*REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE*

*MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SIENTIFIQUE*

*UNIVERSITE SAAD DAHLEB BLIDA*

*FACULTE DES SCIENCES*

*DEPARTEMENT D'INFORMATIQUE*

***MEMOIRE***

*Présenté pour l'obtention du grade de*

***MASTER***

*Spécialité : informatique*

*Option : Génie logiciel*



***Thème***

# *Proposition d'une extension sur NS-2 pour l'intégration des protocoles de méta-routage basé sur la redirection DNS dans les réseaux de distribution de contenu*



*Présenté par :*

- *MENASSERI Mohamed*
- *MEKHATI Alla Eddine*

*Promotrice :*

- *M^me^ AROUSSI Sana*

Président : M^me^ Rezoug

Examinateur : M^me^ Zahra

Examinateur : M^me^ Arkame

**2013/2014**

# REMERCIEMENTS MENASSERI

En préambule à ce mémoire, je souhaitais adresser mes remerciements les plus sincères aux personnes qui m'ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

Je tiens à remercier sincèrement **Mme Aroussi**, qui, en tant que Directrice de mémoire, s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu me consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Je remercie mes parents pour leur contribution, leur soutien et leur patience. Je n'oublie pas mes amis **Yacine**, **Aniss**, **Oussama** et **Islam** pour leur soutien, leurs conseils et encouragements.

J'exprime ma gratitude à tous les consultants et internautes rencontrés lors des recherches effectuées et qui ont accepté de répondre à mes questions avec gentillesse.

Enfin, j'adresse mes plus sincères remerciements à tous mes proches et amis, qui m'ont toujours soutenue et encouragée au cours de la réalisation de ce mémoire.

Je dédie ce modeste travail et ma profonde gratitude à **ma Mère** et **mon Père** pour l'éducation qu'ils m'ont prodigué ; avec tous les moyens et au prix de toutes les sacrifices qu'ils ont consentis à mon égard, pour le sens du devoir qu'ils mon enseigné depuis mon enfance.A ma chère sœur **Asmaa …**

Merci à tous et à toutes.

# REMERCIEMENTSMEKHATI

Le travail présenté dans ce mémoire de fin d'études a été effectué au sein de l'université de SAAD DAHLEB à Blida , je tiens à adresser mes vifs remerciements à tous les responsables de cette université,

Mes sincères remerciements vont à madame AROUSSI Sana notre promotrice de ce projet, qui a bien voulu nous accordé l'honneur de travailler avec elle, je la remercie pour la documentation mise à notre disposition, son aide précieuse et ses conseils tout au long de ce projet,

Je rends hommage à tous les enseignants que j'ai été élève pendant ces cinq ans d'études et les nombreux personnes qui ont contribué à la réalisation de ce travail de loin ou de près.

## *Dédicace*

*À ma Mère,*

*« Tu m'as donné la vie, la tendresse et le courage pour réussir.*

*Tout ce que je peux t'offrir ne pourra exprimer l'amour et la reconnaissance que je te porte.*

*En témoignage, je t'offre ce modeste travail pour te remercier pour tes sacrifices et pour l'affection dont tu m'as toujours entourée. »*

*À mon père,*

*« L'épaule solide, l'œil attentif compréhensif et la personne la plus digne de mon estime et de mon respect.*

*Aucune dédicace ne saurait exprimer mes sentiments, que Dieu te préserve et te procure santé et longue vie. »*

*À mes frères Merwan et Imad,*

*À mes sœurs Amira et Hiba,*

*À ma Famille,*

*À mes amis ...*

*Alla-Eddine*

# RÉSUMÉ

L'évolution des services vidéo de ces dernières années a provoqué l'invention de plusieurs applications vidéo récentes. Néanmoins, l'architecture internet n'a pas été conçue pour supporter ces types des services vidéo, à l'exemple de l'architecture client-serveur qui permet à l'utilisateur du réseau d'envoyer une requête de connexion au serveur de manière directe pour bénéficier de service vidéo. Cette architecture centralisée présente en effet plusieurs problèmes notamment le problème de passage à l'échelle face à des milliers d'utilisateurs. Pour remédier à ce problème, plusieurs autres architectures ont été proposées. Parmi celles-ci, nous trouvons l'architecture des réseaux de distribution de contenu (Content Distribution Network, CDN) où le contenu vidéo est répliqué du serveur d'origine vers plusieurs autres serveurs réplica, situés à proximité de l'utilisateur final. Les requêtes de l'utilisateur sont ensuite routées vers le meilleur service réplica ayant le contenu désiré.

