets multimédias à été rempli avec des objets de formats divers, encodés à différents débits.

Nous avons configuré les clients pour travailler chacun avec un objet différent, et nous avons commencé à démarrer des clients. Tant que la

\overline{QoP}

descend pas au-dessous de 1, nous démarrons de nouveaux simulateurs, jusqu'à

$$\overline{QoP} < 1).$$

Les résultats représentent le nombre maximal de clients satisfaits en parallèle et le débit du serveur pour ces clients. Comme le nombre de clients satisfaits dépend aussi du débit de chaque flux, nous avons effectués plusieurs expériences, sur des objets encodés à différents débits: MPEG1 à 1.5 Mbps, MPEG2 à 3 Mbps, MPEG2 à 6 Mbps, MPEG2 à 8 Mbps. Ces formats de fichiers ont été choisis parce qu'ils sont les plus utilisés dans le domaine de la VoD haut-débit.

Les résultats pour un système avec des clients recevant des flux à 6 Mbps sont présentés dans la Figure 46.

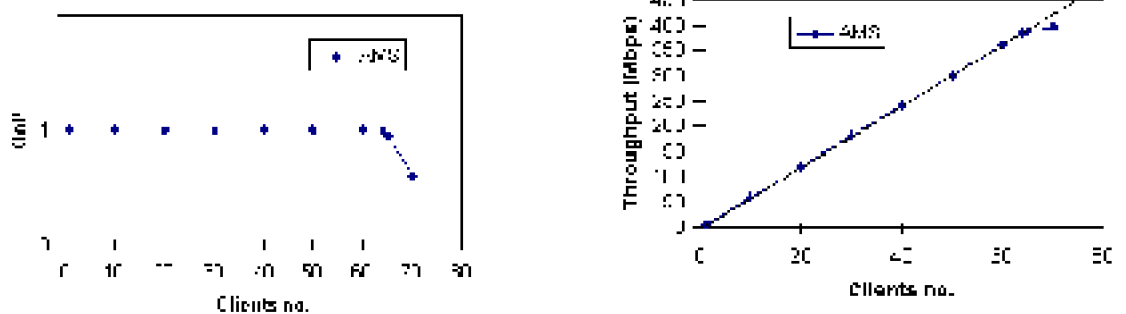


Figure 46 Les performances maximales du serveur AMS envoyant des flux MPEG2 à 6 Mbps. Dans la configuration présentée, il peut servir dans des bonnes conditions 64 flux différents pour un débit total de 384 Mbps.

On observe que le serveur fournit une bonne QoS pour maximum 64 clients concurrents. Dès que l'on dépasse le nombre de 64 clients la QoS descend au-dessous de 1, un lecteur témoin jouant un flux MPEG2 à 6 Mbps nous montre les effets : des problèmes de son et d'affichage. Le débit maximal lui aussi croît linéairement selon la droite d'équation :

$y = x * \text{débit}_{\text{flux}}$	(12)
$y = x * \text{débit}_{\text{flux}}$	(12)

où x représente le nombre de clients et y le débit théorique total des flux demandés. Pour plus de 64 clients, le débit total du système reste au-dessous de la droite définie par (12).

Les performances du serveur AMS pour l'émission des flux à différents débits sont présentées dans la Figure 47.

Figure 47 Les performances du serveur AMS pour l'émission des flux à différents débits

Débit par flux	Clients satisfaits	Débit maximal
1.5 Mbps	205	310 Mbps
3 Mbps	114	342 Mbps
6 Mbps	64	384 Mbps
8 Mbps	50	400 Mbps

Les résultats obtenus avec notre méthode de test respectent les performances de l'AMS vérifié avec des clients réels. L'avantage de notre méthode consiste dans le fait qu'elle est moins coûteuse en termes de ressources humaines et matérielles.

On observe également que le débit total peut augmenter d'une manière importante si on utilise moins de flux à des débits élevés qu'en utilisant plus de flux à des débits plus bas. Le raison de ce comportement réside dans les performances du système de stockage, qui sont plus élevées s'il y a moins des fichiers en accès concurrente. Des expérimentation justifiant cette affirmation se trouvent dans le Chapitre 6.

4.4 Conclusion

La méthodologie de tests présentée dans ce chapitre permet de déterminer les performances maximales d'un serveur VoD (en termes de nombre de clients et de débit). Elle correspond aux méthodologies d'évaluation de performances de serveurs Web (Chapitre 2), mais tient compte des caractéristiques particulières des flux multimédias. Le facteur de qualité défini (la qualité de perception) et son utilisation dans notre méthodologie permettent l'exécution automatique des tests, sans faire appel à des témoins humains pour mesurer la qualité de flux multimédia.

Chapitre 5. Une stratégie d'utilisation du protocole HTTP pour la VoD

5.1 Introduction

Le protocole HTTP est conçu pour une communication continue monodirectionnelle. De ce fait, pour l'implémentation de fonctionnalités telles que le déplacement de la position de lecture sur un objet multimédia ("seek"), le client doit attendre que les données entre la position courante et la position du seek soient transférées. Ceci impose une attente à l'utilisateur et un transfert important de données inutiles.

Un autre problème dû à l'utilisation de ce protocole réside dans l'utilisation inefficace de la bande passante. Le serveur classique, en réponse à une requête sur un objet, essaie d'envoyer l'objet le plus vite possible, en générant une utilisation inefficace de la bande passante, qu'il s'agisse d'un clip audio de quelques secondes ou d'un film de plusieurs heures. RealNetworks a publié des statistiques selon lesquelles la longueur moyenne par *écoute* est d'approximativement 4 minutes, mais souvent le matériel source a une heure ou plus [rn-http]. Il y a donc un besoin fondamental de contrôler le taux de transfert d'un objet multimédia.

Plusieurs lecteurs multimédias utilisent le protocole HTTP ([ms-wmp], [rn-rp], [vivo],

[xing], [mpegtv]), souvent comme alternative à d'autres protocoles multimédias spécialisés (d'habitudes pour passer les pare-feu), mais leurs stratégies se heurtent aux problèmes présentés ci-dessus.

Pour résoudre ces problèmes, nous décrivons une nouvelle méthode d'utilisation du protocole HTTP. Cette méthode correspond à une stratégie de travail type "pull" depuis le client (voir Chapitre 3) et permet l'utilisation efficace de l'opération "seek" et un certain contrôle de la bande passante. La quantité de données inutiles qui traversent le réseau sera ainsi limitée à des quantités négligeables. Cette méthode ne demande aucune modification du serveur Web : seule la stratégie des requêtes HTTP côté client est modifiée, en gardant le protocole inchangé.

5.1.1 Pourquoi utiliser HTTP dans la VoD ?

Les impacts industriels de cette méthode sont importants. Elle permet l'utilisation des serveurs Web standards comme des serveurs VoD sans que les performances côté clients soient dégradées (elles sont même améliorées dans certains cas). Les prix des deux serveurs VoD commerciaux parmi les plus répandus pour Intranet, NetShow et RealServer, sont présentés dans la Figure 48, conformément à [netshow-price] et [real-price]. Par rapport à eux, la plupart des serveurs Web sont gratuits.

Figure 48 Les prix (en \$) des deux serveurs VoD plus répandus: NetShow et RealSystem Server

	RealServer Intranet	NetShow²⁰
60 clients	2793 \$	5337 \$
200 clients	7270 \$?
500 clients	18191 \$?

On voit très bien que le prix d'un tel serveur est élevé, et d'autres serveurs utilisant des architectures matérielles dédiées arrivent à des prix d'un ordre de grandeur plus important ([ams], [oracle], [ncube], [alex]).

Nous essayons de montrer qu'il existe des solutions beaucoup moins coûteuses pour obtenir des services de même qualité, à travers des serveurs Web utilisés comme des serveurs VoD. Les essais réalisés chez Microsoft et RealNetworks ont mis en doute l'utilisation du protocole HTTP pour la VoD ([ms-http], [rn-http]), en soutenant que HTTP ne permet pas l'implémentation des fonctionnalités VCR étendues. Notre but dans ce chapitre est de démontrer le contraire.

²⁰ NetShow est remplacé depuis 2000 par Windows Media Technologies, une plate-forme multimédia optimisée pour distribuer des flux bas-débit sur Internet. Cette plate-forme est gratuite, mais elle demande l'achat d'une licence WinNT pour le serveur et des licences pour chaque client.

5.1.2 La stratégie de transfert HTTP actuelle

La méthode de transfert d'objet utilisée jusqu'à présent à l'aide de protocole HTTP est d'envoyer une requête HTTP au serveur pour tout l'objet et d'attendre la réponse. Ensuite, le serveur envoie le plus vite possible la réponse qui contient l'objet demandé (Figure 49). On appelle cette stratégie de transfert "transfert complet", et elle correspond à une stratégie "push" dans la VoD.

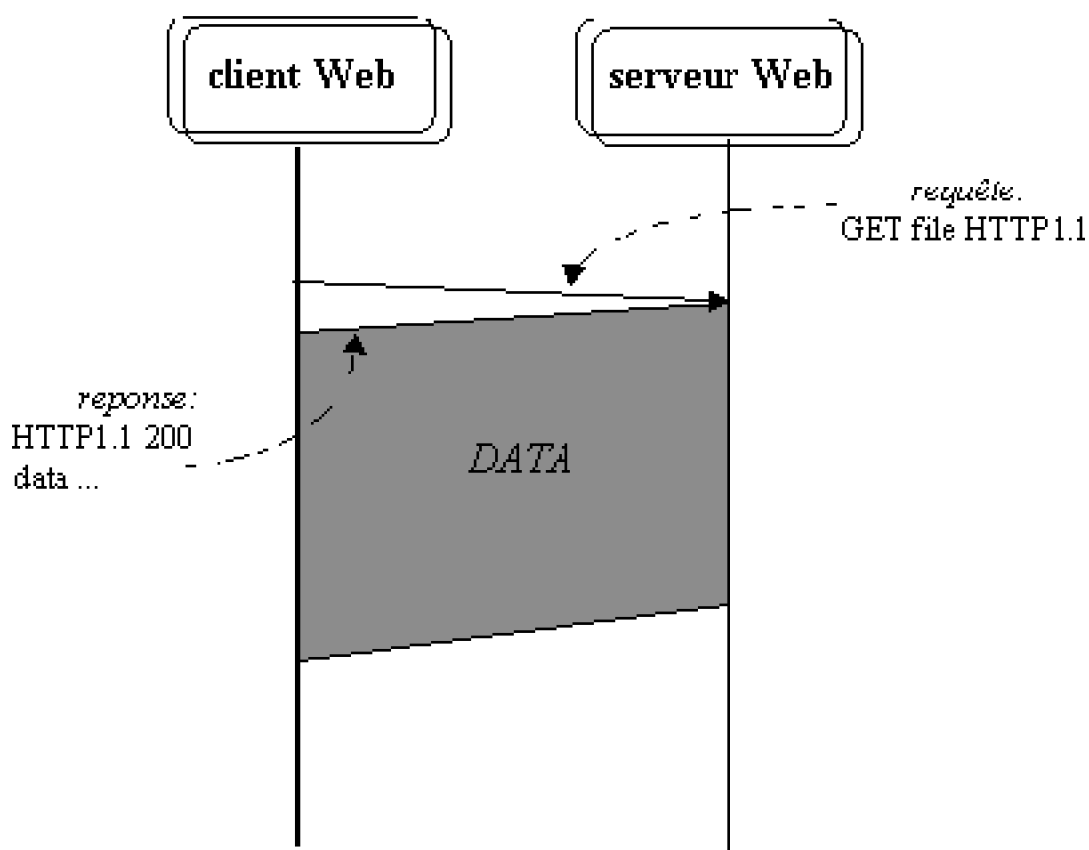


Figure 49 L'interaction client - serveur actuelle dans le cadre du protocole HTTP

Pour chaque objet le client envoie une requête HTTP comme par exemple :

```

GET /ads15mbs.mpg HTTP/1.1Host: soft15Connection: Keep-Alive
GET /ads15mbs.mpg HTTP/1.1Host: soft15Connection: Keep-Alive
    
```

et le serveur lui répond avec :

```
HTTP/1.1 200 OKDate: Mon, 13 Sep 1999 14:17:34 GMTServer: Apache/1.3.6Last-Modified: Fri, 11 Jun 1999 20:39:23 GMTContent-Length: 35005360
```

```
HTTP/1.1 200 OKDate: Mon, 13 Sep 1999 14:17:34 GMTServer: Apache/1.3.6Last-Modified: Fri, 11 Jun 1999 20:39:23 GMTContent-Length: 35005360
```

5.1.3 Les défauts de la stratégie actuelle

Comme nous l'avons précisé dans l'introduction du chapitre, cette stratégie de transfert à un certain nombre de défauts.

Premièrement, le temps d'interaction pour la réalisation d'un seek est important. Une fois que le transfert est commencé, le lecteur client ne peut pas faire une opération SEEK sans attendre qu'une quantité de données inutiles soit transférée (Figure 50). Les données situées entre la position courante à la réception du SEEK et la nouvelle position désirée seront transférées même si elles ne sont pas nécessaires pour la visualisation (dans le cas où les données ne seraient déjà transférées).

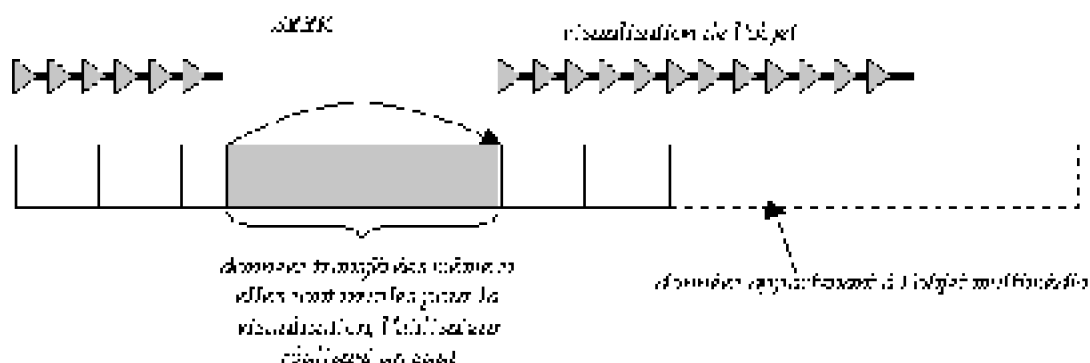


Figure 50 L'effet de l'utilisation du seek dans HTTP

Nous sommes tentés de nous demander pourquoi ne pas fermer la connexion HTTP et la re-ouvrir après chaque seek. Dans ce cas les données inutiles ne sont pas transférées. Mais on tombe sur le problème de "slow start" du HTTP Chapitre 2), problème qui générera des latences importantes pour chaque opération seek.

Un autre effet de cette utilisation du HTTP réside dans l'utilisation inefficace de la bande passante. L'exemple de la Figure 51 montre la différence entre le débit avec les données arrivant au client (le serveur les envoie le plus vite possible) et le débit réel du flux. On voit que le flux est transféré assez vite par rapport à son taux réel d'encodage (le débit réel du flux). Si la visualisation s'arrête après un certain temps, une bonne quantité de données sera transférée inutilement.

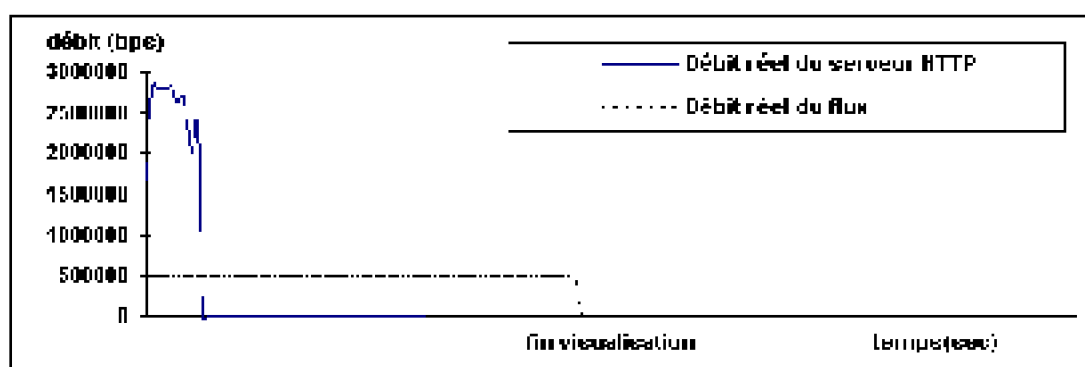


Figure 51 Le débit de transfert d'un flux avec la stratégie HTTP actuelle par rapport à son débit d'encodage

5.2 Présentation d'une nouvelle stratégie de transfert HTTP

Pour éviter les problèmes générés par la méthode de transfert HTTP actuelle, nous utiliserons la propriété du protocole HTTP1.1 qui permet de demander un bloc d'un fichier en utilisant l'option "Range" (détaillée en [http]). Donc, nous transférerons ce fichier en petits blocs, en général de taille équivalente (Figure 52). Le client ne demandera que les blocs nécessaires et seulement quand il en a besoin.

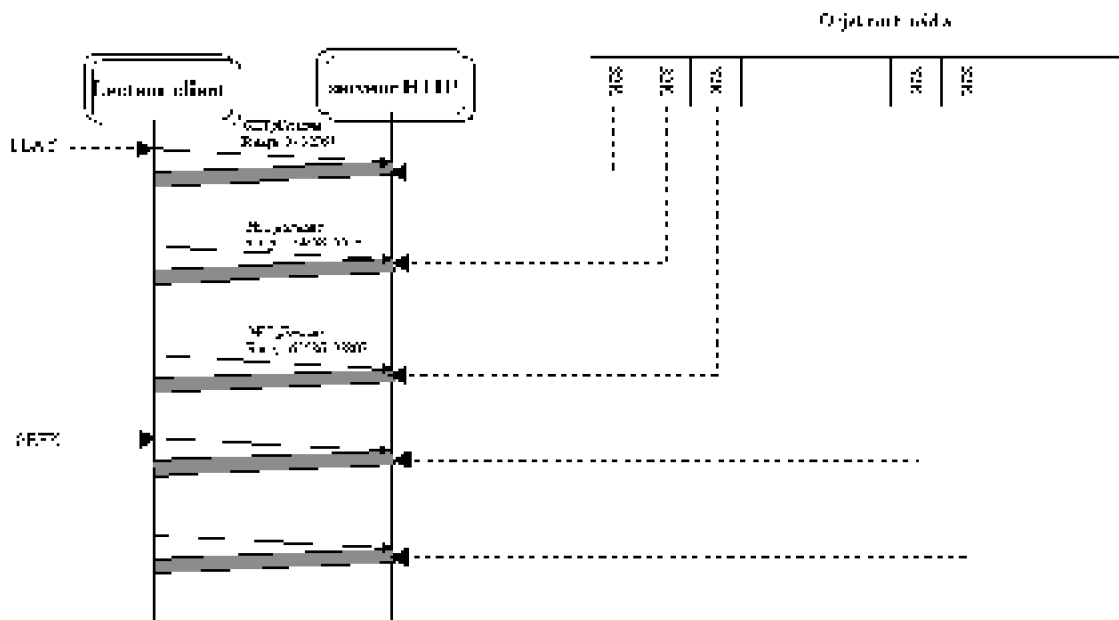


Figure 52 L'interaction entre un lecteur multimédia et un serveur Web proposée pour le transfert d'objets multimédia en utilisant HTTP

Pour chaque bloc le client envoie une requête HTTP, comme par exemple :

```
GET /ads15mbs.mpg HTTP/1.1Host: soft15Connection: Keep-AliveRange:
bytes=32768-65535
```

```
GET /ads15mbs.mpg HTTP/1.1Host: soft15Connection: Keep-AliveRange: bytes=32768-65535
```

et le serveur lui répond avec :

```
HTTP/1.1 206 Partial ContentDate: Mon, 13 Sep 1999 14:17:34 GMTServer:
Apache/1.3.6Last-Modified: Fri, 11 Jun 1999 20:39:23 GMTContent-Length:
32768Content-Range: bytes 32768-65535/35005360DATA ...
```

```
HTTP/1.1 206 Partial ContentDate: Mon, 13 Sep 1999 14:17:34 GMTServer:
```

```

HTTP/1.1 206 Partial ContentDate: Mon, 13 Sep 1999 14:17:34 GMTServer:
Apache/1.3.6Last-Modified: Fri, 11 Jun 1999 20:39:23 GMTContent-Length:
32768Content-Range: bytes 32768-65535/35005360DATA ...
Apache/1.3.6Last-Modified: Fri, 11 Jun 1999 20:39:23 GMTContent-Length:
32768Content-Range: bytes 32768-65535/35005360DATA ...
    
```

Cette stratégie est l'équivalent de la méthode "pull" dans la VoD, et nous l'appelons "transfert par blocs".

5.2.1 Avantages et inconvénients

L'utilité de cette méthode réside dans la possibilité de réaliser des opérations seek efficaces, que nous ne pouvons pas réaliser avec le paradigme "transfert complet" ("full file transfer"). Quand le client réalise un seek, il finira le transfert du bloc en cours de transfert (s'il y a un) et le prochain bloc demandé commencera à la position du seek (Figure 53).

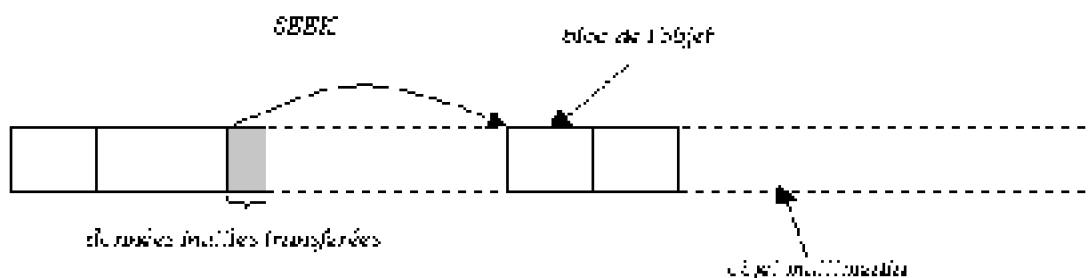


Figure 53 La stratégie du seek avec la méthode proposée

La quantité de données inutiles transférées sur le réseau sera au maximum la taille d'un bloc pour chaque seek. La quantité maximale des données inutiles transférée pour un seek dans les deux stratégies est présenté dans la Figure 54.

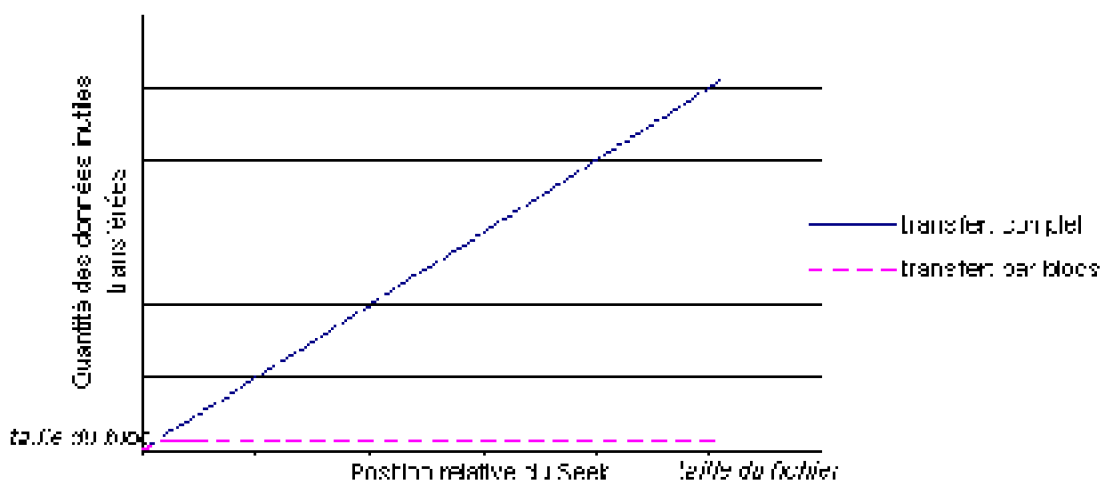


Figure 54 La quantité maximale des données inutiles transférées pour un seek dans les deux stratégies: "transfert par blocs" et "transfert complet"

Un autre avantage de cette méthode : le client peut contrôler, jusqu'à un certain degré, le débit de transfert pour un objet multimédia. L'uniformité de ce transfert dépend, parmi d'autres choses, de la taille des blocs du fichier transféré. Parce que le client demande les blocs nécessaires quand il a besoin, le serveur va répondre à ces requêtes le plus vite possible, mais les requêtes vont arriver suivant le débit de l'objet multimédia.

Dans la Figure 55 nous présentons une comparaison entre le débit réseau de ces deux stratégies HTTP, en utilisant un objet MPEG1 encodé à 1.5 Mbps. Nous observons que l'utilisation de notre stratégie permet de diffuser l'objet à un débit qui suit le taux réel d'encodage de l'objet. Par contre, l'autre stratégie de transfert HTTP (transfert complet) génère une utilisation qui n'a rien à voir avec le débit réel de l'objet.

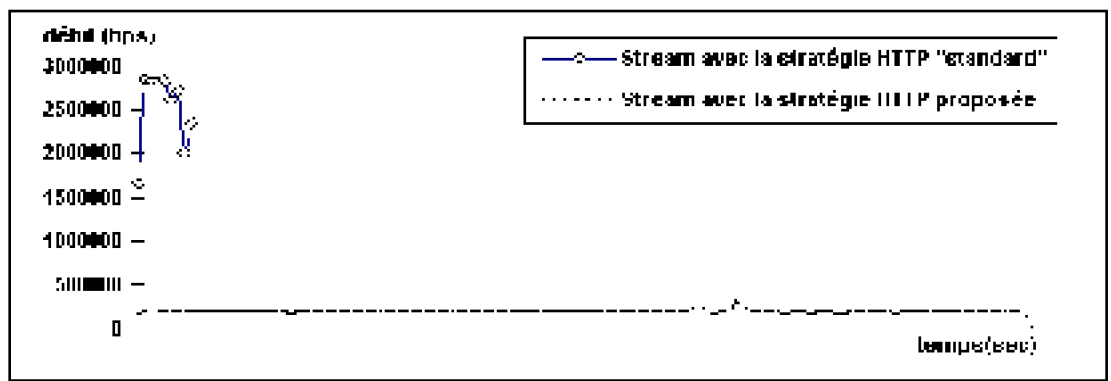


Figure 55 Une comparaison entre les deux stratégies HTTP côté client (en utilisant un fichier MPEG1 a 1.5Mbps)

Un effet secondaire réside dans la croissance du nombre de requêtes traitées par le serveur Web. Dans la stratégie conventionnelle, le serveur ne devait traiter qu'un seul en-tête HTTP pour chaque objet. Au contraire, avec la stratégie "transfert par blocs" le nombre d'en-têtes est beaucoup plus élevé. La formule suivante permet de faire une approximation de ce nombre de requêtes pour le transfert d'un fichier :

$requêtes = \frac{taille_{fichier}}{taille_{bloc}}$	(13)
$requêtes = \frac{taille_{fichier}}{taille_{bloc}}$	(13)

Mais cette croissance n'est pas très contraignante, les serveurs Web actuels étant capables de traiter des milliers de requêtes simultanément (Chapitre 2). A la fin de ce chapitre nous montrons les performances réelles d'un serveur Web qui serve des flux

avec la stratégie "transfert par blocs", et nous verrons que l'augmentation du nombre de requêtes ne pénalise pas les performances du système.

5.3 Analyse de la stratégie

Pour analyser la performance de la stratégie proposée nous utilisons un modèle simplifié du trafic HTTP présenté en [heideman97]. Nous évaluons le trafic des données sur le réseau pour la stratégie proposée ("transfert par blocs") par rapport à l'ancienne stratégie HTTP ("transfert complet"). On évalue aussi la surcharge déterminée par les en-têtes HTTP supplémentaires transférés pour chaque paire requête/réponse.

5.3.1 Analyse de la stratégie HTTP standard

La stratégie HTTP standard pour le transfert d'un fichier consiste en un échange d'informations entre le serveur et le client Web d'après le scénario suivant :

- le client réalise une connexion TCP avec le serveur sur le port 80 (le port par défaut)
- le client envoie une requête au serveur
- le serveur répond au client

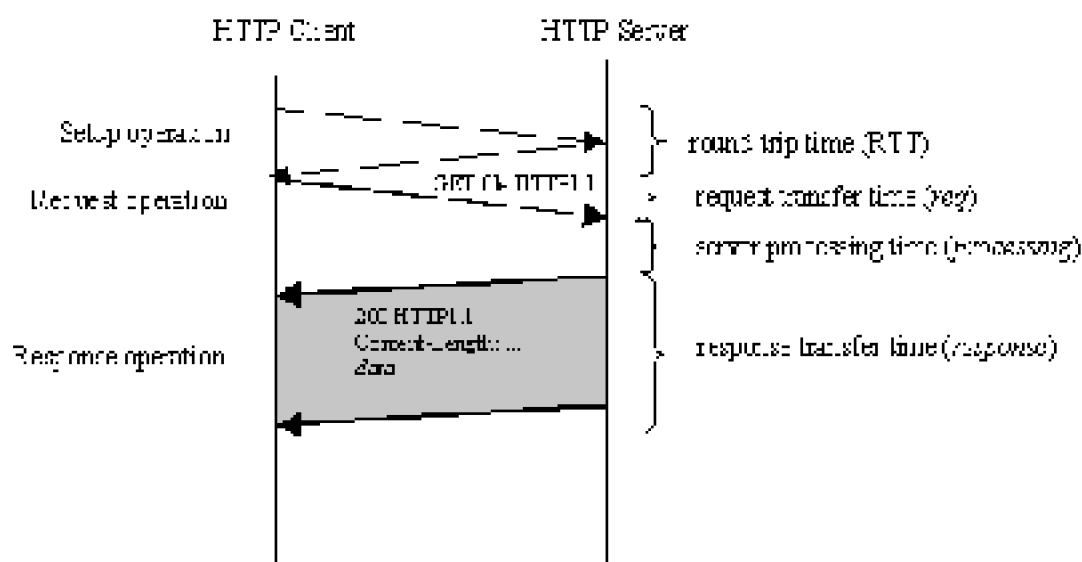


Figure 56 Les paquets échangés sur une connexion HTTP/TCP. Les flèches en gras indiquent un transfert des données et les flèches en trait fin indiquent des paquets de synchronisation (SYN) ou d'acquittement (ACK)

La Figure 56 montre la séquence d'échange habituelle des paquets TCP pour réaliser une requête HTTP. Plusieurs paramètres sont nécessaires pour caractériser cet échange :

$taille_{fichier}$	la taille de l'objet à transférer
$taille_{req}, taille_{rep}$	la taille d'une requête (réponse)
D	la quantité de données transférées sur réseau pour le transfert d'un objet multimédia
bw	la bande passante du réseau
$taille_H$	la taille d'un en-tête HTTP pour une requête ou une réponse (nous supposons que les en-têtes HTTP ont la même taille)

Dans ces conditions, la quantité de données transférées avec cette stratégie HTTP est:

$D = taille_{req} + taille_{rep}$ $= (taille_H) + (taille_H + taille_{fichier})$ $= 2 * taille_H + taille_{fichier}$	(14)
$D = taille_{req} + taille_{rep}$ $= (taille_H) + (taille_H + taille_{fichier})$ $= 2 * taille_H + taille_{fichier}$	(14)

Le temps nécessaire pour le transfert :

$T = RTT + t_{req} + t_{traitement} + t_{rep}$	(15)
$T = RTT + t_{req} + t_{traitement} + t_{rep}$	(15)

Si on suppose que le temps de traitement est négligeable ($t_{traitement} = 0$, le cas où la puissance de calcul du serveur est suffisante) et le réseau est un réseau rapide (Intranet, où on peut supposer que le temps nécessaire pour le transfert d'une quantité D de données est $t_D = D/bw$ et $RTT \approx 0$) :

$T = (2 * taille_H + taille_{fichier}) / bw$	(16)
$T = (2 * taille_H + taille_{fichier}) / bw$	(16)

5.3.2 Analyse de la stratégie HTTP proposée

La stratégie HTTP "transfert par blocs" que nous proposons suit le scénario suivant pour le transfert d'un objet :

-
- le client réalise une connexion TCP avec le serveur sur le port 80 (le port implicite)
-
- le client envoie une requête HEAD au serveur, demandant la taille de l'objet et éventuellement d'autres informations supplémentaires
-
- le serveur répond au client
-
- chaque fois que le client à besoin de données, il adresse une requête GET au serveur pour un bloc de données, qui lui répond le plus vite possible

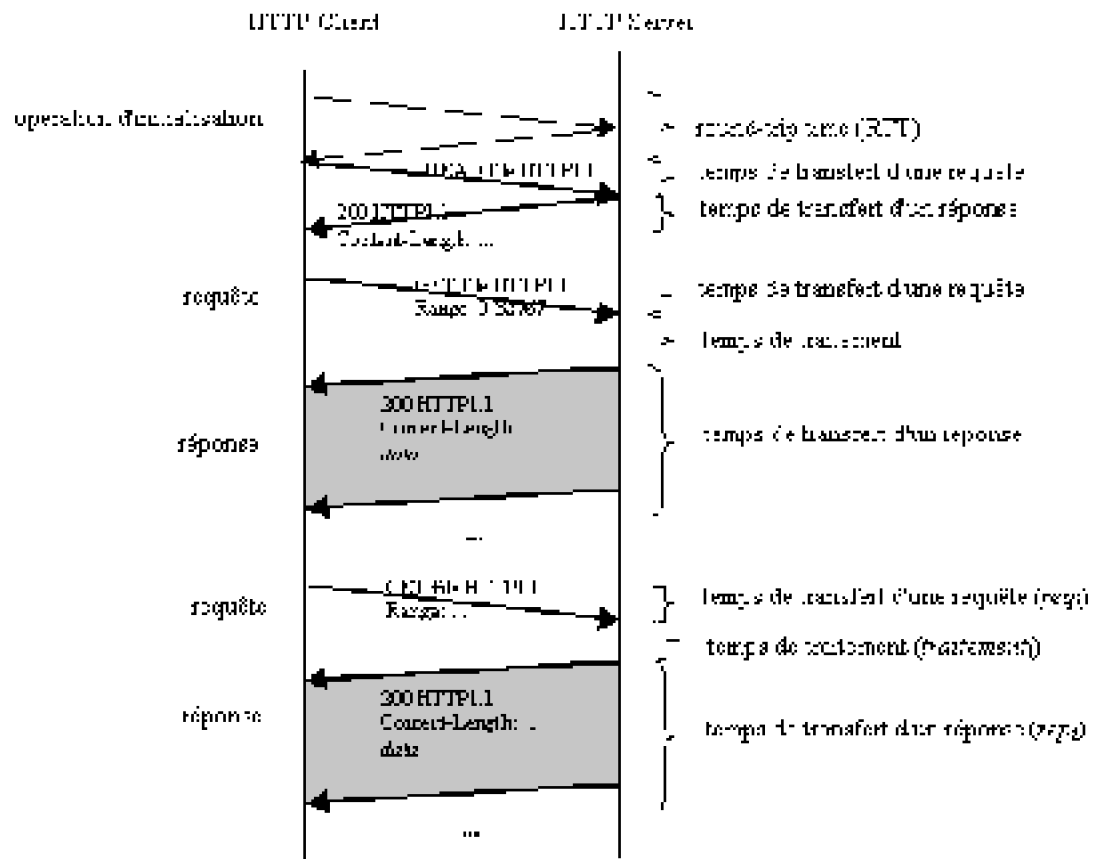


Figure 57 Les paquets échangés sur une connexion HTTP/TCP utilisant la stratégie "transfert par blocs"

Pour la nouvelle stratégie HTTP, on utilise encore quelques notations :

n	le nombre de requêtes nécessaires pour le transfert de l'objet fichier
taillebloc(i)	la taille du bloc i utilisé pour le transfert du fichier fichier ($taille_{fichier} = \sum_{i=1}^n taille_{bloc}(i)$)
taillereq(i)	la taille de la requête i (taillereq(i) = tailleH)
taillerep(i)	la taille de la réponse i (taillerep(i) = H+ taillebloc(i))

La quantité de données transférées avec cette stratégie HTTP est de (dans le contexte Intranet, où $t_D \approx D/bw$ et $RTT \approx 0$) :

$D' = \text{taille}_{req}(HEAD) + \text{taille}_{rep}(HEAD) + \sum_{i=1}^n \text{taille}_{req}(i) + \sum_{i=1}^n \text{taille}_{rep}(i)$ $= \text{taille}_H + \text{taille}_H + \sum_{i=1}^n \text{taille}_H + \sum_{i=1}^n (\text{taille}_H + \text{taille}_{bloc}(i))$ $= 2 * (n+1) * \text{taille}_H + \sum_{i=1}^n \text{taille}_{bloc}(i)$	(17)
$D' = \text{taille}_{req}(HEAD) + \text{taille}_{rep}(HEAD) + \sum_{i=1}^n \text{taille}_{req}(i) + \sum_{i=1}^n \text{taille}_{rep}(i)$ $= \text{taille}_H + \text{taille}_H + \sum_{i=1}^n \text{taille}_H + \sum_{i=1}^n (\text{taille}_H + \text{taille}_{bloc}(i))$ $= 2 * (n+1) * \text{taille}_H + \sum_{i=1}^n \text{taille}_{bloc}(i)$	(17)

Si on suppose que les blocs transférés ont la même taille (taille_{bloc}), alors :

$D' = 2\text{taille}_H * (n+1) + n * \text{taille}_{bloc}$	(18)
$D' = 2\text{taille}_H * (n+1) + n * \text{taille}_{bloc}$	(18)

5.3.2.1 Comparaison entre les deux stratégies

Le rapport entre les quantités de données transférées dans les deux stratégies est :

$\frac{D'}{D} = \frac{2\text{taille}_H * (n+1) + n * \text{taille}_{bloc}}{2\text{taille}_H + n * \text{taille}_{bloc}} = \frac{2\text{taille}_H + \text{taille}_{bloc} + \frac{2\text{taille}_H}{n}}{\frac{2\text{taille}_H}{n} + \text{taille}_{bloc}} \underset{2H/n \rightarrow 0}{\approx} 1 + \frac{2\text{taille}_H}{\text{taille}_{bloc}}$	(19)
$\frac{D'}{D} = \frac{2\text{taille}_H * (n+1) + n * \text{taille}_{bloc}}{2\text{taille}_H + n * \text{taille}_{bloc}} = \frac{2\text{taille}_H + \text{taille}_{bloc} + \frac{2\text{taille}_H}{n}}{\frac{2\text{taille}_H}{n} + \text{taille}_{bloc}} \underset{2H/n \rightarrow 0}{\approx} 1 + \frac{2\text{taille}_H}{\text{taille}_{bloc}}$	(19)

La croissance des données transférées est approximativement $2\text{taille}_H / \text{taille}_{bloc}$ (pour n assez grand : $2\text{taille}_H / \text{taille}_{bloc} = 0$). Pour une taille de bloc $\text{taille}_{bloc} = 32 \text{ KB}$ et la taille d'un en-tête HTTP $\text{taille}_H = 256 \text{ octets}$ nous obtenons $D'/D = 1.015$, donc une croissance de 1.5%. Dans la Figure 58 nous présentons la variation du trafic des cellules ATM en rapport avec des différentes tailles des blocs. On observe bien que les résultats réels suivent de près les résultats théoriques (avec une légère différence due à l'implémentation du protocole TCP sur un réseau ATM).

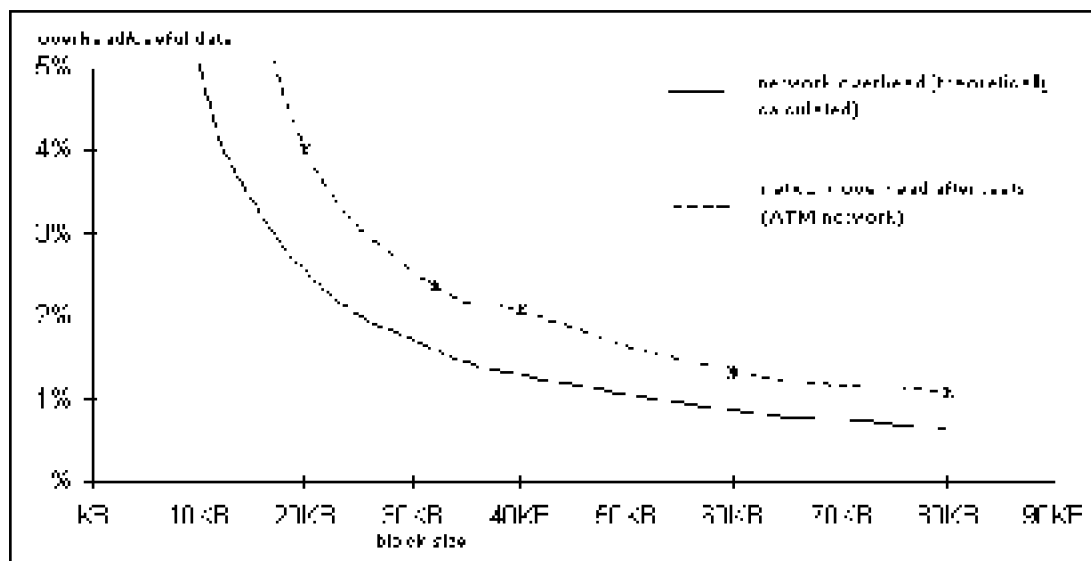


Figure 58 Représentation graphique du trafic de cellules ATM par rapport à la taille des blocs

5.3.2.2 Latence pour l'opération Seek

Le temps moyen passé entre l'émission de la commande seek par l'utilisateur et l'instant correspondant à la réponse est appelé "latence moyenne du seek" (en anglais: Average Seek Latency). Pour la stratégie "transfert par blocs", quand un client réalise une opération Seek, le bloc en cours de transfert est acheminé et le prochain bloc demandé au serveur commence exactement à la position du seek.

La latence moyenne du seek a le même ordre de grandeur que le temps de transfert d'un bloc, à la différence de la stratégie "transfert complet" qui a un temps du même ordre de magnitude que le temps de transfert du fichier complet.

Pour avoir une bonne latence du seek, il faut que la taille d'un bloc ne soit pas très grande. Dans les conditions d'un Intranet, la latence satisfait la relation:

$latence_moyenne_{seek} \geq \frac{taille_{bloc}}{bw}$	(20)
$latence_moyenne_{seek} \geq \frac{taille_{bloc}}{bw}$	(20)

ou

$taille_{bloc} \leq bw * latence_moyenne_{seek}$	(21)
$taille_{bloc} \leq bw * latence_moyenne_{seek}$	(21)

En réalité la bande passante disponible pour un client dépend également de combien de clients partagent la bande passante du serveur. Si le serveur est prévu pour servir un nombre de clients $clients_{no}$, nous considérons que la bande passante pour un client ne dépasse pas $bw / clients_{no}$. On en déduit donc :

$taille_{bloc} \leq \frac{(bw * latence_moyenne_{seek})}{clients_{no}}$	(22)
$taille_{bloc} \leq \frac{(bw * latence_moyenne_{seek})}{clients_{no}}$	(22)

Si on considère qu'une latence est acceptable si elle ne dépasse pas 0.5 secondes, pour un serveur planifié pour 50 clients sur un réseau Fast-Ethernet (100 Mbps) les blocs doivent avoir une taille qui respecte :

$taille_{bloc} \leq \frac{(100Mbps * 0.5s)}{50} = 1000000bits = 122Koctets$	(23)
$taille_{bloc} \leq \frac{(100Mbps * 0.5s)}{50} = 1000000bits = 122Koctets$	(23)

5.3.2.3 Considérations pour choisir la taille d'un bloc

Le problème que nous nous posons est de déterminer les tailles de blocs acceptables pour cette stratégie de transfert. On sait que la taille d'un bloc doit être assez grande pour ne pas surcharger le réseau et le serveur HTTP (section 5.2.1), mais pas trop grande pour garder une bonne latence pour les commandes VCR étendues, notamment *seek* (section précédente).

Si on connaît la surcharge maximale que l'on accepte pour le réseau, à partir de (20) on obtient :

$surcharge_{réseau} \geq \frac{2 * taille_H}{taille_{bloc}}$	(1)
$surcharge_{réseau} \geq \frac{2 * taille_H}{taille_{bloc}}$	(1)

où **surcharge**_{réseau} représente la surcharge du réseau par rapport à l'autre stratégie de transfert HTTP.

Pour exprimer la taille du bloc on arrive à :

$taille_{bloc} \geq \frac{2 * taille_H}{surcharge_{réseau}}$	(2)
$taille_{bloc} \geq \frac{2 * taille_H}{surcharge_{réseau}}$	(2)

On en déduit, en utilisant (22) et (2) :

$\frac{2 * taille_H}{surcharge_{réseau}} \leq taille_{bloc} \leq \frac{(bw * latence_{moyenne_{seek}})}{clients_{no}}$	(3)
$\frac{2 * taille_H}{surcharge_{réseau}} \leq taille_{bloc} \leq \frac{(bw * latence_{moyenne_{seek}})}{clients_{no}}$	(3)

Dans le cas particulier où l'en-tête HTTP à une taille de $taille_H = 256$ octets, la latence maximale tolérable sur un seek $latence_{moyenne_{seek}} = 0.5$ seconds, le nombre maximal de clients $clients_{no} = 50$, le réseau est un réseau Fast-Ethernet ($bw = 100$ Mbps) et la surcharge maximale acceptable du réseau de 2,5% ($surcharge_{réseau} = 0.025$), la taille d'un bloc devra se situer dans les limites [20 Koctets, 122 Koctets].