Dans ce contexte, notre travail consistait à implémenter, sur le simulateur réseau NS-2, un protocole de méta-routage des requêtes dans les CDN de type vidéo. Une première version du protocole de méta-routage a été écrite dans [ARO, 12]. Elle permet de rediriger en temps réel les requêtes de l'utilisateur au niveau de la phase de résolution de noms (DNS) vers le meilleur serveur. Cependant, la mise en place de ce protocole sur NS-2 a fait ressortir plusieurs problèmes techniques, ce qui nous a poussés à revoir cette version et à introduire autre solution plus simple capable de simuler à la fois la redirection DNS et la transmission vidéo dans les réseaux de distribution de vidéo

**Mots clés :** services vidéo, protocoles méta-routage, réseaux de distribution de contenu, résolution de nom (DNS), simulateur réseau NS-2.

# ABSTRACT

The evolution of video services in recent years has led to the invention of several recent video applications. However, the Internet architecture was not designed to withstand these types of video services, like the client-server architecture that enables the network user to send a connection request to the server to direct to benefit from video service. This provides centralized architecture indeed several issues including the transition from wide problem facing thousands of users. To remedy this problem, several architectures have been proposed. Among these we find the architecture of content distribution networks (Content Distribution Network CDN) where the video content is replicated from the original server to several other replica servers, located close to the end user. User queries are then routed to the best replica service with the desired content.

In this context, our job was to implement, on the simulator NS-2 network, a meta-protocol routing queries in CDN video type. A first version of the meta-routing protocol is written in [ARO, 12]. It allows real-time redirect user requests at the stage name resolution (DNS) to the best server. However, the implementation of the protocol on NS-2 revealed several technical problems, which prompted us to review this version and introduce more simple solution that can simulate both DNS redirection and video transmission in the video distribution networks

**Keywords**:Video service, Routing protocoles, Content Distributing Networks, Domain Name Server (DNS), Network Simulator NS-2.

# ملخص

لقد أدى تطور خدمات الفيديو في السنوات الأخيرة لاختراع عدة تطبيقات الفيديو الأخيرة. ومع ذلك، لم يتم تصميم الهيكل الإنترنت على تحمل هذه الأنواع من خدمات الفيديو، مثل الهندسة المعمارية خدمة العملاء التي تمكن المستخدم شبكة لإرسال طلب اتصال إلى الملقم ل مباشرة من الاستفادة من خدمة الفيديو. وهذا يوفر بنية مركزية في الواقع العديد من القضايا بما في ذلك الانتقال من مشكلة واسعة تواجه الآلاف من المستخدمين. ولمعالجة هذه المشكلة، تم اقتراح عدة أبنية. ومن بين هذه نجد بنية شبكات توزيع المحتوى CDN) شبكة توزيع المحتوى (حيث يتم نسخ محتوى الفيديو من الملقم الأصلي إلى عدة خوادم نسخة أخرى، وتقع بالقرب من المستخدم النهائي. ثم يتم توجيه الاستفسارات للمستخدم أفضل الخدمات المتماثلة مع المحتوى المطلوب.