La relation (3) nous oblige également à accepter des valeurs raisonnables pour $surcharge_{réseau}$ et $clients_{no}$, des valeurs respectant la relation :

$$\frac{2 * taille_H}{surcharge_{réseau}} \leq \frac{(bw * latence_{moyenne_{seek}})}{clients_{no}}$$

	(4)
	(4)

Si cette relation n'est pas satisfaite, la stratégie de transfert par blocs ne pourra pas fournir une solution efficace pour le système étudié.

Dans les tests que nous faisons dans cette thèse nous travaillons avec de blocs de taille 32 KB ou 64 KB. Ces valeurs vérifient les conditions (3) pour toutes les configurations que nous avons utilisées. D'autres valeurs peuvent être utilisées, en fonction des caractéristiques particulières d'un système VoD (des objets à débit très élevé ou très bas, par exemple) mais elles doivent vérifier (3).

5.4 Implémentation

Afin de montrer les performances réelles de la stratégie "transfert par blocs" proposée, nous avons réalisé une implémentation de cette stratégie dans les deux lecteurs multimédias le plus utilisés sur Internet : Windows Media Player de Microsoft ([ms-wmp]) et Real Video de Real Networks ([rn-rv]).

Notre intérêt est de remplacer l'implémentation de HTTP dans ces deux lecteurs avec notre implémentation, et de faire une comparaison entre les deux.

Comme nous l'avons présenté dans le Chapitre 3, les deux lecteurs multimédias utilisent une même architecture logicielle. Les objets multimédias sont traités à travers un graphe composé de plusieurs modules (filtres), qui se transmettent les données. Il existe plusieurs types de modules mais ceux qui nous concernent ici sont les modules '*source de données*'. Un module source alimente le graphe du lecteur avec les données d'un objet, sans savoir nécessairement ce qu'il se passera avec ces données plus loin dans le graphe. D'habitude un module source implémente un protocole réseau pour récupérer les objets multimédias.

Les deux lecteurs n'acceptent pas les mêmes modules (de par leur stratégie de développement), donc nous avons implémenté un module source pour Windows Media Player et un autre pour Real Video. Les différences entre ces deux modules ne sont pas significatives, il s'agit d'une API différente à utiliser pour s'interfacer avec le lecteur.

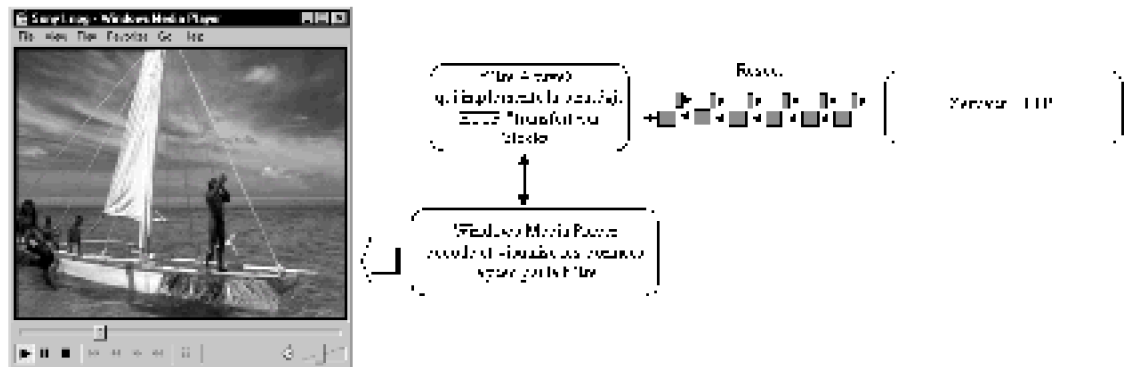


Figure 59 Stratégie de travail d'un module pour un lecteur multimédia utilisant la méthode de transfert par blocs

Comme résultat nous avons obtenu des lecteurs multimédias capables de jouer les principaux types des formats multimédias compressés (parmi d'autres MP3, MPEG1, MPEG2, MPEG4, QuickTime), mais utilisant notre implémentation de HTTP.

5.4.1 Qualité des fonctionnalités VCR de l'implémentation

Nous nous posons maintenant la question : est-ce que notre implémentation offre une bonne satisfaction aux utilisateurs en termes d'interactivité VCR? La réponse à cette question ne peut être donnée que par une étude sur un certain nombre d'utilisateurs.

Pendant ma thèse j'ai participé à la réalisation d'un système VoD commercial nommé AMS ([ams]). Une première version de ce système a été installée dans un hôtel à Dubai²¹, avec un lecteur VoD (Windows Media Player) utilisant le module présenté. Les clients de l'hôtel utilisent un système d'appel pour signaler les éventuels mécontentements concernant la qualité du service VoD. Aucun des événements signalés pendant trois mois d'utilisation intensive ne concerne les fonctionnalités VCR du lecteur. Elles concernent spécialement la bibliothèque des objets disponible, les services de DelayedTV implémentés en utilisant IP multicast et d'autres services connexes.

Conséquemment, nous considérons que la réponse à notre question est que les utilisateurs habitués à des magnétoscopes et des vidéodisques sont satisfaits de la qualité des fonctionnalités VCR étendues de notre lecteur VoD.

5.4.2 Performances par rapport aux autres implémentations

Ayant à notre disposition l'implémentation HTTP de Microsoft et celle de Real Networks, nous étudions le comportement de notre implémentation par rapport à celles-ci. Les points que nous avons considérés importants et que nous pouvons mesurer sont :

la latence avant le début de la visualisation d'un objet ; selon la méthode de

²¹ Il s'agit d'un hôtel cinq étoiles, "Emirates Tower", avec 500 chambres de lux connectées au système VoD.

compression, les lecteurs multimédias ont besoin de plusieurs informations pour décoder l'objet; ces informations sont situées à des endroits différents dans l'objet; récupérer ces informations pose un vrai problème pour une implémentation HTTP

la latence moyenne pour une commande seek

La Figure 60 présente les temps de latence avant le début de la visualisation d'un objet multimédia pour les implémentations étudiées. Nous observons que notre stratégie de transfert par blocs offre des meilleures performances que les autres implémentations HTTP, que nous pouvons qualifier de "standard". Les modules de communication de WMP et RV sont optimisés pour le transport des données bas-débit sur Internet, et moins adaptés pour une communication haut-débit sur Intranet. A la différence de ces derniers, notre implémentation est optimisée pour une communication Intranet à haut-débit.

Figure 60 Temps de latence avant le début de la visualisation d'un objet de plusieurs implémentations HTTP

	WMP	RV	WMP + module transfert par blocs	RV + module transfert par blocs
MPEG1	2 sec	5 sec	1 sec	1 sec
MP3	2 sec	3 sec	1 sec	1 sec
MPEG2	3 sec	10 sec	1 sec	1 sec
MPEG4	5 sec	10 sec	2 sec	2 sec

La commande seek peut avoir des latences très importantes pour les implémentations classiques de HTTP dans WMP et RV. En fonction de la taille de l'objet et de la position du seek, la latence peut arriver à des valeurs d'ordre de minutes ou même plus. Dans la Figure 61 nous présentons les temps moyens de latence pour un seek avec l'implémentation "transfert par blocs". Les sauts sont réalisés d'une manière quasi instantanée, condition importante pour un lecteur VoD. La différence de performance entre les modules pour WMP et RV est déterminée par les performances meilleures de WMP sur le système d'exploitation Windows.

Figure 61 Temps de latence de la commande "seek" pour plusieurs implémentations de HTTP

	WMP + module transfert par blocs	RV + module transfert par blocs
MPEG1	0.5 sec	1 sec
MP3	0.5 sec	1 sec
MPEG2	0.5 sec	1 sec
MPEG4	1 sec	2 sec

5.5 Résultats expérimentaux - les performances de plusieurs serveurs Web dans la VoD

Nous avons montré que l'utilisation du HTTP avec une stratégie de transfert par blocs ne pénalise pas les consommateurs de VoD en ce qui concerne les fonctionnalités VCR. Mais l'autre problème qui se pose est de savoir si un serveur HTTP a la capacité de servir un nombre de clients comparable à un serveur VoD.

Pour cette raison nous avons réalisé une série de tests sur deux serveurs Web très connus sur Internet : Apache ([apache]) et Internet Information Server ([iis]).

5.5.1 Plate-forme de tests

La plate-forme de tests utilisée est la même que celle utilisée pour l'évaluation de performances d'un serveur VoD (Chapitre 4, Figure 44). Un serveur PC de haute performance est connecté sur un réseau GigabitEthernet. Les machines clientes, au nombre de 24, se trouvent sur le même réseau et sont capables de jouer plusieurs flux multimédias en parallèle.

5.5.2 Méthodologie de tests

Pour mesurer la performance d'un serveur Web dans la VoD, nous utilisons la méthodologie décrite dans le chapitre précédent. Sur les machines clientes nous démarrons des simulateurs de lecteurs VoD/HTTP avec une stratégie de transfert par blocs, qui calculent la QoP pour chaque flux. Les simulateurs ont à leur disposition plusieurs objets multimédias de taille importante, utilisant approximativement 80 % de l'espace disponible sur le serveur. On utilise une grande quantité de données pour charger au maximum le système de stockage du serveur et pour éviter que le système d'exploitation n'utilise trop son cache pour améliorer ses performances.

Notre objectif dans ces expériences est de trouver les limites de ce système en termes de flux envoyés correctement. Au fur et à mesure que le temps passe, on augmente le nombre de flux jusqu'à la dégradation de la QoP. De cette manière nous obtenons le nombre maximal de clients VoD supportés par notre configuration, en fonction du débit des flux multimédias utilisés.

5.5.3 Performances

Les résumés de tests faits sur ces deux serveurs sont présentés dans la Figure 62. Nous avons mis en évidence le débit total pour le nombre maximal de clients satisfaits, en fonction du débit des différents types de flux multimédias: de MPEG1 à 1.5 Mbps jusqu'au MPEG2 à 8 Mbps.

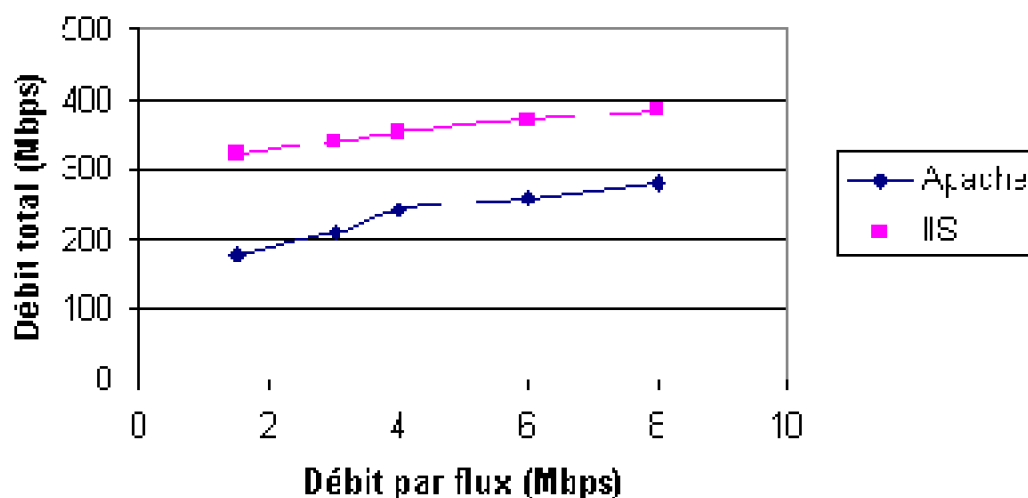


Figure 62 Les performances de deux serveurs Web (Apache et IIS) dans la VoD. Les courbes montrent le débit maximal par rapport au débit d'un flux

On observe que les performances de IIS dans la VoD sont très proches d'un serveur VoD natif (AMS, voir Chapitre 4), et il se comporte même mieux pour des flux à bas débit (MPEG1 à 1.5 Mbps).

La différence de performance entre Apache et IIS sur WindowsNT est déterminée par l'utilisation d'une API spécifique à ce système pour le transfert disque-réseau (*TransmitFile*) par IIS. Apache utilise les méthodes standard de travail avec le système de stockage et avec le réseau (*Read*, *Send*), méthodes qui ne sont pas la meilleure solution pour WindowsNT ([msdn]) mais qui restent portables sur d'autres systèmes d'exploitation.

5.6 Conclusion

La méthode d'utilisation de HTTP proposée dans ce chapitre permet une utilisation efficace dans la VoD des serveurs Web existants. Cette méthode a été validée avec les principaux lecteurs multimédias du marché ([ms-wmp], [rp-rv]), dans lesquels nous avons modifié l'implémentation du protocole HTTP. Les tests faits sur des serveurs Apache et IIS permettent de conclure que l'on peut utiliser avec de bons résultats les serveurs Web existants pour diffuser des flux multimédias.

Par rapport aux serveurs VoD, les résultats nous montrent que la différence entre les performances de ces deux types de serveurs (Web et VoD) est négligeable.

En essayant d'obtenir un rendement maximal d'un serveur Web (utilisé comme serveur VoD) sur une machine, nous avons mis en place un serveur Web optimisé pour les requêtes multimédia (Chapitre 6). Les performances d'un tel serveur, en fonction des facteurs que nous avons considérés importants, apportent un plus de performance aux

serveurs Web standards, comme nous le montrerons dans le prochain chapitre.

Chapitre 6. Design et implémentation d'un serveur Web optimisé pour la VoD

6.1 Introduction

Comme nous l'avons montré dans le Chapitre 3, les requêtes multimédia adressées à un serveur VoD ont des caractéristiques particulières. Le serveur doit fournir un débit soutenu pour chaque flux, afin d'assurer la disponibilité des données côté client. Dès que le nombre de flux concurrents augmentent, les performances du serveur VoD sont fortement mises en cause et plus particulièrement son système de stockage.

Nous avons étudié de près les performances d'un système de stockage constitué de plusieurs disques et d'une carte RAID. Un modèle décrivant ces performances est décrit dans la section 6.2. En utilisant ce modèle nous étudions l'efficacité de plusieurs serveurs VoD (section 6.6). On observe que les serveurs VoD ont des performances fortement différentes sur un même système physique, principalement selon la stratégie de travail avec leurs systèmes de stockage.

En essayant d'obtenir un rendement maximal d'un serveur Web (utilisé comme serveur VoD) sur une machine, nous avons mis en place un serveur Web optimisé pour les requêtes multimédia (les sections 6.3- 6.5). Les performances d'un tel serveur, en

fonction des facteurs que nous considérons importants, sont décrites à la fin de ce chapitre.

Finalement nous arrivons à la conclusion que notre proposition de serveur Web fournit des performances comparables avec les meilleures serveurs VoD existants. Les impacts commerciaux sont importants : il n'est pas forcément nécessaire d'acheter un système onéreux et très spécialisé pour faire de la vidéo à la demande sur un réseau Intranet, dans la mesure où un serveur Web existant suffit.

6.2 Les performances d'un système matériel VoD

Un système matériel VoD est influencé par les performances des ses trois sous-systèmes principaux : le système de stockage, le système de calcul et le réseau (Figure 63).

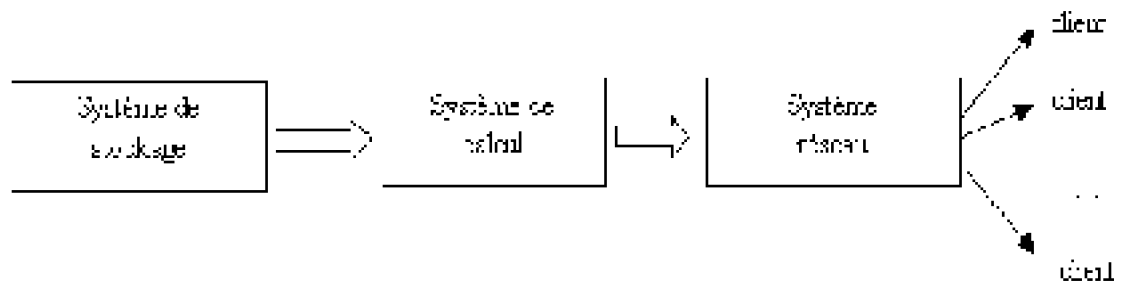


Figure 63 Les trois sous-systèmes influençant les performances d'un système VoD

Si on mesure la performance d'un système VoD par le débit utile à destination des clients, on respecte la relation :

$perf(VoD) \leq \min \{ perf(stockage), perf(calcul), perf(reseau) \}$	(5)
$perf(VoD) \leq \min \{ perf(stockage), perf(calcul), perf(reseau) \}$	(5)

La performance maximum d'un système VoD s'obtient quand le système atteint le minimum des performances maximales d'un sous-système qui le compose (réseau, calcul ou stockage).

Nous nous situons dans le cas où c'est le système de stockage qui limite les performances globales. Les processeurs sont de plus en plus puissants, les cartes réseau Gigabit sont assez répandues, seules les disques et les cartes RAID n'évoluent pas aussi vite au niveau performance. Notre objectif dans cette partie est de modéliser les performances d'un système de stockage composé de plusieurs disques et d'une carte RAID en utilisant leurs caractéristiques techniques.

Plus exactement, en connaissant les détails techniques d'un disque dur et d'une carte RAID nous estimons quelles sont les performances maximales d'un système de stockage

dans un contexte de VoD.

6.2.1 Architecture d'un disque dur

L'architecture physique simplifiée d'un disque dur est présentée en Figure 64. Un disque contient plusieurs plateaux. Sur chaque plateau nous avons un certain nombre de pistes circulaires (*tracks*), et sur chaque piste un nombre de secteurs. Les pistes circulaires situées à la même distance de l'axe de rotation (une piste sur chaque plateau) forment un cylindre. Les informations sont emmagasinées dans les secteurs (par exemple 512 octets par secteur). L'accès aux secteurs se fait à l'aide d'un certain nombre de tête de lecture/écriture, situées sur un même bras.

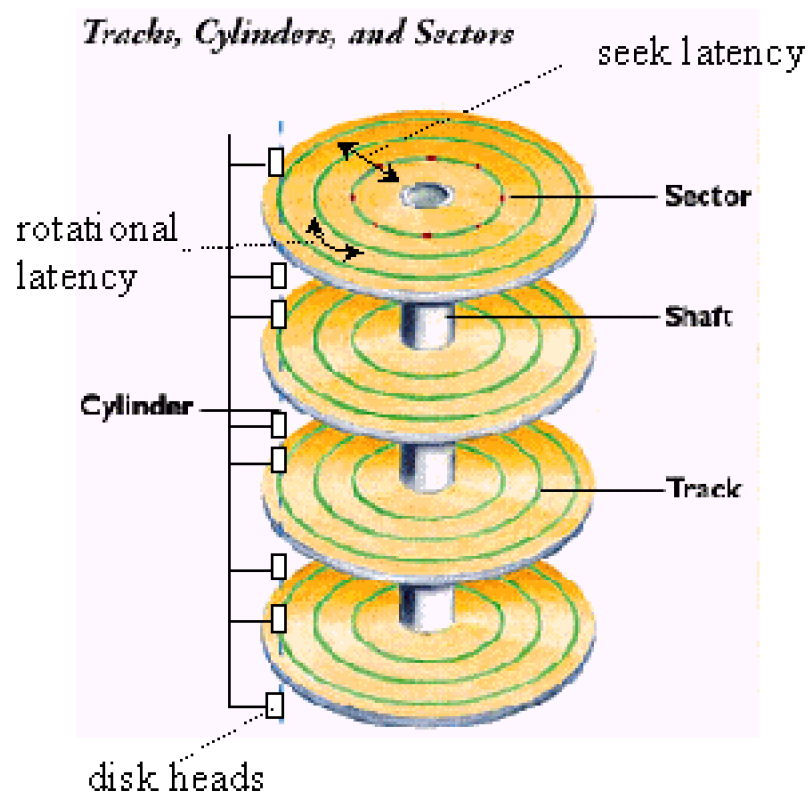


Figure 64 L'architecture physique très simplifiée d'un disque dur ([quantum])

Une opération de lecture consiste en le positionnement des têtes de lecture/écriture du disque sur le cylindre et les secteurs désirés par un mouvement de translation et de rotation.

Les performances d'un disque dur dépendent d'un grand nombre de paramètres ([storagereview] présente 36 paramètres influençant le débit d'un disque dur). Il existe plusieurs modèles pour les disques SCSI avec différents niveaux d'abstraction, tenant compte de plus ou moins de paramètres ([rangan92], [lee97]). Nous souhaitons modéliser d'une manière simple deux paramètres plus généraux: *le débit soutenu* du disque et *le temps d'accès* à un bloc de données.

Le *débit soutenu* d'un disque dur est une mesure de la quantité de données transférée depuis le disque vers l'extérieur dans des conditions particulières (lecture d'un certain nombre de blocs consécutifs, etc.). En réalité le débit dépend aussi de la position des blocs à l'intérieur du disque (sur des parties plus rapides ou plus lentes). Nous prenons en compte les blocs situés sur la partie la plus lente du disque. Les disques avec lesquels nous avons travaillé ont un taux de transfert situé entre 12 et 20 MBytes/s ([quantum]).

Le *temps d'accès* à un bloc (*Access Time*) est une mesure dépendant de plusieurs paramètres :

Access Time = Command Overhead Time + Seek Time + Settle Time + Latency	(6)
Access Time = Command Overhead Time + Seek Time + Settle Time + Latency	(6)

Le temps de réaction du disque (*Command Overhead Time*) représente le temps passé entre la réception d'une commande et le début d'une action physique. Généralement il a une valeur assez négligeable par rapport à d'autres paramètres, située autour de 0.5 ms.

Le temps d'un saut (*Seek Time*) mesure la durée nécessaire pour que les têtes lecture/écriture du disque changent de cylindres. Comme le temps nécessaire pour changer la position d'une tête de lecture dépend aussi de la position des cylindres, on mesure la moyenne du temps d'un saut entre des cylindres avec des positions aléatoires, qu'on appelle *le temps moyen de saut*. Les valeurs de ce paramètre se situent entre 6 et 12 millisecondes pour les disques durs de nos jours.

Le temps de stabilisation (*Settle Time*) mesure le temps nécessaire pour stabiliser la tête de lecture avant de commencer la lecture. La valeur est située autour de 0.1 ms.

La latence (*Latency*) mesure le temps nécessaire pour que la tête de lecture trouve le bon secteur sur un cylindre (voir la Figure 64). Si nous connaissons le nombre de rotations que le disque peut faire par minute (RPM), la latence²² est donnée par la relation :

$latence = \frac{1}{RPM/60} * 0.5 * 1000 = \frac{30000}{RPM}$	(7)
$latence = \frac{1}{RPM/60} * 0.5 * 1000 = \frac{30000}{RPM}$	(7)

Pour un disque à 7200 rotations par minute (7200 rpm), la latence est de 4.2 ms. Dans la Figure 65 nous montrons les caractéristiques d'un disque dur Quantum Atlas, à partir de ses spécifications techniques ([quantum-atlas]).

²² la latence est mesuré en millisecons (ms)

Figure 65 Les caractéristiques d'un disque dur Quantum Atlas V

Disque Quantum Atlas V	
Paramètres fournis par le constructeur :	
Capacité	18.2 GBytes
Débit soutenu	17 MBytes/s
Command Overhead Time	0.5 ms
Temps moyen de saut (seek time)	6.3 ms
Settle Time	0.1 ms
RPM	7200 rpm
Paramètres déduits :	
Latency	4.2 ms
Temps d'accès	0.5+6.3+0.1+4.2 = 11.1 ms

6.2.2 Estimation de performances d'un disque

On suppose que l'on a n fichiers ouverts sur un disque (l'équivalent de n flux distincts sur un serveur VoD avec un seul système de stockage). L'accès à ces fichiers se fait par transfert de blocs de taille fixe (noté B). La tête de lecture/écriture effectue une lecture d'un bloc de taille B avant de sauter à un bloc d'un autre fichier.

Le temps de transfert d'un bloc de taille B est B/T , où T est le débit soutenu du disque vers l'extérieur. Le temps nécessaire pour transférer n blocs de taille B situés dans des fichiers différents est alors $n*(B/T)+n*S$, où S est le temps d'accès à un bloc, donné par la formule (6).

Donc, le débit réel d'un disque avec n flux en lecture est :

$T_R(n, B, S, T) = \frac{n*B}{n*\frac{B}{T} + n*S} = \frac{B}{\frac{B}{T} + S}$	(8)
$T_R(n, B, S, T) = \frac{n*B}{n*\frac{B}{T} + n*S} = \frac{B}{\frac{B}{T} + S}$	(8)

Ou, dans une forme plus compactée :

$T_R(n, B, S, T) = \frac{1}{\frac{1}{T} + \frac{S}{B}}$	(9)
$T_R(n, B, S, T) = \frac{1}{\frac{1}{T} + \frac{S}{B}}$	(9)

Cette formule (9) est approximative, en raison de la multitude des facteurs qui déterminent les performances d'un disque. Nous avons utilisé les facteurs que nous les avons considérés essentiels pour un système VoD (T et S).

6.2.3 Estimation des performances d'un système de stockage RAID

Une architecture simplifiée d'un serveur contenant un système de stockage RAID est présentée dans la Figure 66. La carte RAID est connectée sur le bus PCI du serveur et les k disques SCSI partagent le même bus SCSI de la carte RAID. La carte RAID dispose d'une mémoire vive (cache), permettant un stockage temporaire des données.

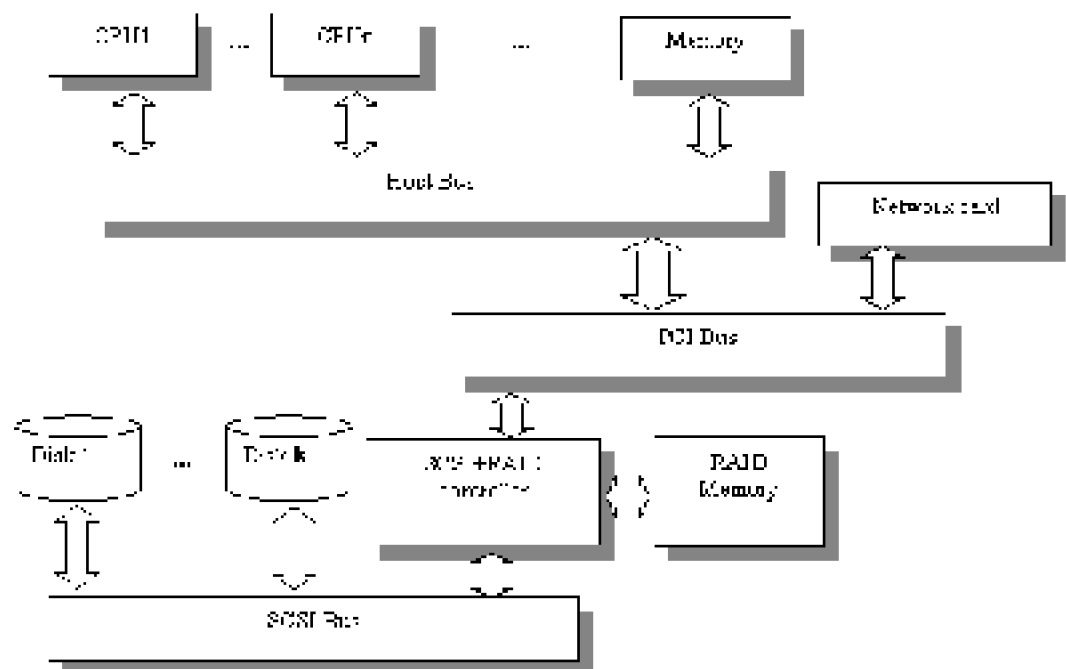


Figure 66 Configuration d'un système de stockage RAID connecté à un ordinateur

Si notre système de stockage contient k disques organisés en un système RAID0, le débit réel de n flux en lecture est k fois supérieur au débit d'un disque dur (Chapitre 3) :

$T_R(n, k, B, S, T) = \frac{k}{\frac{1}{T} + \frac{S}{B}}$	(10)
$T_R(n, k, B, S, T) = \frac{k}{\frac{1}{T} + \frac{S}{B}}$	(10)

Généralement les constructeurs des cartes RAID implémentent différents algorithmes pour la gestion du cache de la carte. Un type d'algorithme assez utilisé ([dell-hdd], [compaq]) consiste à lire à l'avance une certaine quantité de données pour chaque lecture pour limiter le nombre d'opérations de saut sur les disques.

De cette manière, pour la lecture d'un bloc de taille B , la carte va lire une quantité de données de taille $4*B$ par exemple. Les données demandées (de taille B) seront fournies et le reste (de taille $3*B$) est gardé dans la mémoire cache de la carte (RAID Memory) pour des requêtes ultérieures (l'accès dans le cache de la carte est beaucoup plus rapide que sur les disques).

Si le système RAID est organisé autour d'une carte physique RAID avec un cache de taille M_{RAID} , la taille maximale d'un bloc à la valeur M_{RAID} / n . Dans ces conditions le débit maximal du système est approximé par :

$T_R(n, k, M_{RAID}, S, T) = \frac{k}{\frac{1}{T} + \frac{n * S}{M_{RAID}}}$	(11)
$T_R(n, k, M_{RAID}, S, T) = \frac{k}{\frac{1}{T} + \frac{n * S}{M_{RAID}}}$	(11)

Dans la formule (11) la taille du bloc de transfert (B) n'intervient pas, elle est calculée par la carte RAID selon un algorithme spécifique.

6.2.3.1 StorePerf: benchmark VoD pour le système de stockage

Pour vérifier les performances réelles d'un système de stockage, nous avons mis au point un logiciel (StorePerf) capable de calculer ces performances en chargeant au maximum le

système. Pour déterminer les résultats pertinents pour la VoD, StorePerf crée sur le système de stockage un jeu de fichiers de taille importante (minimum 256 MB par fichier, en fonction de l'espace de stockage). Pour chaque fichier StorePerf démarre un fil d'exécution (thread) qui exécute des lectures avec une certaine taille de bloc et à un débit préétabli. Cette simulation de serveur VoD (en mode lecture seule sur le système de stockage) produit une série des résultats que l'on utilise pour prévoir les performances possibles d'un serveur VoD. En quelque sorte, StorePerf est la version appliquée de (11).

Le StorePerf s'exécute dans la ligne des commandes de Windows (CMD.EXE), et il est démarré avec la syntaxe :

```
· storeperf [-Rf ReadFilesize] [-Rc ReadClients] [-Rb ReadBlocksize] [-Rt ReadThroughput] [-Wf WriteFilesize] [-Wc WriteClients] [-Wb WriteBlocksize] [-Wt WriteThroughput] [-o outputFile] [-t time] [-g graphUpdate] [-n graphName] [-a AccesType]
```

où la signification des paramètres est :

- ReadFilesize : la taille d'un fichier en lecture ; il sera créé par le logiciel,
- ReadClients : le nombre de clients en lecture (chaque client aura son propre fichier),
- ReadBlocksize : la taille d'un bloc dans chaque opération de lecture,
- ReadThroughput : le débit contrôlé pour un client en lecture (mesuré en octets par seconde) ; s'il a la valeur 0 le logiciel essaye d'obtenir le débit maximal possible,
- WriteFilesize : la taille d'un fichier en écriture ; il sera créé par le logiciel et rempli d'une manière cyclique : chaque fois qu'il est plein les données sont écrites depuis le début du fichier,
- WriteClients : le nombre de clients en écriture (chaque client aura son propre fichier),
- WriteBlocksize : la taille d'un bloc dans chaque opération d'écriture,
- WriteThroughput : le débit contrôlé pour un client en écriture (mesuré en bps),
- outputFile : le fichier qui va contenir les résultats,
- time : la durée d'exécution,
-

graphUpdate : la période de rafraîchissement du graphique,

graphName : le nom du graphique contenant les résultats,

AccesType : l'API Windows utilisée pour accéder aux données (CFile::Read, mmioRead, ReadFile ou _read).

Pour calculer les performances maximales en lecture d'un système de stockage pour 10 clients concurrents sur des fichiers de 100 MBytes avec des lectures de blocs de 8 KBytes à la fois on utilise la syntaxe :

```
StorePerf -Rf 100 -Rc 10 -Rb 8192 -Rt 0 -o "test.txt" -t 600 -g 5
```

6.2.3.2 Exemple

Nous calculons les performances possibles d'un système de stockage constitué de quatre disques Quantum Atlas V connectés en RAID0 sur une carte RAID PERC2DC avec 128 MB cache. Les caractéristiques d'un tel disque sont données dans la Figure 65.

En fonction du nombre de flux, les performances de transfert en lecture estimées avec la formule (11) par rapport aux résultats obtenus avec StorePerf sont présentées en Figure 67. On observe que la formule (11) donne une estimation environ 20% plus grande que les données réelles obtenues. Cet écart s'explique par le modèle simplifié utilisé pour la modélisation des performances d'un système de stockage.

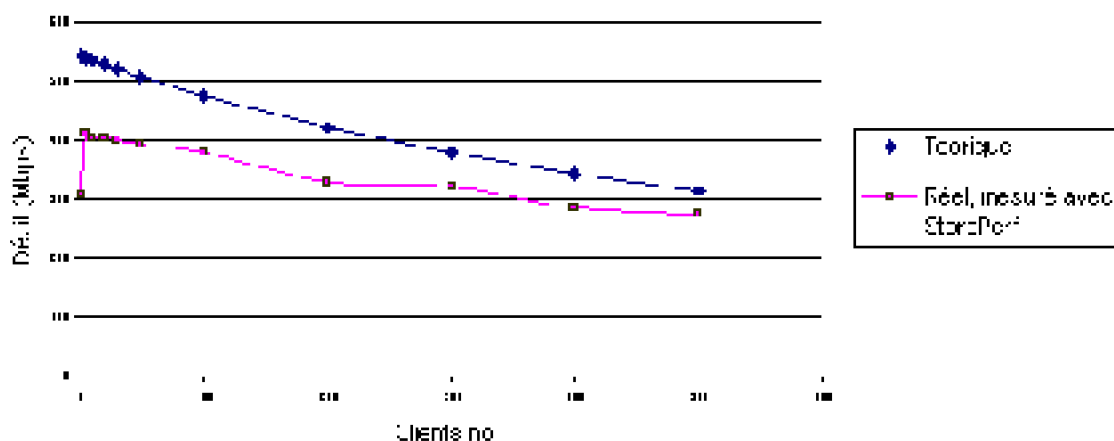


Figure 67 Les performances d'un système de stockage constitué de 4 disques en RAID0

6.2.3.3 Autres facteurs influençant les performances du système de stockage

Un autre facteur très important qui influence les performances du système de stockage est la taille élémentaire d'allocation sur le système de stockage. Sur le système d'exploitation Windows on peut avoir plusieurs tailles possibles²³, en fonction du système de fichiers

utilisé :

Figure 68 Les tailles d'allocation élémentaire sur le système de stockage accepté par les systèmes de fichiers du Windows: FAT, FAT32, NTFS

	FAT	FAT32	NTFS
512 bytes	oui	oui	oui
1024 bytes	oui	oui	oui
2048 bytes	oui	oui	oui
4096 bytes	oui	oui	oui
8192 bytes	oui	oui	oui
16 KB	oui	oui	oui
32 KB	oui	oui	oui
64 KB	oui	oui	oui
128 KB (secteur > ;512 bytes)	oui	oui	non
256 KB (secteur > ;512 bytes)	oui	oui	non

L'effet de la taille élémentaire d'allocation sur le performances d'un système de stockage composé d'une carte RAID et 6 HDD SCSI à 7200 RPM donnent les résultats décrits à la Figure 69 :

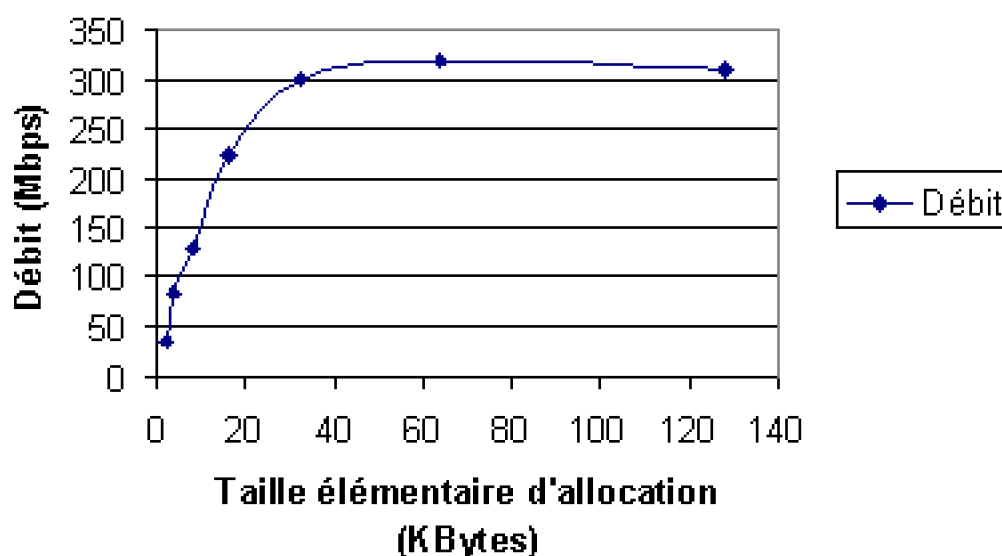


Figure 69 Performances d'un système de stockage par rapport à la taille de l'unité élémentaire d'allocation

On observe que cette taille à un impact très important sur les performances en lecture du système. Si pour un formatage avec une allocation élémentaire de 2 KB nous

²³ Sur Windows, pour spécifier la taille élémentaire d'allocation (*default allocation unit size*) sur un système de stockage on utilise la commande "format /A:taille"

obtenons un débit maximal de 37 Mbps, pour une taille de 64 KB le débit maximal s'élève à 320 Mbps. Dans nos travaux nous avons utilisé des valeurs qui ne pénalisent pas les performances I/O du système (64 KB comme unité élémentaire pour le formatage).

Suite à ces considérations, nous souhaitons construire un serveur Web capable de fournir le maximum de performances pour un nombre donné de clients et un débit maximal préétabli.

6.3 L'architecture du serveur

Les expériences menées sur le système de stockage nous ont conduit à développer notre propre serveur Web spécialisé dans la VoD. Ce serveur est nommé MMServer.

Le serveur Web spécialisé dans la VoD que nous proposons est basé sur le modèle du serveur Apache (version WinNT). Pour chaque nouveau client le serveur crée un nouveau thread (fil d'exécution) qui répond aux requêtes du client. Dès que le client se déconnecte, le thread est détruit.

Le serveur communique avec les clients en utilisant le protocole HTTP 1.1, ce qui fait que n'importe quel client travaillant avec ce protocole est capable de récupérer l'information située sur le serveur.

Le serveur consiste en un processus qui "écoute" les clients sur le port TCP 80 . Pour chaque client qui se connecte sur ce port, un agent de contrôle d'accès vérifie si le serveur a les ressources nécessaires pour répondre dans des bonnes conditions à ce client et aux clients déjà acceptés. Dans le cas positif, le client est accepté et il va recevoir les données qu'il demande. Dans le cas contraire, la connexion avec le client est coupée.

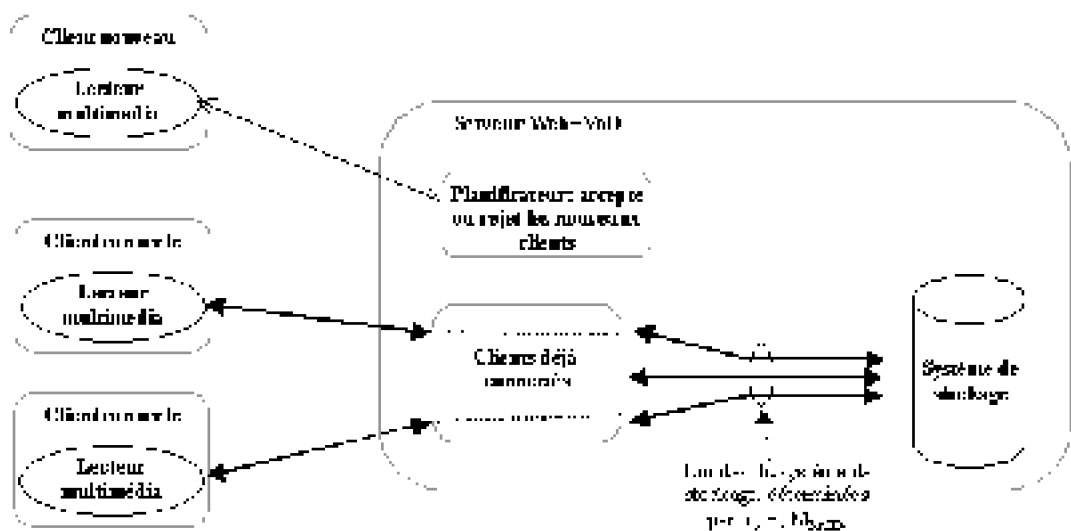


Figure 70 Vue d'ensemble sur l'activité du serveur

La différence par rapport aux autres serveurs Web réside dans le mode de travail de chaque thread, optimisé pour exploiter au maximum le système de stockage dans le

cadre de la stratégie présentée dans le Chapitre 2. Le principe est de limiter le nombre de sauts (seek) sur le système de stockage, en transférant une quantité de données plus grande que celle demandée à la fois par un client.

Le système d'exploitation et la carte RAID utilisent eux-mêmes des stratégies de transfert de données à travers un cache plus ou moins grand, en fonction des caractéristiques de la machine et du système d'exploitation. Mais dans le cas particulier d'un serveur VoD cette stratégie ne donne pas souvent de bons résultats. Cela nous a amenés à proposer une stratégie de cache en alternative à celle du système d'exploitation.

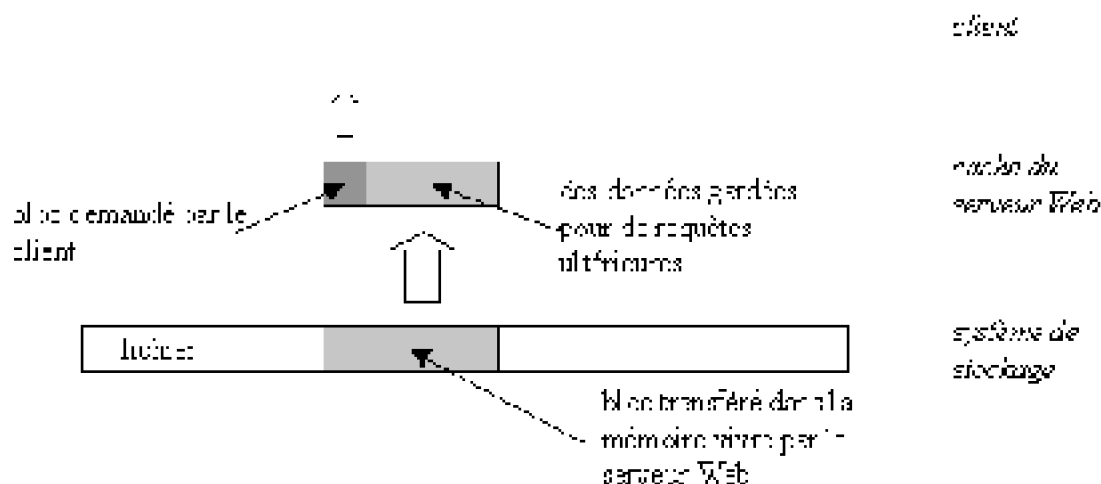


Figure 71 Le principe de travail avec les données de notre serveur : l'utilisation de la mémoire vive du serveur pour améliorer les performances en lecture

Chaque fois que le client demande un bloc, une quantité plus grande des données (contenant le bloc et multiple de la taille d'allocation élémentaire sur le système de fichiers) sont transférées à partir du système de stockage dans la mémoire vive du serveur. A partir de là, le bloc est envoyé au client comme réponse, et les autres données sont gardées pour des requêtes ultérieures.

Tenant compte de la spécificité de la représentation séquentielle des données multimédia, les prochaines requêtes provenant du même client ont de fortes chances d'être les prochains blocs du fichier, donc ils se trouvent dans la mémoire vive du serveur. Les exceptions sont les opérations de saut sur la présentation, par exemple, mais qui représentent une faible proportion des accès.

Bien entendu si le bloc se trouve déjà dans la mémoire vive du serveur, celui-ci sera envoyé comme réponse sans accéder au système de stockage.

D'une telle manière on minimise le nombre de requêtes adressées au système de stockage. Il est vrai que la carte RAID effectuent la même chose, mais à un niveau plus bas. Ce deuxième niveau d'optimisation est utile dans le cas où le serveur dispose d'une quantité de mémoire plus importante que la carte RAID (et c'est le cas actuellement²⁴).

²⁴ La mémoire vive sur les cartes RAID ne dépassent pas 256 MBytes (sur [dell-hdd]), à la différence des ordinateurs qui peuvent utiliser des GB de mémoire vive.

Par exemple, une carte RAID avec 16 MB RAM ne peut pas travailler d'une manière efficace avec plus de 60 clients dans la VoD.

Une autre différence par rapport aux serveurs Web classiques réside dans l'existence d'un agent de contrôle d'accès. Il utilise les résultats (11) pour vérifier à chaque instant l'occupation du serveur. Il représente un filtre nécessaire pour éviter le dépassement des limites du serveur.