وفي هذا السياق، كانت مهمتنا أن تنفذ، على شبكة محاكاة NS-2، على بروتوكول الفوقية الاستفسارات التوجيه في CDN نوع الفيديو. يتم كتابة النسخة الأولى من بروتوكول التلوي التوجيه في [ARO، 12]. انها تسمح طلبات المستخدمين إعادة توجيه في الوقت الحقيقي في تحليل الاسم مرحلة (DNS) إلى أفضل خدمة. ومع ذلك، كشفت تنفيذ بروتوكول بشأن NS-2 العديد من المشاكل الفنية التي دفعتنا لمراجعة هذه النسخة وتقديم حل أكثر بسيط يمكن أن تحاكي كل من إعادة توجيه DNS والبث التلفزيوني في شبكات توزيع الفيديو

# TABLE DES MATIERES

# CHAPITRE III : SIMULATEUR DE RESEAU NS-2

# CHAPITRE IV : IMPLEMENTATION DU PROCOTOLE

# LISTE DES FIGURES

# LISTE DES TABLEAUX

# INTRODUCTION GENERALE

Le succès d'internet a donné naissance à une nouvelle génération de services vidéo comme la télévision IP, la vidéo conférence ou la vidéo à la demande (VoD). Ces services sont devenus très courants et font partie de beaucoup de recherches. Plusieurs offres commerciales existent pour la distribution de services vidéo comme les chaines numériques sur xDSL ou réseaux mobiles, les services de vidéo conférence sur mobiles de troisième génération. Les fournisseurs de ces services sont appelés à fournir à l'utilisateur final un meilleur service, cependant, l'architecture d'internet n'a pas été conçu pour supporter ce type de services. Il est donc nécessaire de développer des architectures et des protocoles réseaux capables de garantir la qualité de ces services. C'est dans ce contexte que s'inscrit notre problématique[ARO, 12].

A l'origine, les services vidéos sur Internet sont exécutés sur l'architecture traditionnelle client/serveur où l'utilisateur contacte le serveur vidéo et demande le flux vidéo approprié. Cependant, cette architecture centralisée présente plusieurs problèmes notamment le problème de passage à l'échelle face à des milliers d'utilisateurs. Pour remédier à ce problème, plusieurs autres architectures ont été proposées. Parmi celles-ci, nous trouvons l'architecture des réseaux de distribution de contenu (Content Distribution Network, CDN) où le contenu vidéo est répliqué du serveur d'origine vers plusieurs autres serveurs, appelés serveurs réplica, situés à proximité de l'utilisateur final. Les requêtes de l'utilisateur sont ensuite routées vers le meilleur service réplica ayant le contenu désiré. Ce meilleur serveur réplica peut être le serveur le plus proche, le moins chargé et/ou possédant une bonne connexion réseau en termes de paramètres de performance du réseau. Le but principal du routage de requêtes (appelé aussi méta-routage) est de garantir la satisfaction de l'utilisateur final[ARO, 12].

L'objectif principal de notre travail consiste à implémenter un protocole de méta-routage basé sur la redirection DNS dans les réseaux de distribution de contenu de type vidéo. Une première version du protocole de méta-routage a été écrite dans [ARO, 12]. Elle permet de rediriger en temps réel les requêtes de l'utilisateur au niveau de la phase de résolution de noms (DNS) vers le meilleur serveur. Pour mettre en place cette version, il est nécessaire d'introduire une extension sur le simulateur réseau NS-2 qui permet à la fois la redirection DNS et la transmission vidéo dans le CDN. Cela fait l'objet de notre travail.

Notre mémoire est structuré en deux parties, l'état de l'art et nos implémentations :

- L'état de l'art se compose des trois premiers chapitres. Le premier chapitre est consacré à la vidéo sur internet. Le deuxième chapitre est une synthèse de concepts nécessaires à la bonne compréhension du fonctionnement des réseaux CDN, notamment leur Système de Routage de Requêtes (SRR).Et le troisième chapitre contient les protocoles de méta-routage trouvés dans la littérature.
- La deuxième partie contient le quatrième et dernier chapitre, dans lequel nous présentons notre implémentation du protocole de méta-routage.