6.4 La gestion de la concurrence

6.4.1 Modèle "thread per request"

Comme nous l'avons déjà dit, la stratégie de gestion de la concurrence implémentée dans notre serveur est une de "thread per request", où les requêtes de chaque client seront gérées par un thread. Ce modèle nous permet d'exploiter les architectures physiques multiprocesseur, en distribuant les threads entre les processeurs.

6.4.2 Le contrôle d'admission

6.4.2.1 Contrôle d'une connexion déjà établie

Conformément à la stratégie HTTP de transfert par blocs proposée, chaque client enverra un certain nombre de requêtes au serveur, pour des blocs d'une taille constante. Pour éviter qu'un client ne demande trop de données, le serveur ne répondra qu'avec une certaine latence entre deux requêtes (Figure 72).

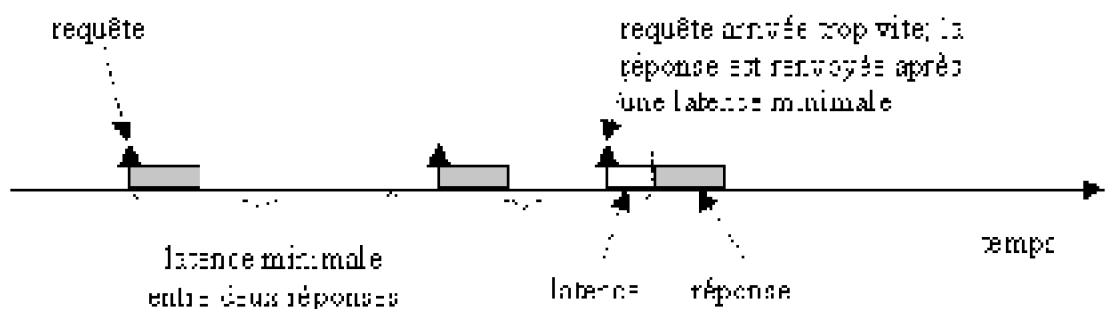


Figure 72 Régularisation du flux de requêtes HTTP sur une connexion. Entre deux requêtes consécutives le serveur garde une latence minimale : intervalle de temps durant lequel on ne donne pas plus d'une réponse.

Ainsi aucun client ne dépassera pas un débit maximal préétabli. En fonction des types et des débits des objets existant sur le serveur, le débit maximal autorisé pour un client peut avoir différentes valeurs.

On suppose que chaque client travaille avec la stratégie décrite dans le Chapitre 5 et

demande des blocs de données de taille constante B_C (Figure 73). Si le serveur sert n clients concurrents, le nombre de blocs transférés par seconde est n_C .

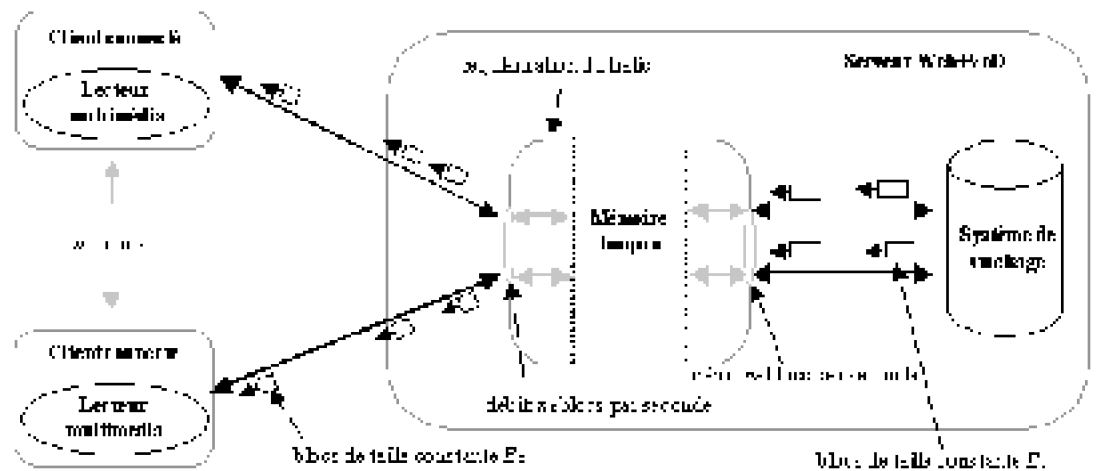


Figure 73 Régularisation du flux de requêtes HTTP au niveau serveur

6.4.2.2 Admission d'un nouveau client

Un problème important qui se pose est : dans quelles conditions un nouveau client peut-il être accepté sans affecter la qualité de perception des autres clients déjà connectés ?

Le serveur transfère en mémoire des blocs de taille B_S à partir du système de stockage. Le nombre maximal de blocs transférés dans une seconde entre le système de stockage et la mémoire du serveur (n_S) respecte la relation (conformément à (11)) :

$n_s * B_s \leq \frac{k}{\frac{1}{T} + \frac{n * S}{M_{RAID}}}$	(12)
$n_s * B_s \leq \frac{k}{\frac{1}{T} + \frac{n * S}{M_{RAID}}}$	(12)

En même temps, le débit des données transférées vers les clients ne dépasse pas le débit des données transférées en mémoire à partir du système de stockage :

$n_c * B_c \leq n_s * B_s$	(13)
$n_c * B_c \leq n_s * B_s$	(13)

Les inégalités (12) et (13) nous conduisent à :

$n_c \leq n_s * \frac{B_s}{B_c} \leq \frac{B_s}{B_c} * \frac{k}{\frac{1}{T} + \frac{n * S}{M_{RAID}}}$	(14)
$n_c \leq n_s * \frac{B_s}{B_c} \leq \frac{B_s}{B_c} * \frac{k}{\frac{1}{T} + \frac{n * S}{M_{RAID}}}$	(14)

En effet le nombre maximal de blocs envoyés par seconde vers les clients ne peut pas dépasser la quantité définie par (14). Dans une implémentation réelle, la limite à une valeur plus restrictive, due aux différentes approximations que nous avons établies pour modéliser le système de stockage.

Donc, un client sera accepté si la nouvelle configuration du système respecte la relation (14).

6.5 Implémentation

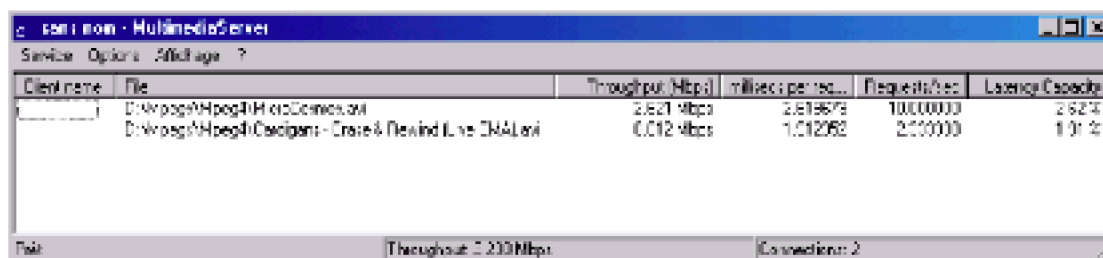
Nous avons implémenté le serveur en utilisant l'environnement MS-Visual C++. La taille du serveur (version binaire) ne dépasse pas 100 KB, mais il offre toutes les fonctionnalités nécessaires pour justifier son utilisation comme serveur VoD.

Au démarrage, le serveur ajoute une icône à la barre d'outils Windows (Figure 74). A travers un menu contextuel on peut accéder à son interface pour visualiser les clients connectés ou pour changer ses paramètres.



Figure 74 L'icône montrant que le serveur est démarré

Pour surveiller l'activité du serveur, une fenêtre permet de visualiser les clients connectés et les fichiers auxquels ils accèdent. Sur chaque connexion un certain nombre d'informations supplémentaires sont fournies, comme le débit et le nombre de requêtes effectuées par seconde.



Client name	File	Throughput (Mbps)	milliseconde...	Requests/sec	License Capacity
	D:\Vopopg\Hpegd\HicrComics.avi	2.821 Mbps	2.819879	10000000	2624
	D:\Vopopg\Hpegd\Casigant - Crise & Newind Live DN4.avi	0.012 Mbps	1.012092	2000000	1014

Stat: Throughput: 2.833Mbps Connections: 2

Figure 75 Visualisation de l'activité du serveur : les connexions ouvertes par les clients et leurs caractéristiques (fichiers, débit, nombre de requêtes par seconde etc.)

Les propriétés du serveur ont été classées en deux catégories : des propriétés générales et des propriétés avancées. Les propriétés générales (Figure 763) sont :

- le port TCP utilisé pour recevoir les connexions HTTP,
- le nom du serveur,
- les types des fichiers visible pour les utilisateurs,
- les répertoires accessible aux clients avec leurs noms (répertoires virtuels)



Figure 76 Propriétés générales du serveur

Le client utilise une interface composée de pages HTML pour accéder au contenu du serveur. Dans le cas où l'interface n'est pas disponible, pour éviter que le client n'introduise l'URL d'un objet en dur, le serveur fournit une page implicite contenant les références vers les objets disponibles (Figure 77).

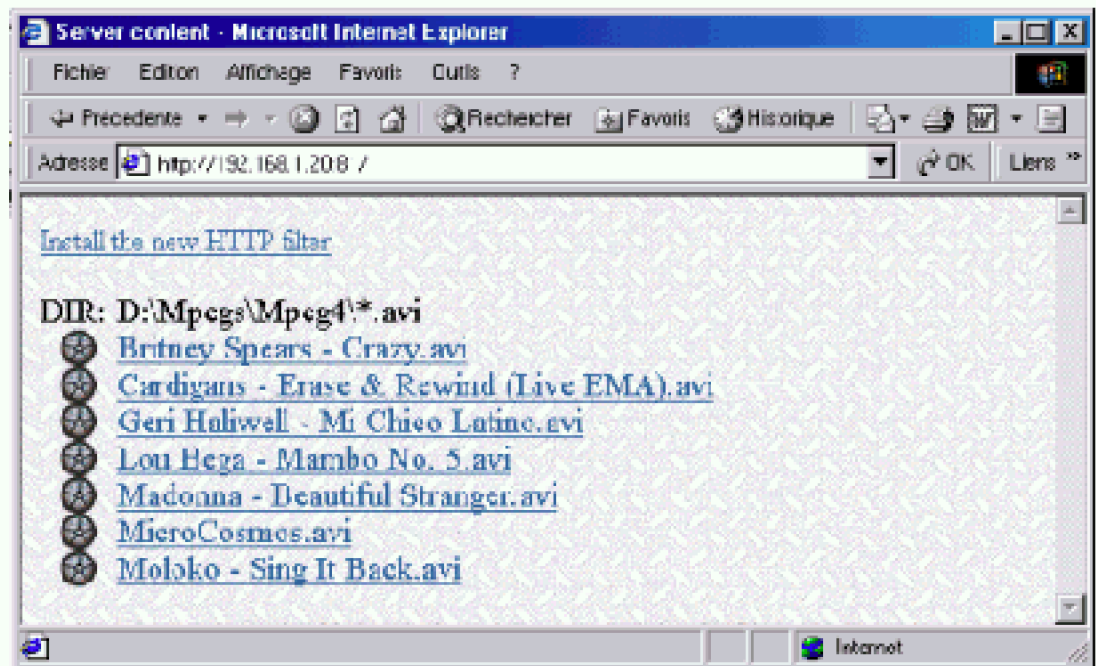


Figure 77 Page implicite fournie par le serveur avec le catalogue des objets disponibles

Les propriétés avancées du serveur (Figure 78) concernent le paramétrage pour l'agent de contrôle d'accès du serveur (QoS), la connexion réseau avec le client (TCP) et l'accès au système de stockage (MMIO).

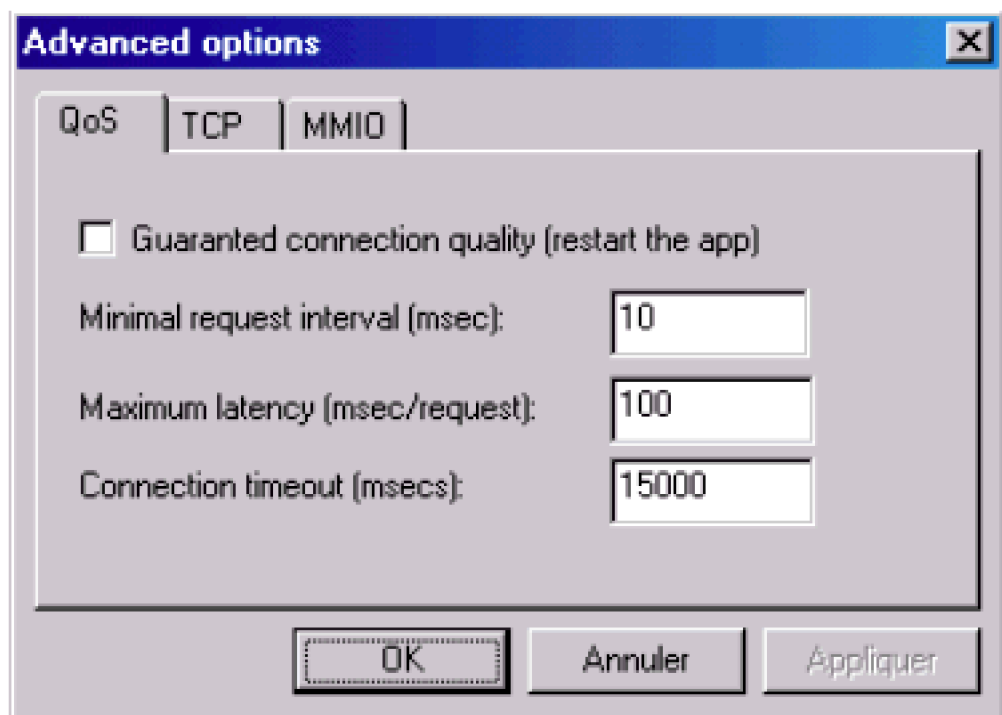


Figure 78 Propriétés avancées du serveur. Paramétrage pour l'agent de contrôle d'accès du serveur (QoS), la connexion réseau avec le client (TCP) et l'accès au système de

stockage (MMIO)

Dans le panel QoS on peut configurer la latence minimale entre deux requêtes, la latence maximale pour répondre à une requête et le temps maximal d'inactivité sur une connexion avant de déconnecter le client HTTP²⁵. Le panel TCP permet de configurer le protocole TCP (taille de la mémoire tampon etc.). Le panel MMIO (MultiMedia Input/Output) permet de configurer la taille d'un bloc pour accéder au système de stockage et la taille de la mémoire tampon utilisée par le serveur pour chaque connexion.

L'implémentation du serveur utilise l'API Windows NT pour une meilleure performance dans le transfert d'un bloc se trouvant sur le système de stockage (la méthode TransmitFile, voir [msdn]). Cette méthode permet de lire et d'envoyer directement sur réseau un bloc se trouvant dans le cache du système de fichiers Windows NT. Cette opération s'effectue d'une manière asynchrone, permettant son utilisation concurrente à l'intérieur de plusieurs threads.

6.6 Etude de la performance

Un pas important dans l'étude de notre serveur Web consiste dans la mesure de performances de notre serveur par rapport à d'autres serveurs Web ou VoD. Les tests que nous avons effectués utilisent la même architecture physique et la même stratégie que ceux détaillés dans le Chapitre 4.

6.6.1 Par rapport à d'autres serveurs Web

Nous avons rapporté les performances de notre serveur MMServer dans la VoD à Apache et IIS. L'effet d'avoir optimisé MMServer pour les accès multimédia nous permet d'obtenir des performances accrues par rapport à un serveur Web standard, comme le montrent les résultats présentés en Figure 79.

²⁵ Nous précisons que la déconnexion d'un client HTTP n'a pas d'effet sur le contenu affiché par le lecteur multimédia du client. Le raison de cette coupure est l'économie de ressources. Dès que le client commence à effectuer des commandes VCR, le lecteur se reconnecte au serveur.

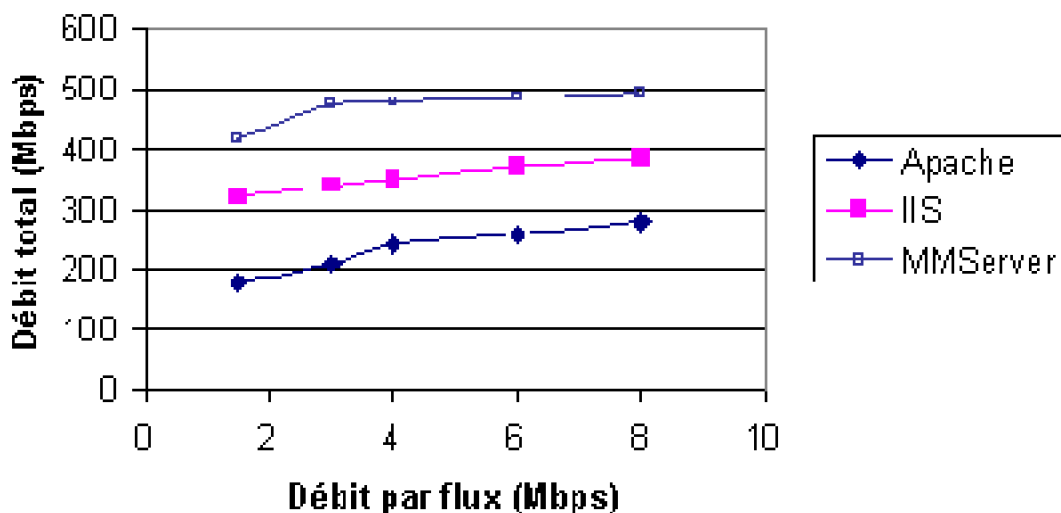


Figure 79 Comparaison entre les performances en VoD de Apache, IIS et MMServer

6.6.2 Par rapport à d'autres serveurs VoD

Le serveur MMServer nous permet même d'obtenir plus de performance qu'un serveur VoD spécialisé (AMS de CSTI). Pour les différents tests effectués MMServer peut servir avec 25% plus de clients par rapport à AMS (Figure 80), tout en gardant la qualité demandée par les clients.

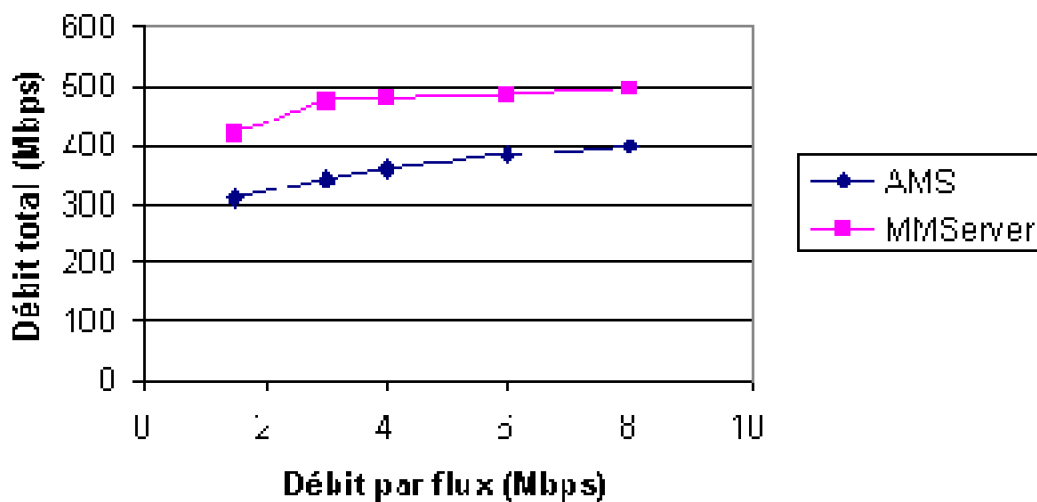


Figure 80 Comparaison entre les performances en VoD de AMS et MMServer

6.7 Conclusion

Les serveurs Web actuels ne sont pas optimisés pour une utilisation intensive en multimédia, même si leur performance avec une utilisation adéquate de HTTP est de même ordre que les performances de serveurs VoD natifs. Ils ne gèrent pas les connexions en termes de qualité de perception, et dans le cas où le serveur serait surchargé toutes les connexions existantes peuvent être affectées.

Pour résoudre ce problème, nous avons mis au point un serveur Web basé sur le modèle d'un serveur Apache, mais implémentant aussi une politique de contrôle d'admission et une politique de travail avec le système de stockage optimisée pour les caractéristiques particulières des requêtes multimédias (débit élevé, requêtes des blocs des données d'un même fichier de manière séquentielle).

Les tests effectués montrent qu'une telle solution est très performante dans un contexte VoD, donnant un meilleur rendement sur la même configuration physique que d'autres serveurs VoD (notamment celui de CSTI), tout en gardant une bonne qualité de l'interaction client-serveur.

Un facteur que nous n'avons pas pris en compte est la fragmentation du système de fichiers. En réalité ce facteur n'influence les performances du système que dans le cas où il a des valeurs élevées. Nous supposons que cet événement ne se produit pas, l'administrateur d'un tel système ayant comme charge d'éviter une fragmentation importante sur le système de stockage.

Chapitre 7. Conclusions et Perspectives

7.1 Résultats

Nous avons analysé dans cette thèse *une stratégie d'utilisation de HTTP* permettant une utilisation efficiente de ce protocole pour la VoD. Cette stratégie a permis le développement d'un système VoD simple et efficace, articulé autour de serveurs Web existants utilisés comme serveurs VoD.

Nous avons de plus développé une méthodologie d'évaluation de performances (*benchmarking*) permettant de trouver les performances réelles des différents serveurs en termes de nombre d'utilisateurs satisfaits. Cette méthodologie a été utilisée pour mesurer l'efficacité de notre solution et la comparer aux solutions concurrentes.

Enfin, nous avons mis en place une *politique de contrôle d'admission* et différentes *politiques d'utilisation du système de stockage* pour optimiser les performances d'un serveur Web. Ces améliorations nous ont permis d'atteindre des performances proches des limites matérielles d'un système VoD.

7.2 Limites

Les serveurs Web en général et le nôtre en particulier autorisent de bonnes performances lorsqu'ils fonctionnent comme des serveurs VoD sur des systèmes Intranet. Les mêmes performances sont cependant difficiles à atteindre lorsque le serveur Web fonctionne sur Internet. La priorité moins élevée des paquets TCP par rapport aux paquets UDP sur certains routeurs peut ralentir l'échange d'informations entre le serveur Web et ses clients.

7.3 Positionnement par rapport à d'autres résultats

7.3.1 Stratégie d'utilisation de HTTP

La stratégie proposée d'utilisation de HTTP pour la multimédia est originale. D'autres stratégies "push" ou "pull" ont été étudiées pour l'utilisation des différents protocoles dans la VoD, comme celles présentées dans [ghandeharizadeh98], [bonhomme00], [oracle], [martin96], [rn-http], [ms-http]. Mais aucune de ces études n'a permis la réalisation d'une plate-forme VoD viable basée sur le protocole HTTP.

7.3.2 Méthodologie pour la mesure des performances d'un serveur VoD

L'état de l'art du domaine VoD ne mentionne pas de stratégies de tests spécifiques pour mesurer les limites d'un système VoD. Un certain nombre des projets de recherche présentent des résultats sans avoir une justification rigoureuse. Différents systèmes commerciaux évaluent les performances attendues d'un système VoD en fonction de la configuration hardware utilisée, mais sans présenter de justification concrète des estimations proposées.

La *qualité de perception* est définie en utilisant une définition similaire présentée dans [ghinea98]. A la différence de cet article, qui étudie l'impact de la qualité de l'image sur l'utilisateur, nous nous intéressons plus à trouver une équivalence de ce facteur avec des paramètres du serveur.

7.3.3 Estimation des performances d'un système matériel VoD

Les performances d'un système de stockage sont largement étudiées dans des publications comme: [chervenak95]. Le modèle que nous proposons utilise la description technique d'un système de stockage présenté dans [dell-hdd] et [storagereview], et essaie de se situer comme alternative aux modèles [rangan92], [chen99].

7.3.4 Conception d'un serveur Web optimisé pour le multimédia

Le serveur que nous avons implémenté utilise comme référence les deux serveurs Web

plus répandus sur Internet : Apache et Internet Information Server. Nous avons simplifié au maximum les diverses fonctionnalités de ces serveurs, éliminant toutes celles qui ne concernaient pas la VoD. Nous avons par contre ajouté les optimisations nécessaires pour exploiter au maximum une architecture physique pour la VoD.

7.4 Perspectives industrielles

Le prix d'un serveur VoD est assez élevé. Une solution basée sur des serveurs Web est une alternative intéressante, surtout si l'on tient compte de la grande disponibilité des serveurs Web gratuits. Les résultats obtenus ont permis une réduction importante du coût du serveur VoD commercialisé par CSTI, l'entreprise qui a financé partialement nos travaux.

Nous avons même montré (fin 2000) la supériorité de notre serveur par rapport à la solution CSTI. Plus exactement, nous avons montré que l'utilisation intelligente d'un serveur Web sur un PC avec un système de stockage performant permettait d'obtenir des performances supérieures à celles d'une machine dédiée à la VoD (le serveur AMS de CSTI).

Ces résultats permettent d'imaginer des solutions VoD complètes basées sur des serveurs Web et des lecteurs multimédias existant sur chaque ordinateur récent ([ms-wmp] ou [rn-rp]). Nous avons détaillé cette proposition dans l'annexe 1 de la thèse.

7.5 Résultats réutilisables

Une partie des résultats obtenus dans notre thèse ont un degré de généralité autorisant leur utilisation en dehors du domaine de la VoD. La stratégie d'utilisation de HTTP permet l'utilisation efficace de tous les serveurs Web avec des stratégies "pull". La méthodologie de tests pour les serveurs VoD s'inscrit dans le cadre plus général défini par les méthodes de benchmarking pour les serveurs Web. La mesure des performances d'un système de stockage peut fournir des résultats intéressants pour une catégorie plus grande d'applications s'appuyant sur un système de stockage performant.

Bibliographie- personnelle

- [ict01] : Vasile-Marian Scuturici, Mihaela Scuturici, Serge Miguet, Jean-Marie Pinon. Measuring Web Servers Performance in VoD. International Conference on Telecommunications (ICT2001), Romania, Bucharest, June 4-7, 2001
- [proms00] : Vasile-Marian SCUTURICI, Mihaela SCUTURICI. Video on Demand Using HTTP. PROtocols for Multimedia Systems - PROMS2000, Cracow, Poland, October 22-25, 2000, p. 228-235, ([http : //proms2000.kt.agh.edu.pl/index.html](http://proms2000.kt.agh.edu.pl/index.html))
- [isimade99] : Serge Miguet, Vasile-Marian Scuturici. Using the Hyper Text Transfer Protocol for Multimedia Streaming. International Symposium on Intelligent Multimedia and Distance Education, Baden-Baden, Germany, 2-7 august 1999, p. 215-223, ([http : //www.ece.ndsu.nodak.edu/ece/research/conferences/isimade/](http://www.ece.ndsu.nodak.edu/ece/research/conferences/isimade/))
- [isei97] : Dorel Bozga, Dan Chiorean, Vasile-Marian Scuturici, Dan Suciu, Dan Vasilescu. Present and Perspectives Of the CASE Tools Used In Object-Oriented Analysis & Design -The RO_CASE Experience. 3rd International Symposium in Economics Informatics. 1997, Bucharest. p. 21-28.
- [studia98] : Serge Miguet, Vasile-Marian Scuturici. Metrics for Measuring the Web Servers Performance. Studia Universitatis Babes-Bolyai, Series Computer Science, Cluj-Napoca, Romania, 1998.
- [studia97] : Vasile-Marian Scuturici, Dan Suciu, Mihaela Scuturici, Iulian Ober. The Active Objects Description Using Statecharts. Studia Universitatis Babes-Bolyai, Cluj-Napoca, 1997.
- [studia96] : Michael Papathomas, Vasile-Marian Scuturici. An Active Object Model for Multimedia Presentations. Studia Universitatis "Babes-Bolyai", Informatica, Volume I, Number 1 1996. p. 21 – 40.
- [coresa99] : Vasile-Marian Scuturici. L'utilisation de serveurs Web pour la VoD. COmpression et REprésentation des Signaux Audiovisuels, CORESA'99, Sophia-Antipolis, juin 1999, p. 203-209.

Bibliographie

- [abdulla96] : G. Abdulla, M. Abrams, E. A. Fox. Scaling the World-Wide Web. Virginia Tech Dept. of Computer Science, Blacksburg, VA, Working paper, March, 1996, disponible à l'adresse : <http://ei.cs.vt.edu/~succeed/96ieeeAAF/>
- [abdulla97] : Ghaleb Abdulla, Edward A. Fox, Marc Abrams. Shared User Behavior on the World Wide Web. Proceedings of WebNet 97. Toronto, Canada, October 1997. p. 54-59, disponible à l'adresse : <http://vtopus.cs.vt.edu/~chitra/docs/97webnet/>
- [activex] : COM : Delivering on the Promises of Component Technology, <http://www.microsoft.com/com/>
- [adsl] : xDSL.com, Analysis of DSL technologies, <http://www.xdsl.com/>
- [afn95] : Association Française de Normalisation. Codage de l'image animée et du son associé pour les supports de stockage numérique jusqu'à environ 1,5 Mbit/s. norme européenne, NF EN ISO/CEI 11172-1, 1995
- [aggarwal97] : Charu C. Aggarwal, Philip S. Yu. On Disk Caching of Web Objects in Proxy Servers. CIKM 97, Las Vegas, 1997. p 238-244.
- [alex] : Alex Temex Multimedia, Serveurs Multimédia Haut Débit, <http://www.alex.com/>
- [almeida96] : V .A .F. Almeida, J.M. de Almeida, C.D Murta, A.A Oliveira. Performance Analysis and Modeling of a WWW Internet Server. In International Conference on Telecommunication Systems, Nashville, March 1996
- [almeida96b] : J. Almedia, V. Almedia, D. Yates. Measuring the behavior of a

- World-Wide Web server. Proceedings of 7th Conference High Performance Networking, April 1997. p. 57-72.
- [ams] : CS Technologies Informatiques. ANTARA Multimedia Server. Document 102-00131, CSTI, 1999
- [antara] : CS Technologies Informatiques. ANTARA System Functional Specification. Document 101-00061, CSTI, April 1996
- [apache] : Apache HTTP Server disponible à l'adresse : [http : //httpd.apache.org/](http://httpd.apache.org/)
- [approach01] : May 25, 2001, New York - Approach Study : Microsoft Windows Media Services and RealNetworks RealSystem Feature Comparison, disponible à l'adresse : [http : //www.approach.com/](http://www.approach.com/)
- [approach99] : Approach, Inc., Streaming Media : A Comparative Cost Analysis Microsoft® Windows® Media Technologies And Realnetworks™ Realsystem G2. White Paper. January 1999 disponible à l'adresse : [http : //www.microsoft.com/windows/windowsmedia/en/compare/wmrcostcompare.asp](http://www.microsoft.com/windows/windowsmedia/en/compare/wmrcostcompare.asp)
- [apteker95] : R.T. Apteker, J.A. Fisher, V.S. Kisimov, and H. Neishlos. Video Acceptability and Frame rate. IEEE Multimedia, 2 (3), 1995. p. 32-40.
- [arlitt97] : Martin Arlitt, Carey Williamson. Internet Web Servers : Workload characterization and Performance Implications. IEEE Transaction on Networking, vol. 5, no. 5, October 1997, p. 631-645.
- [ata] : National Committee on Information Technology Standards, Technical Committee T13, AT Attachment, disponible à l'adresse : [http : //www.t13.org/](http://www.t13.org/)
- [atm-forum] : ATMForum, Traffic management Specification version 4.0, Document af-tm-0056.000, ATMForum, April 1996
- [atsc] : Advanced Television Standards Committee, disponible à l'adresse : [http : //www.atsc.org/](http://www.atsc.org/)
- [baird-smith97] : Baird-Smith, Anselm. Jigsaw : An object oriented server. World Wide Web Consortium, February 1997, disponible à l'adresse : [http : //www.w3.org/Jigsaw/User/Introduction/wp.html](http://www.w3.org/Jigsaw/User/Introduction/wp.html)
- [banga97] : Gaurav Banga, Peter Druschel. Measuring the capacity of a Web Server. In USENIX Symposium of Internet Technologies and Systems, Monterey, CA, December 1997. p. 61-71.
- [barford98] : Paul Barford, Mark Crovella. Generating representative Web workloads for network and server performance evaluation. Proceedings of the 1998 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, July 1998. p. 151-160, disponible à l'adresse : [http : //www.cs.bu.edu/faculty/crovella/paper-archive/sigm98-surge.ps](http://www.cs.bu.edu/faculty/crovella/paper-archive/sigm98-surge.ps)
- [belloum98] : A. Belloum, A.J.H. Peddemors, L. O. Hertzberger. JERA : A Scalable Web Server. Proceedings of the PDPTA'98 conference, Las Vegas, NV, 1998, p. 167-174.
- [benoit97] : Herve Benoit. La Télévision Numérique. DUNOD, 1997
- [benoit97b] : Benoit, Herve. Digital Television : MPEG- 2 & Principles of the DVB System. ISBN : 0471238104, Wiley, John & Sons, Incorporated, January 1997.
- [bolosky96] : W. J. Bolosky, J. S. Barrera, R. P. Draves, R. P. Fitzgerald, G. A. Gibson,

- M. B. Jones, S. P. Levi, N. P. Myhrvold, R. F. Rashid. The tiger video fileserver. Proceedings of the 6th Int'l Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), April 23-26, 1996, Zushi, Japan. p. 97-104.
- [bolot96] : J.-C. Bolot, P. Hoschka. Performance engineering of the World-Wide Web : Application to dimensioning and cache design. Computer Networks and ISDN Systems, Vol. 28, No. 6; Proceedings of 5th International Conference on the World-Wide Web, Paris, May 1996.
- [bonhomme00] : Alice Bonhomme, Loïc Prylli. A distributed storage system for a video-on-demand server. Euro-Par 2000 : Parallel Processing, volume 1900 of Lect. Notes in Comp. Science, p. 1110–1114, Munchen, Germany, August 2000. Springer-Verlag.
- [bonhomme01] : Alice Bonhomme, Loïc Prylli. Un système de stockage distribué pour un serveur de vidéo à la demande. Revue Électronique sur les Réseaux et l'Informatique Répartie, Mars 2001
- [britannica] : Encyclopaedia Britannica disponible à l'adresse : [http : //www.britannica.com](http://www.britannica.com)
- [burger93] : Jeff Burger. The Desktop Multimedia Bible. Addison-Wesley, 1993
- [chang00] : S.F. Chang, Q. Huang, A. Puri, B. Shahraray, T. Huang. Multimedia search and retrieval. Multimedia Systems, Standards, and Networks, MARCEL DEKKER, 2000.
- [chen95] : Zhigang Chen Chen, See-Mong Tan, Roy H. Campbell, Yongchen Li. Real time video and audio in the world wide web. World Wide Web Journal, Volume 1, Number 1, December 1995. p. 333-348.
- [chen99] : Meng Chang Chen, Jan-Ming Ho, Ming-Tat Ko, Shie-Yuan Wang, A SCSI disk model for multimedia storage systems, International Journal of Computer Systems Science and Engineering, vol. 14, no. 3, p. 147-154, 1999
- [chervenak95] : Ann Chervenak, David Patterson, Randy Katz. Choosing the Best Storage System for Video Service. ACM Multimedia 1995. p.109-119.
- [cisco] : Cisco Systems, Inc., Cisco LocalDirector, disponible à l'adresse : [http : //www.cisco.com/warp/public/cc/cisco/mkt/scale/locald/index.shtml](http://www.cisco.com/warp/public/cc/cisco/mkt/scale/locald/index.shtml)
- [colajanni01] : Michele Colajanni, Philip S. Yu. A Performance Study of Robust Load Sharing Strategies for Distributed Heterogenous Web Servers. IEEE Transactions on Knowledge and Data Engineering, 2001.
- [Compaq] : Compaq Computer Corporation, [http : //www.compaq.com/](http://www.compaq.com/)
- [dell-hdd] : Dell Storage Systems, [http : //www.dell.com/](http://www.dell.com/)
- [dias96] : D. Dias, W. Kish, R. Mukherjee, R. Tewari. A Scalable and Highly Available Web Server. Proceedings of the 1996 IEEE Computer Conference (COMPCON), February 1996.
- [directx] : Microsoft® DirectX® Web site, [http : //www.microsoft.com/directx/](http://www.microsoft.com/directx/)
- [divx] : DivX Format, disponible à l'adresse : [http : //www.divx.com](http://www.divx.com)
- [duda94] : Andrzej Duda, Ron Weiss, David K. Gifford. Content-based access to algebraic video. Proceedings of the International Conference on Multimedia

- Computing and Systems, Boston, MA, May 1994. p. 140-151.
- [dvb] : Digital Video Broadcasting Consortium, disponible à l'adresse : [http : //www.dvb.org/](http://www.dvb.org/)
- [dygert] : Tim Dygert. High Quality Video Streaming Over Local Area Networks, disponible à l'adresse : [http : //www.netology.com](http://www.netology.com)
- [encarta] : Microsoft® Encarta® Online Encyclopedia 2000, "Multimedia" disponible à l'adresse : [http : //encarta.msn.com](http://encarta.msn.com)
- [ethernet] : IEEE Standards, disponible à l'adresse : [http : //standards.ieee.org/](http://standards.ieee.org/)
- [fdida97] : Fdida S., Fourmaux O., Onvural R. Enabling Multimedia Networks. Revue Electronique sur les reseaux et l'Informatique Repartie (RERIR/EJNDP), ISSN 1262-3261, vol 5, Mars 97, p. 28-32.
- [ferrari94] : D. Ferrari, A. Banerjea, H. Zhang. Network Support for Multimedia : A Discussion of the Tenet Approach. Computer Networks and ISDN Systems, vol. 26, 1994. p. 1267 - 1280.
- [fry97] : Fry, M., Seneviratne, A., Vogel, A., Witana, V. QoS management in a World Wide Web environment which supports continuous media. Distributed Systems Engineering vol 4, IOP Publishing, 1997. p. 38-47.
- [garcia00] : Andrew R. Garcia. Benchmark Tests : Web Platforms. May 2, 2000, disponible à l'adresse : [http : //www.zdnet.com/pcmag/stories/reviews/0,6755,2551188,00.html](http://www.zdnet.com/pcmag/stories/reviews/0,6755,2551188,00.html)
- [ghandeharizadeh98] : Shahram Ghandeharizadeh, Richard Muntz. Design and Implementation of Scalable Continuous Media Servers. Parallel Computing, Elsevier, Volume 24, p. 91-122, 1998, disponible à l'adresse : [http : //dblab.usc.edu/Users/shkim/dblab_papers.html](http://dblab.usc.edu/Users/shkim/dblab_papers.html)
- [ghinea98] : G. Ghinea and J.P. Thomas. QoS Impact on User Perception and Understanding of Multimedia Video Clips. Proceedings of ACM Multimedia 1998. Bristol, United Kingdom, 1998. p 49-54, disponible à l'adresse : [http : //water.ite.ntnu.edu.tw/doc/electronic_proceedings/ghinea/index.html](http://water.ite.ntnu.edu.tw/doc/electronic_proceedings/ghinea/index.html)
- [ghinea99] : G. Ghinea, J.P. Thomas, R.S. Fish. Multimedia Network Protocols and Users Bridging the Gap. Proceedings of ACM Multimedia 1999, disponible à l'adresse : [http : //www.kom.technik.tudarmstadt.de/acmmm99/ep/ghinea/index.html](http://www.kom.technik.tudarmstadt.de/acmmm99/ep/ghinea/index.html)
- [gibbs91] : Simon Gibbs. Composite Multimedia and Active Objects. Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA), ACM SIGPLAN Notices, Vol. 26, No. 11, 1991, ACM Press. p. 97-112.
- [Gibbs94] : Simon Gibbs, Christian Breiteneder, and Dennis Tschritzis. "Data Modeling of Time-Based Media". In Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, pages 91--102, Minneapolis, Minnesota, May 1994.
- [hafid98] : Abdelhakim Hafid, Gregor von Bochmann, Rachida Dssouli. Distributed Multimedia Applications and Quality of Service : A Review. Electronic Journal on Networks and Distributed Processing, No. 6, 2, 1998. p. 1-50
- [heideman97] : John Heideman, Katia Obraczka, Joe Touch. Modeling the Performance of HTTP Over Several Transport Protocols. IEEE/ACM Transactions on Networking.

-
- Vol. 5, No. 5. October 1997. p. 616-630.
- [heidemann97b] : John Heidemann. Performance Interactions between P-HTTP and TCP Implementations. ACM Computer Communications Review. April 1997. p. 65-73.
- [http1.0] : Fielding, R., Frystyk, H., Berners-Lee, T. Hypertext Transfer Protocol -- HTTP/1.0. RFC-1945, disponible à l'adresse : [http : //www.w3.org/Protocols/rfc1945/rfc1945](http://www.w3.org/Protocols/rfc1945/rfc1945)
- [http1.1] : Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Berners-Lee, T. Hypertext Transfer Protocol-- HTTP/1.1. RFC-2068, disponible à l'adresse : [ftp : //ds.internic.net/rfc/ rfc2068.txt](ftp://ds.internic.net/rfc/rfc2068.txt)
- [hu97] : J. Hu, I. Pyarali, and D. C. Schmidt. Measuring the Impact of Event Dispatching and Concurrency Models on Web Server Performance Over High-speed Networks. Proceedings of the 2 nd Global Internet Conference, IEEE, November 1997.
- [hu98] : James Hu, Sumedh Mungee, Douglas C. Schmidt. Principles for Developing and Measuring High-performance Web Servers over High Speed Networks. INFOCOM '98 Proceedings, March/April 1998.
- [hu99] : James C. Hu, Douglas C. Schmidt. JAWS : A Framework for High-Performance Web Servers. Domain Specific Application Frameworks : Frameworks Experience by Industry, Wiley & Sons, 1999.
- [ibm-storage] : IBM Storage, [http : //www.storage.ibm.com/](http://www.storage.ibm.com/)
- [iis] : Microsoft Internet Information Server disponible à l'adresse : [http : //www.microsoft.com/](http://www.microsoft.com/)
- [ipv6] : R. Gilligan, S. Thomson, J. Bound, W. Stevens, April 1997, Basic Socket Interface Extensions for IPv6, disponible à l'adresse : [http : //www.cis.ohio-state.edu/cgi-bin/rfc/rfc2133.html](http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2133.html)
- [iqms] : iQms disponible à l'adresse : [https : //www.qsound.com/](https://www.qsound.com/)
- [iso13818-1] : International standard. Information technology – Generic coding of moving pictures and associated audio information : Systems. ISO/IEC 13818-1, 1996
- [iso13818-2] : Norme internationale. Technologie de l'information – Codage générique des images animées et du son associé : Données vidéo. ISO/CEI 13818-2, 1996
- [iyengar00] : A. Iyengar, J. Challenger, D. Dias, and P. Dantzig. High-Performance Web Site Design Techniques. IEEE Internet Computing, Vol. 4., N. 2, March/April 2000. p. 17-26.
- [jong97] : Alex De Jong, Karen Hsing, David Su. A VoD Application Implemented in Java. Multimedia Tools and Applications, Kluwer Academic Publishers, 1997
- [jong97-1] : Alex de Jong, Taewoon Kang. Acces to a DAVIC-Based Video on Demand System Using the World Wide Web. 11-th International Conference on Information Networks, Taipei, Taiwan, 1997
- [katz94] : Eric D. Katz, Michelle Butler, Robert McGarth. A Scalable HTTP Server : The NCSA Prototype. Computer Networks and ISDN Systems. vol. 27, 1994. p. 155-164.
- [khoshafian96] : Setrag Khoshafian, A. Brad Baker. Multimedia and Imaging Databases. Morgan Kaufmann Publishers, 1996
- [krishnamurthy99] : Balachander Krishnamurthy, Jeffrey C. Mogul, David M. Kristol. Key