# CHAPITRE I

## La Vidéo Sur Internet

# INTRODUCTION

Au début, Internet offrait un service simple : se connecter plusieurs ordinateurs de façon simple et économique. Ces dernières années, les capacités des ordinateurs ont augmenté de façon rapide, les machines ont abouti le traitement des vidéos. Donc, Internet était obligé de s'évoluer pour pouvoir transmettre ce type de données. D'où la naissance de plusieurs services vidéo comme la visiophonie, la vidéo conférence, la télévision sur IP et VoD (Video on Demand). L'avancement technologique a impliqué la création de nouveaux types de compression vidéo (MP4/H.264, OGG/Theora, WebM/VP8…), qualité de vidéo (HD, Full HD, 4K) et aussi la vitesse de la connexion grâce à la bande passante de haut débit (ADSL, 3G, LTE, Wi-Fi, Li-Fi...) ce qui a amélioré les services vidéo sur internet. Selon [VNI, 13], le trafic vidéo sur Internet va représenter 69% de tout le trafic Internet grand public en 2017 (contre 57% en 2012).La vidéo en ligne consommée à la télévision a doublé en 2012. Elle va continuer à croître à un rythme rapide, et devrait quintupler d'ici 2017 pour représenter 14 % du trafic vidéo sur Internet. Le trafic vidéo à la demande devrait presque tripler d'ici 2017. La quantité de trafic VoD en 2017 sera équivalente à 6 milliards de DVD par mois. Donc, il faudrait à une personne plus de 5 millions d'années pour visionner la quantité de vidéo qui va passer chaque mois par les réseaux IP en 2017. Chaque seconde, près d'un million de minutes de contenu vidéo transiteront sur le réseau

Nous allons parler dans ce chapitre de la vidéo sur internet. Nous commençons par définir la vidéo. Ensuite, nous expliquons les services de streaming vidéo, leurs méthodes et leurs architectures.

# I NOTIONS DE BASE

La **vidéo** regroupe l'ensemble des techniques permettant l'enregistrement ainsi que la restitution d'images animées, accompagnées ou non du son, sur un support électronique et non de type photochimique [JCF, 07].

Techniquement, un flux vidéo est composé d'une succession d'images qui défilent à un rythme fixe, pour donner l'illusion du mouvement [FPV, 06]. Chaque image est décomposée en lignes horizontales, chaque ligne étant une succession de points. La lecture et la restitution d'une image s'effectue donc séquentiellement ligne par ligne comme un texte écrit : de gauche à droite puis de haut en bas[PHB, 06].

Dans un réseau de communication tel que Internet, Jiong Sun a expliqué la relation entre le système vidéo, l'application vidéo et le service vidéo [JSU, 06], comme illustré dans la Figure I.1.

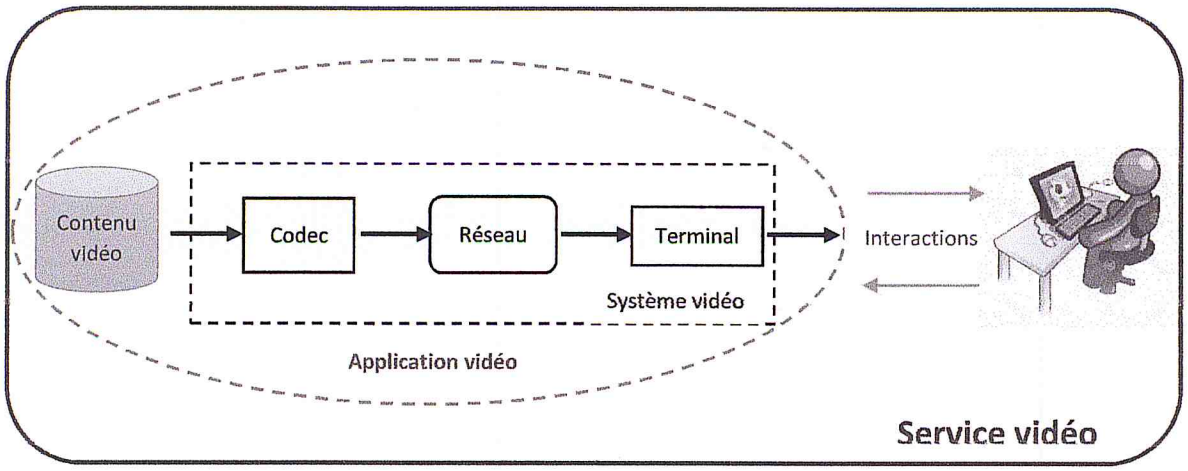



Figure I.1. Système, application et service vidéo [JCF, 07]