- differences between HTTP/1.0 and HTTP/1.1. Eighth International World Wide Web Conference, Toronto, Canada, May 1999. p. 659-673, disponible à l'adresse : [http : //www8.org/w8-papers/5c-protocols/key/key.html](http://www8.org/w8-papers/5c-protocols/key/key.html)
- [kwan95] : T. T. Kwan, R. E. McGrath, D. A. Reed. NCSA's World Wide Web Server : Design and Performance. IEEE Computer, 28(11), November 1995. p. 68-74.
- [laursen95] : A. Laursen, J. Olkin, M. Porter. Oracle media server framework. Compcon : Digest of Papers, IEEE, 1995. p. 203-208.
- [lee97] : Meng-Huang Lee, Meng Chang Chen, Jan-Ming Ho, Ming-Tat Ko. Disk Layout of Near-Video-on-Demand System. Storage and Retrieval for Image and Video Databases (SPIE) 1997, p. 390-397
- [leon98] : Ernie G. Leon. The Impact of Digital Video Servers on Broadcast Studio Efficiency, Profitability and Growth. Concurrent Computer Corporation, Ft. Lauderdale, Florida, 1998 disponible à l'adresse : [http : //www.ccur.com](http://www.ccur.com)
- [lizzi00] : Christophe Lizzi. Conception d'un système distribué temps réel fondé sur ATM. thèse de doctorat, Conservatoire National d'Arts et Métiers, Paris 2000
- [machanick98] : Philip Machanick. A Scalable Architecture for Video on Demand : SAVoD. Technical Report, 1998. 13. disponible à l'adresse : [http : //www.cs.wits.ac.za/~philip/papers/SAVoD.html](http://www.cs.wits.ac.za/~philip/papers/SAVoD.html)
- [manley97] : Stephen Manley, Margo Seltzer. Web Facts and Fantasy. Proceedings of the 1997 USENIX Symposium on Internet Technologies and Systems. p. 125-133, disponible à l'adresse : [http : //www.eecs.Harvard.edu/~vino/web/sits.97.html](http://www.eecs.Harvard.edu/~vino/web/sits.97.html)
- [martin96] : C. Martin, P.S. Narayanan, B. Ozden, R. Rastogi, A. Silberschatz. The Fellini Multimedia Storage Server. Multimedia Information Storage and Management, Editor S.M. Chung. Kluwer Academic Publishers, 1996.
- [mcgrath96] : Robert E. McGrath. Measuring the Performance of HTTP Daemons. NCSA, February 5, 1996 disponible à l'adresse : [http : //www.ncsa.uiuc.edu/InformationServers/Performance/Benchmarking/bench.html](http://www.ncsa.uiuc.edu/InformationServers/Performance/Benchmarking/bench.html)
- [mcgrath96-2] : Robert E. McGrath. Performance of Several Web Server Platforms. NCSA, January 22, 1996, disponible à l'adresse : [http : //www.ncsa.uiuc.edu/InformationServers/Performance/Platforms/report.html](http://www.ncsa.uiuc.edu/InformationServers/Performance/Platforms/report.html)
- [menon] : Satish Menon. SGI MediaBase : Intelligent Media Streaming for Intranets and the Internet. White Paper, disponible à l'adresse : [http : //www.sgi.com/software/mediabase](http://www.sgi.com/software/mediabase)
- [midisoft] : Midisoft Internet Media Player disponible à l'adresse : [http : //www.midisoft.com/](http://www.midisoft.com/)
- [mogul95] : Mogul, J., The Case for Persistent Connection HTTP. Proceedings of the ACM SIGCOMM`95 Conference on Communications Architectures and Protocols. August 1995. Boston, MA, USA. p. 299-313.
- [mosberger98] : D. Mosberger, T. Jin. httpperf : A tool for measuring web server performance. Proceedings of the 1998 Workshop on Internet Server Performance (WISP), Madison, WI, June 1998, ACM. p. 59 – 67.
- [MPEG1] : ISO/IEC 11172-1, ISO/IEC 11172-2, ISO/IEC 11172-3, ISO/IEC 11172-4, ISO/IEC 11172-5 : 1993 Information technology -- Coding of moving pictures and

associated audio for digital storage media at up to about 1,5 Mbit/s, Part 1-5

- [MPEG2] : ISO/IEC 13818-1, ISO/IEC 13818-2, ISO/IEC 13818-3, ISO/IEC 13818-4, ISO/IEC 13818-5, ISO/IEC 13818-6, ISO/IEC 13818-7, ISO/IEC 13818-8, ISO/IEC 13818-9 : 2000 Information technology -- Generic coding of moving pictures and associated audio information
- [MPEG4] : ISO/IEC JTC1/SC29/WG11, Coding of moving pictures and audio, Overview of the MPEG-4 Standard, [http : //www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm](http://www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm)
- [mpeg4tv] : Mpeg4TV Player, disponible à l'adresse : [http : //www.mpeg4tv.com](http://www.mpeg4tv.com)
- [mps] : MPS, Media Player Shell, disponible à l'adresse : [http : //mps.comcen.com.au/](http://mps.comcen.com.au/)
- [msdn-raid] : Microsoft Developers Network. RAID Levels and SQL Server disponible à l'adresse : [http : //msdn.microsoft.com/library/default.asp?URL=/library/psdk/sql/tun_1_6.htm](http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/sql/tun_1_6.htm)
- [msdn-vs] : The TransmitFile() API, MSDN Library Visual Studio 6.0
- [ms-http] : Microsoft White Paper. Streaming Methods : Web Server vs. Streaming Media Server disponible à l'adresse : [http : //www.microsoft.com/ntserver/mediaserv/exec/comparison/webservvstreamserv.asp](http://www.microsoft.com/ntserver/mediaserv/exec/comparison/webservvstreamserv.asp)
- [ms-netshow] : Microsoft Corporation (2000), NetShow Theater Server Web Page disponible à l'adresse : [http : //www.microsoft.com/Theater/](http://www.microsoft.com/Theater/)
- [ms-technet] : IIS Documentation, disponible à l'adresse : [http : //www.microsoft.com/technet](http://www.microsoft.com/technet)
- [ms-wmp] : Windows Media Player, disponible à l'adresse : [http : //www.microsoft.com](http://www.microsoft.com)
- [ms-wmp] : Windows Media Services disponible à l'adresse : [http : //www.microsoft.com/windowsmedia/](http://www.microsoft.com/windowsmedia/)
- [nahrstedt95] : Nahrstedt, K., Steinmetz, R. Resource Management in Networked Multimedia Systems. IEEE Computer, Vol. 28, No. 5, May 1995. p. 52-63.
- [nahrstedt96] : Klara Nahrstedt, Lintian Qiao. Tuning system for distributed multimedia applications. Technical Report No. UIUCDCS-R-96-1958, UILU-ENG-96-1721, University of Illinois, May 1996, disponible à l'adresse : [http : //citeseer.nj.nec.com/15905.html](http://citeseer.nj.nec.com/15905.html)
- [ncube] : nCUBE, [http : //www.ncube.com/](http://www.ncube.com/)
- [netcraft] : Netcraft Survey, disponible à l'adresse : [http : //www.netcraft.com/survey/](http://www.netcraft.com/survey/)
- [netscape] : Netscape Corporation, [http : //www.netscape.com](http://www.netscape.com)
- [netshow-price] : NetShow Theater Server : Capacity and Cost Planner, disponible à l'adresse : [http : //www.microsoft.com/Theater/planner.htm](http://www.microsoft.com/Theater/planner.htm)
- [nielsen97] : Nielsen, H., Gettys, J., Baird-Smith, A., Prud'hommeaux, E., Lie, H., Lilley, C. Network Performance Effects of HTTP/1.1, CSS1, and PNG. Proceedings of ACM SIGCOMM'97 Conference. September 1997. Cannes, France. p. 155-166 disponible à l'adresse : [http : //www.w3.org/pub/WWW/Protocols/HTTP/Performance/Pipeline.html](http://www.w3.org/pub/WWW/Protocols/HTTP/Performance/Pipeline.html)
- [nwosu96] : Kingsley C. Nwosu, Bhavani Thuraisingham, P. Bruce Berra. Multimedia Database Systems, Design and Implementation Strategies. Kluwer Academic Publishers, 1996

- [ozden95] : B. Ozden, R. Rastogi, and A. Silberschatz. A Framework for the Storage and Retrieval of Continuous Media Data. Proceedings of the 1995 International Conference on Multimedia Computing and Systems. Washington, D.C., May 1995. p. 2-13.
- [pcguide] : The PC Guide disponible à l'adresse : [http : //www.pcguide.com/](http://www.pcguide.com/)
- [pcguide01] : Hard Disk Performance, Quality and Reliability. PC-Guide, 2001, disponible à l'adresse : [http : //www.pcguide.com/ref/hdd/perf/perf/spec/posLatency-c.html](http://www.pcguide.com/ref/hdd/perf/perf/spec/posLatency-c.html)
- [pc-mag] : PC-Magazine : Web Servers disponible à l'adresse : [http : //www.zdnet.com/pcmag/features/webserver98/intro.html](http://www.zdnet.com/pcmag/features/webserver98/intro.html)
- [petit-robert] : Le Petit Robert de la langue française, [http : //www.lerobert.com.fr/](http://www.lerobert.com.fr/)
- [pham98] : C.D. Pham, H. Brunst, S. Fdida. How Can We Study Large and Complex Systems. Proceedings of the 31th Annual Simulation Symposium, April 5-9 1998, Boston, USA, p. 126-134
- [podgorny97] : Marek Podgorny and Geoffrey C. Fox. Video on Demand Technologies and Demonstrations. Tech. Rep., Northeast Parallel Architectures Center. Syracuse. March 1997, disponible à l'adresse : [http : //trurl.npac.syr.edu/rivod/report/report.html](http://trurl.npac.syr.edu/rivod/report/report.html)
- [quantum] : Quantum Corporation, [http : //www.quantum.com/](http://www.quantum.com/)
- [quicktime] : Apple Quicktime, disponible à l'adresse : [http : //www.apple.com/quicktime/](http://www.apple.com/quicktime/)
- [radware] : Radware Ltd., Complete IP Load balancing Solutions from RADWARE disponible à l'adresse : [http : //www.radware.co.il](http://www.radware.co.il)
- [rangan92] : P. Rangan, H. Vin, S. Ramanathan. Designing an On-Demand Multimedia Service. IEEE Communications Magazine, Vol. 30, No. 7, July 1992, p. 56-65
- [rao96] : Sriram S. Rao, Harrick M. Vin, Ashis Tarafdar. Comparative Evaluation of Server-push and Client-pull Architectures for Multimedia Servers. Proceedings of the 6th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'96), Japan, April 1996. p. 45-48.
- [real-price] : Compare RealSystem Server Intranet Configurations, disponible à l'adresse : [http : //www.realnetworks.com/products/servers/intranet/comparison_matrix.html](http://www.realnetworks.com/products/servers/intranet/comparison_matrix.html)
- [resonate] : Resonate, Inc., Central Dispatch - Data Sheets, disponible à l'adresse : [http : //www.resonate.com/products/central_dispatch/data_sheets.html](http://www.resonate.com/products/central_dispatch/data_sheets.html)
- [rn-http96] : Real Networks White Paper. HTTP versus RealAudio Client-Server Streaming. 1996 disponible à l'adresse : [http : //service.real.com/help/content/http_vs_ra.html](http://service.real.com/help/content/http_vs_ra.html)
- [rn-rs] : RealSystem Server, disponible à l'adresse : [http : //www.realnetworks.com/](http://www.realnetworks.com/)
- [rowe92] : L. Rowe, B. Smith. A Continuous Media Player. Network and Operating System Support for Digital Audio and Video : Third International Workshop. La Jolla, California, USA, November 12-13, 1992. Springer-Verlag, Berlin; New York p. 334-344
- [rn-rp] : Real Networks Player disponible à l'adresse : [http : //www.real.com](http://www.real.com)
- [rtsp] : Real Time Streaming Protocol, RFC 2326, disponible à l'adresse : [http : //www.ietf.org/rfc/rfc2326.txt](http://www.ietf.org/rfc/rfc2326.txt)

[//www.cis.ohio-state.edu/htbin/rfc/rfc2326.html](http://www.cis.ohio-state.edu/htbin/rfc/rfc2326.html)

- [rubarth96] : James Rubarth-Lay. Keeping the 400lb. Gorilla at Bay : Optimizing Web Performance. Class paper for UT Austin LIS385T.6, Spring 96 1996, disponible à l'adresse : [http : //eunuch.ddg.com/LIS/CyberHornsS96/ j.rubarth-lay/PAPER.html](http://eunuch.ddg.com/LIS/CyberHornsS96/j.rubarth-lay/PAPER.html)
- [schulzrinne95] : H. Schulzrinne, S. Casner, V. Jacobson. RTP : A Transport for Real-Time Applications. Internet Engineering Task Force, Internet Draft, 1995
- [scsi] : ANSI X3T9.2. Small Computer System Interface-2. 1993. Draft Revision 10k.
- [scsi] : Small Computer System Interface, National Committee on Information Technology Standards, T10 Technical Committee, AT Attachment, disponible à l'adresse : [http : //www.t10.org/](http://www.t10.org/)
- [servetto99] : Sergio Servetto, Klara Nahrstedt. Video Streaming over the Public Internet : Multiple Description Codes and Adaptive Transport Protocols. Proceedings of the IEEE International Conference on Image Processing (ICIP), October 25-29, 1999, Kobe, Japan.
- [siren] : Sonic Foundry Siren Player. Sonic Foundry disponible à l'adresse : [http : //www.sonicfoundry.com/](http://www.sonicfoundry.com/)
- [sonique] : Sonique Player disponible à l'adresse : [http : //sonique.lycos.com/](http://sonique.lycos.com/)
- [specweb99] : The Standard Performance Evaluation Corporation, SpecWeb99 Benchmark, disponible à l'adresse : [http : //www.specbench.org/osg/web99](http://www.specbench.org/osg/web99), 1999
- [spero94] : Simon E Spero. Analysis of HTTP Performance problems. Electronical Proceedings of the 2nd. International WWW Conference, Chicago, USA, October 1994 disponible à l'adresse : [http : //www.ibiblio.org/mdma-release/http-prob.html](http://www.ibiblio.org/mdma-release/http-prob.html)
- [steinmetz95] : Ralf Steinmetz, Klara Nahrstedt. Multimedia : Computing, Communications, and Applications. Prentice Hall, July 1995
- [steinmetz96] : R. Steinmetz. Human Perception of Jitter and Media Synchronization. IEEE Journal on Selected Areas in Communications, 14 (1), 1996. p. 61-72.
- [storagereview] : StorageReview.com, [http : //www.storagereview.com](http://www.storagereview.com)
- [tanenbaum97] : Andrew S. Tanenbaum. Retele de calculatoare. Computer Press Agora, Targu-Mures, Romania, 1997
- [tcp], [udp] : Eric A. Hall, Internet Core Protocols, O'Reilly, 2000
- [tcp-tech.net] : Microsoft Windows 2000 TCP/IP Implementation Details, disponible à l'adresse : [http : //www.microsoft.com/TechNet/network/tcpip2k.asp](http://www.microsoft.com/TechNet/network/tcpip2k.asp)
- [ting-sun01] : Ming Ting-Sun, Amy R. Reibman. Compressed Video over Networks. 2001, Marcel Dekker Inc., ISBN : 0-8247-9423-0
- [trent95] : Gene Trent, Mark Sake. WebStone : The first generation in HTTP Server Benchmarking. February 1995, disponible à l'adresse : [http : //www.mindcraft.com/webstone/paper.html](http://www.mindcraft.com/webstone/paper.html).
- [videonow] : CVideoNow Player, disponible à l'adresse : [http : //www.cvideo.com/](http://www.cvideo.com/)
- [vivo] : Vivo Software, disponible à l'adresse : [http : //www.vivo.com](http://www.vivo.com)
- [w3c] : Web Characterization Terminology & Definitions Sheet, disponible à l'adresse : [http : //www.w3.org/1999/05/WCA-terms/](http://www.w3.org/1999/05/WCA-terms/)

- [webbench] : WebBench 4.0 disponible à l'adresse : [http : //www.zdnet.com/](http://www.zdnet.com/)
- [WebForce] : WebForce Server Tuning Guide, disponible à l'adresse : [http : //www.sgi.com/Technology/web/formal_tuning_guide.html](http://www.sgi.com/Technology/web/formal_tuning_guide.html)
- [webopedia] : Webopedia : The online dictionary for computer and Internet technology disponible à l'adresse : [http : //webopedia.internet.com/](http://webopedia.internet.com/)
- [winamp] : Nullsoft Winamp disponible à l'adresse : [http : //winamp.com/](http://winamp.com/)
- [wolf94] : L.C. Wolf, R.G. Herrtwich. The System Architecture of the Heidelberg Transport System. ACM Operating Systems Review, Vol28. No.2, Apr 1994. p.51-64.
- [xing] : Xing Player, disponible à l'adresse : [http : //www.xingtech.com](http://www.xingtech.com)
- [yeager96] : Nancy J. Yeager and Robert E. McGrath. Web Server Technology : The advanced Guide for World Wide Web information Providers. Morgan Kaufmann, 1996
- [yes-tv] : Yes Television, disponible à l'adresse : [http : //www.yes-tv.co.uk/](http://www.yes-tv.co.uk/)

Annexe. Système VoD basé sur des serveurs Web - étude de cas

Afin de démontrer la possibilité d'une utilisation commerciale des résultats présentés de cette thèse, nous décrivons dans cette partie un système VoD repartitionné complet utilisant des serveurs Web comme serveurs vidéo, dont un prototype a été réalisé au sein de la société CSTI.

L'architecture

Le système que nous proposons comprend plusieurs serveurs Web hébergeant les objets multimédias destinés aux clients, un ou plusieurs nœuds administrateur et les clients à servir. Un client peut accéder aux objets en utilisant un lecteur multimédia fonctionnant avec le protocole HTTP selon le principe décrit dans le Chapitre 5 (comme Windows Media Player ou Real Video Player). Le contenu multimédia partagé sur les nœuds disponibles est géré par un logiciel spécifique qui génère des pages Web permettant un accès facile à tous ces objets.

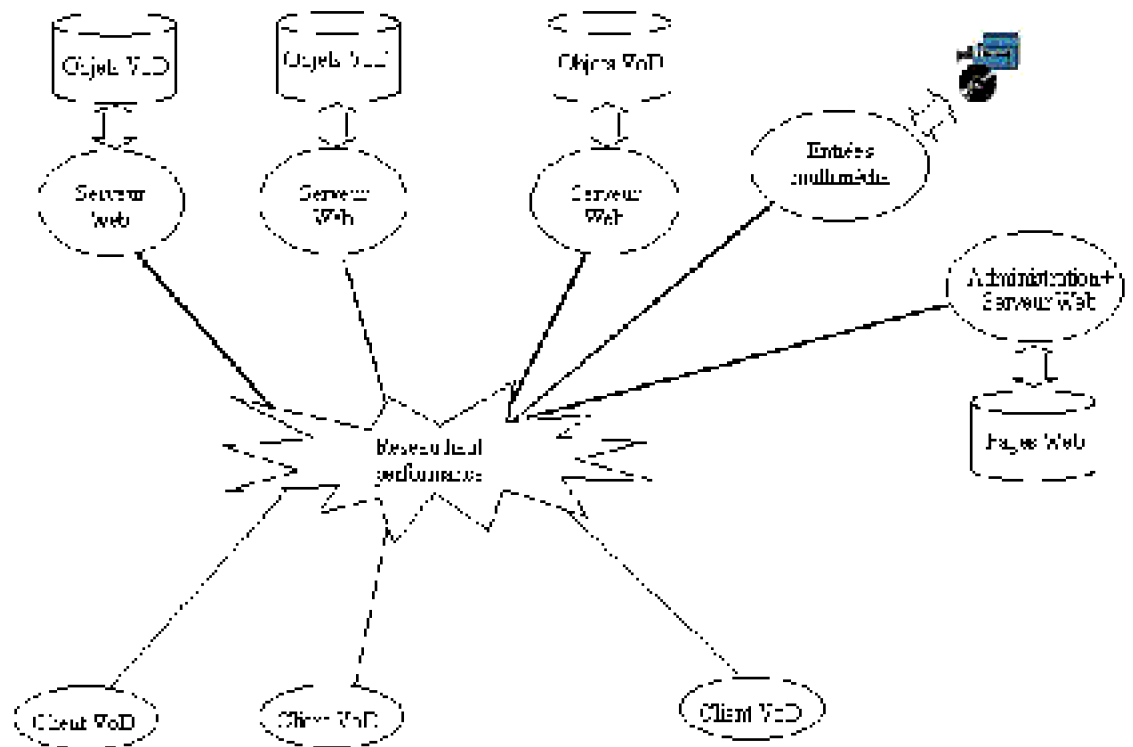


Figure 81 L'architecture d'un système VoD basé sur des serveurs Web

L'accès au contenu se fait au travers de pages Web organisées de manière arborescente, avec des liens transversales liant les pages d'un même niveau (Figure 82). Plusieurs pages sommaires permettent l'accès à d'autres pages contenant la description de chaque objet multimédia partagé. A partir de ces pages, l'utilisateur peut démarrer la visualisation d'un objet qui sera joué par le lecteur multimédia disponible (WMP ou RP). Les pages Web sont stockées sur le serveur d'administration hébergeant aussi le serveur Web pour fournir les pages aux clients.

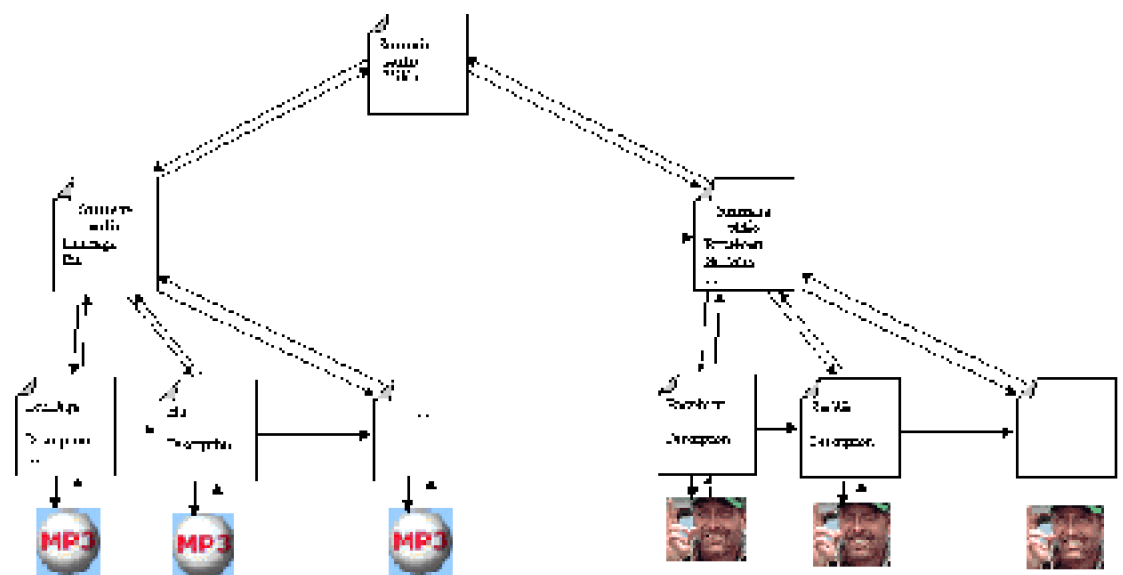


Figure 82 Structure d'organisation du contenu d'un système VoD

Pour consulter ces pages et accéder au service VoD, chaque client dispose d'un navigateur Web ([ms-ie] ou [netscape]), configuré pour occuper tout l'écran. Les pages Web permettent à l'utilisateur de passer en revue l'offre du service VoD à l'aide d'une télécommande (pour les TV avec STB), d'un clavier ou d'une souris (pour un ordinateur).

Un exemple simple d'organisation du contenu de ce service est présenté dans la Figure 82. Il comporte deux pages de sommaire, l'une pour les fichiers audio et l'autre pour les films disponibles. L'utilisateur peut choisir à tout instant ce qu'il souhaite regarder en utilisant le numéro correspondant de sommaire, mais il peut également naviguer parmi les objets pour obtenir plus d'information sur chacun d'entre eux (un extrait de film, par exemple).

Dès que l'utilisateur a fait son choix, le lecteur multimédia est démarré pour permettre l'exécution des commandes VCR sur l'objet choisi. Le lecteur multimédia utilise une interface modifiée, plus appropriée aux interfaces tout écran (la majorité des utilisateurs potentiels utiliseront des TV).

Si l'objet multimédia est dupliqué sur plusieurs serveurs (c'est souvent le cas), le lecteur va choisir automatiquement le serveur le moins chargé. Si le serveur choisi tombe en panne, le client continuera le transfert des blocs à partir d'un autre serveur contenant le même objet.

lorsque la visualisation d'un objet est terminée (ou lorsque le client arrête la présentation) le lecteur multimédia arrête son exécution et l'interface Web le remplace.

L'acquisition de contenu se fait à l'aide d'applications externes à notre système, fournissant les objets multimédia sous forme de fichiers dans des formats standards. Un exemple d'application externe est une application qui contrôle une carte TV ou des magnétoscopes.

Composants

Administration: Media Administrator (MA)

L'administration du système est confiée à un logiciel que nous nommerons MediaAdministrator (MA). Les principales fonctionnalités de ce logiciel sont :

la surveillance des utilisateurs connectés au système ; MA garde une trace dans un fichier log des opérations effectuées (visualisations des objets). Pour la surveillance de clients, MA écoute sur un port particulier (1001 par défaut) les messages UDP arrivant de ceux-ci. Le format de ces messages est décrit plus bas.

la gestion de contenu comprenant (entre autres) :

- l'ajout un objet (plusieurs sources possibles : fichier ou logiciel qui pilote différents périphériques, comme des magnétoscopes, des cameras vidéo, des antennes satellite etc.)
- la suppression ou modification des paramètres d'un objet
- la duplication d' un objet sur plusieurs serveurs
- la création des pages Web à partir de la liste d'objets disponible.

Chaque client communique avec un MA (le système peut en compter plusieurs) à travers UDP, en envoyant régulièrement un message pour l'informer de son état. Ce message a le format:

- Client: client_name
- Object: object_name
- Server: server_name
- State: state_type (OPEN, PLAY, PAUSE, CLOSE)
- CR+LF

Pour gérer les objets disponibles sur les serveurs Web, MA utilise un fichier appelé *catalogue*. Ce fichier catalogue contient un certain nombre d'informations sur chaque objet disponible: format, taille, durée, type, description, etc.

Les sources de données gérées par MA sont des répertoires disponibles sur les serveurs Web (partagés avec MA à travers le système de fichiers de Windows). Chaque répertoire comporte un catalogue listant les objets qu'il contient.

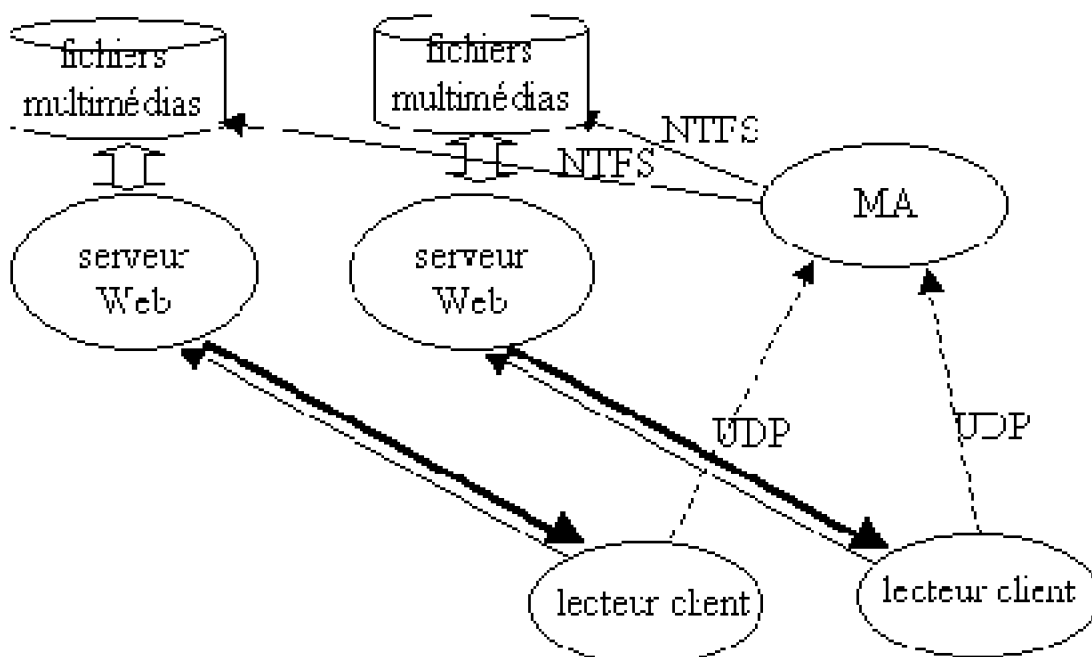


Figure 83 Composants communiquant avec le MediaAdministrator

Pour ajouter un objet multimédia, MA copie cet objet dans les répertoires gérés et met à jour les catalogues correspondants. La modification des caractéristiques d'un objet (comme la description, par exemple) a comme effet la modification du catalogue.

MA assure en temps réel la cohérence entre le catalogue et l'ensemble des pages Web permettant l'accès aux objets stockés sur les serveurs. A chaque objet correspond une page Web, liée aux autres pages par des liens hypertextes. Les pages sont dupliquées sur chaque serveur Web géré par MA.

Equilibrage de charges (load-balancing)

Ce composant est un service (nommé *ResourceSpy*) qui surveille le comportement des serveurs de contenu. Il connaît à chaque instant le niveau d'encombrement de chaque serveur et partage cette information avec les clients à travers un protocole spécifique.

ResourceSpy utilise l'interface Performance Data Helper (PDH) de Windows permettant de connaître les performances de différents drivers, applications ou services. Après une configuration préalable, *ResourceSpy* peut surveiller à travers PDH les performances des systèmes de stockage de tous les serveurs de contenu d'un réseau.

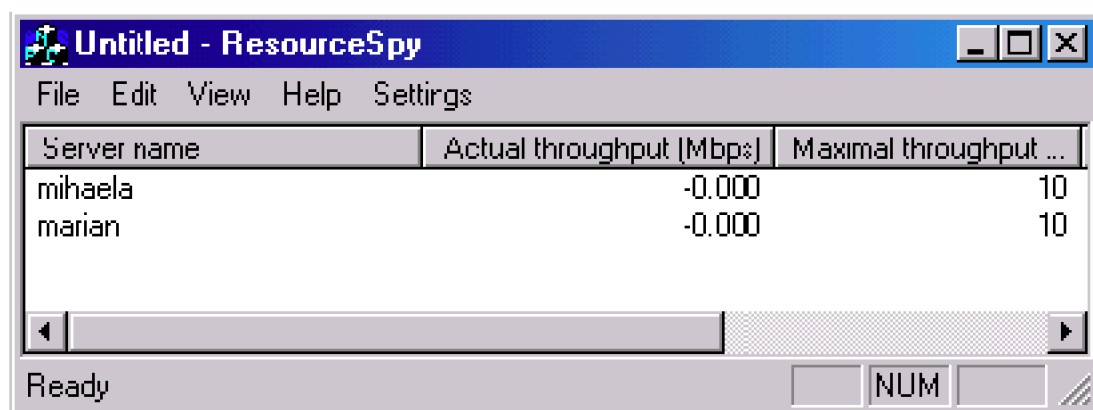


Figure 84 Module d'équilibrage de charges (ResourceSpy)

Le service utilise TCP (port 1000) pour la communication avec ses clients. Il envoie à chaque nouvelle connexion un message spécifiant le degré d'occupation de chaque serveur. Ce message respecte le modèle suivant :

- MA: MA_Address: MA_port; Server1: Server1_occupancy; Server2: Server2_occupancy;CRLF

Il fournit l'adresse d'un MA à utiliser, la liste des serveurs disponibles et leur niveau d'encombrement (pourcentage entre 0 et 100). Le message suivant :

- MA: soft03: 1001; soft15: 12; soft03: 0;

signifie par exemple que le réseau contient un serveur MA (soft03) écoutant sur le port 1001. Ce MA gère deux serveurs de contenu (soft15 et soft03), avec un niveau d'encombrement de 12% et 0%. Le client utilisera donc soft03 pour accéder au contenu.

Un système VoD peut comporter plusieurs composants d'équilibrage de charges pour pouvoir gérer d'une manière hiérarchique un nombre important de serveurs Web.

Lecteur multimédia

Le lecteur multimédia communique avec un ou plusieurs serveurs Web et avec le poste d'administration.

Il utilise la stratégie HTTP présentée dans cette thèse (Chapitre 5) à travers un filtre pour WMP ou RV; si le filtre est indisponible il peut travailler avec une autre stratégie HTTP, mais le système n'offrira pas dans ce cas des performances optimales (Chapitre 5).

Le filtre HTTP est lancé avec une adresse URL comme:

http://RServer/path/object.mpg

RServer représente l'adresse de la machine hébergeant le composant d'équilibrage de charge (ResourceSpy). Le filtre se connecte à ce logiciel qui lui fournit les noms de

serveurs Web contenant l'objet *object.mpg* et leur degré d'occupation. Le filtre utilisera le serveur le moins occupé (*server1*, Figure 85) pour retrouver l'objet *object.mpg*. Si le serveur *server1* n'est pas en mesure de répondre à la demande, le deuxième serveur le moins occupé (*server2*) sera utilisé pour le transfert par blocs de *object.mpg*.

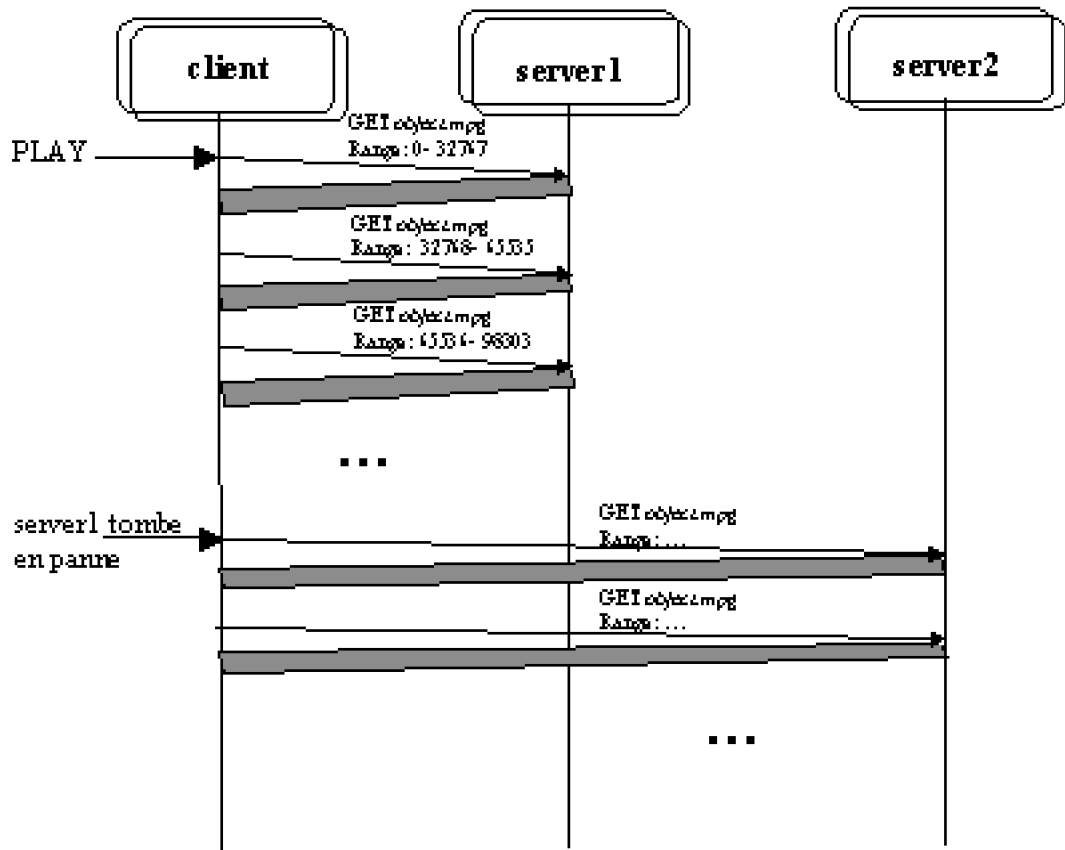


Figure 85 Utilisation de la stratégie HTTP "transfert par blocs" avec plusieurs serveurs Web. Si un serveur est indisponible, le client utilise un autre serveur contenant le même objet.

Nous avons effectué des tests montrant que le passage d'un serveur Web à un autre serveur contenant le même objet *object.mpg* n'influence pas la qualité de la perception au niveau utilisateur (le transfert entre les deux serveurs se fait en moins d'une seconde, 3 secondes de contenu étant disponibles dans la mémoire tampon).

Le composant d'équilibrage de charge (ResourceSpy) fournit aussi au filtre l'adresse de MA, où le filtre enverra les informations nécessaires pour la facturation et la surveillance du système.

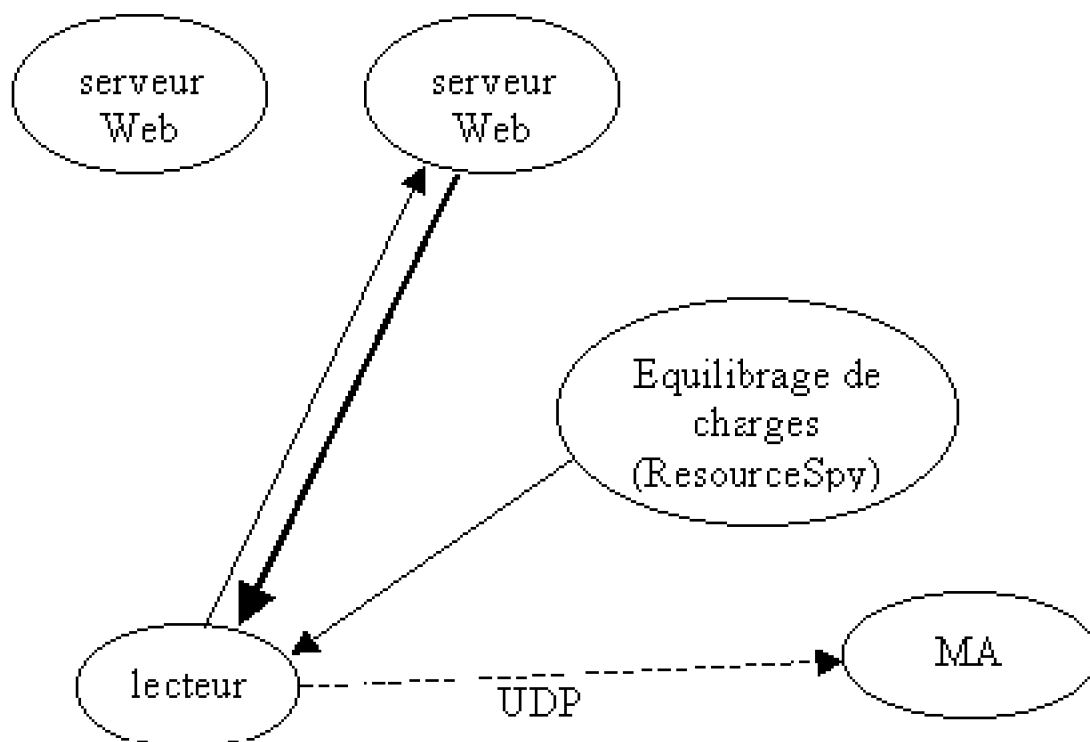


Figure 86 Composants communiquant avec le lecteur client

Dans le cas où ResourceSpy n'est pas joignable, le filtre considérera *RServer* comme le serveur Web où se trouve *object.mpg*, conformément à la sémantique http. Cette adresse reste valide si ResourceSpy est hébergé sur un serveur de contenu.

Avantages et inconvénients par rapport aux autres systèmes VoD

Fiabilité et haute-disponibilité

Les deux grands problèmes auxquels sont confrontés les systèmes VoD sont la stabilité des lecteurs multimédias et la robustesse des serveurs.

Le choix que nous avons fait d'utiliser des lecteurs multimédias existants (wmp et rv), nous garantit une bonne qualité de présentation. Leur intégration à notre système ne requiert que l'adjonction d'un module (Plug-In) permettant l'utilisation d'une stratégie HTTP de transfert par blocs (200 lignes de code C++) et la communication avec ResourceSpy et MediaAdministrator (moins de 100 lignes de code). La simplicité de cette approche nous a permis la réalisation d'un composant avec un risque très bas d'erreur.

Les serveurs Web choisis (IIS ou Apache) sont largement utilisés depuis plusieurs années sur Internet ([netcraft]), et la probabilité d'avoir une erreur est très limitée pour les

requêtes statiques que nous leurs adressons.

Le logiciel *ResourcesSpy* se trouve sur le poste administrateur, mais peut aussi être installé sur plusieurs serveurs de contenu. Dans ce cas, si une machine tombe en panne (serveur de contenu ou poste administrateur), les lecteurs multimédias sont capables de trouver automatiquement un autre serveur pouvant offrir le même service que le serveur défaillant. Cette possibilité est offerte par la stratégie de transfert par blocs : si une erreur apparaît sur le transfert d'un bloc, le même bloc et les suivants seront demandés à un autre serveur.

Intégration avec le Web

Le système présenté est accessible à travers tous les navigateurs Web et tous les lecteurs multimédias qui savent travailler avec HTTP, même si la stratégie utilisée n'est pas celle présentée dans le Chapitre 5.

Performances

Comme nous l'avons montré dans le Chapitre 4, un serveur Web utilisé comme serveur VOD permet d'obtenir un très bon rendement par rapport aux performances physiques d'un ordinateur. Ce rendement est du même ordre de grandeur que celui des serveurs VoD commerciaux actuels.

Au niveau client, le temps de réaction aux commandes VCR du client est très performant. Pratiquement, sur des réseaux locaux, les lecteurs travaillant avec la stratégie HTTP présentée se comportent mieux que d'autres lecteurs multimédias existants (voir Figure 61).

Exemple d'architecture concrète de système VoD

Un exemple concret d'architecture d'un système VoD basé sur des serveurs Web est présenté en Figure 87. Il s'agit d'un système conçu pour servir les clients dans un hôtel avec 800 chambres. Les chambres sont au nombre de 40 par étage, sur un total de 20 étages.

Sur chaque étage, les postes STB existants dans chaque chambre sont connectés sur un *switch* Gigabit-Ethernet, mais en utilisant des connexions Fast-Ethernet (100 Mbps). Sur le même étage (ou délocalisé dans une "salle machines") est placé un serveur de contenu (serveur Web), contenant aussi un module d'équilibrage de charges (ResourceSpy). Chaque lecteur client va accéder au système de stockage existant sur son niveau en utilisant le module d'équilibrage de charges.

Les modules d'équilibrage de charges établissent la connexion entre deux étages consécutifs. Si un serveur de contenu tombe en panne, ses clients peuvent être servis par

l'autre serveur sans dégrader les performances du système.

Les contenus de tous ces serveurs sont gérés par deux applications MediaAdministrator qui génèrent aussi des statistiques sur l'utilisation des serveurs.

La description du système est très schématique, en raison de la complexité de facteurs intervenant dans le fonctionnement du système (interface avec les logiciels de gestion de facturation etc.). Néanmoins un prototype de ce système a été testé avec de bons résultats par la société CSTI, il peut remplacer des solutions commerciales existantes.

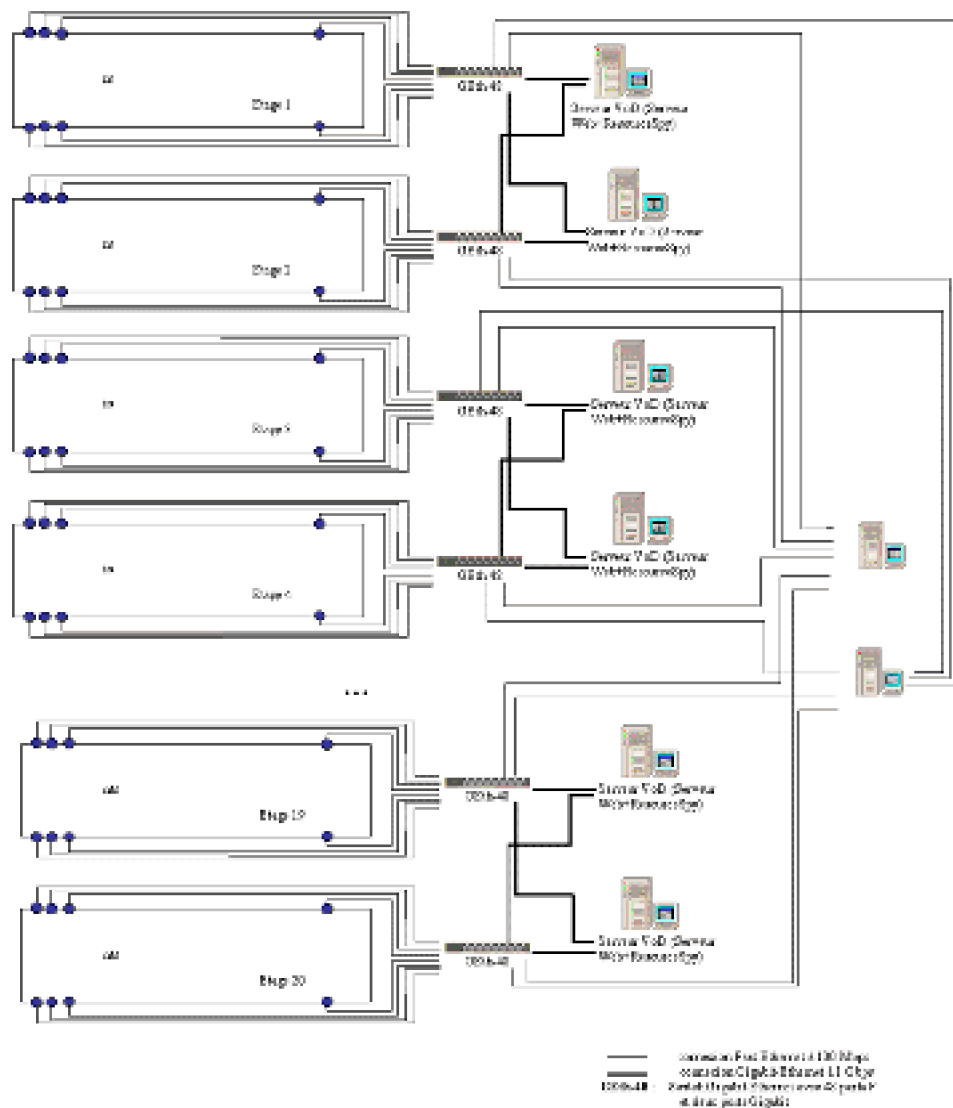


Figure 87 Système VoD basé sur des serveurs Web dans un grand hôtel