Ainsi, **un système vidéo** typique se compose de trois blocs fonctionnels distincts, dans l'ordre de la manière dont les flux vidéo partent de fournisseur de contenu à des utilisateurs finaux :

- Codec : pour le codage et décodage de la vidéo.
- Réseau : pour la transmission de la vidéo.
- Terminal : de l'utilisateur pour l'affichage de la vidéo

En ajoutant un contenu vidéo à ce système vidéo ça devient **une application vidéo. Un service vidéo**englobe toute interaction entre un utilisateur final et une application vidéo.

## DIFFUSION DE LA VIDEO SUR INTERNET

Pour diffuser une vidéo sur Internet, il y'a deux méthodes connues, le téléchargement ou le streaming. La première méthode consiste à envoyer la vidéo en entier avant de la visionner au niveau du client. Ce téléchargement peut être réalisé grâce aux protocoles communs comme TCP/FTP ou TCP/HTTP [ASM, 12]. Cette méthode possède plusieurs inconvénients :

- Un temps de téléchargement très long : l'utilisateur doit patienter jusqu'à la fin de téléchargement pour pouvoir visionner la vidéo.
- Une occupation de la mémoire importante : dans le cas des fichiers volumineux ça devient un problème et peut causer un excès de mémoire au niveau de disque dur.
- La qualité et type du contenu ne peuvent être vérifiés qu'à la fin de téléchargement et ça peut jouer à la satisfaction de l'utilisateur.

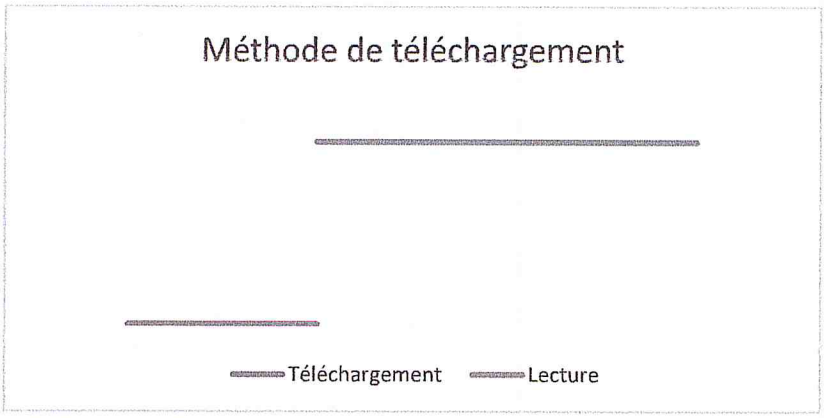



Figure I.2 Méthode de téléchargement [ATW, 02]

La deuxième méthode, Streaming vidéo, est une solution alternative pour résoudre les problèmes rencontrés dans la solution de téléchargement. Elle permet en effet la livraison et la lecture de la vidéo en même temps mais avec un délai entre le début de la livraison et le début de la lecture [ATW, 02], appelé le délai de pré-enregistrement ou bien le temps de démarrage (Figure I.4). L'utilisateur peut ainsi visionner la vidéo avant la fin de la livraison de la vidéo ce qui donne à cette méthode les avantages suivants :

- Un faible retard entre le début de la livraison et le début de la lecture.
- Un gain de mémoire : faible espace de stockage car une petite portion de la vidéo qui va être stockée au niveau du client.

- Vérification de la qualité et type du contenu : le client peut consulter la vidéo dès le début ce qui lui permet de voir la qualité et le contenu.

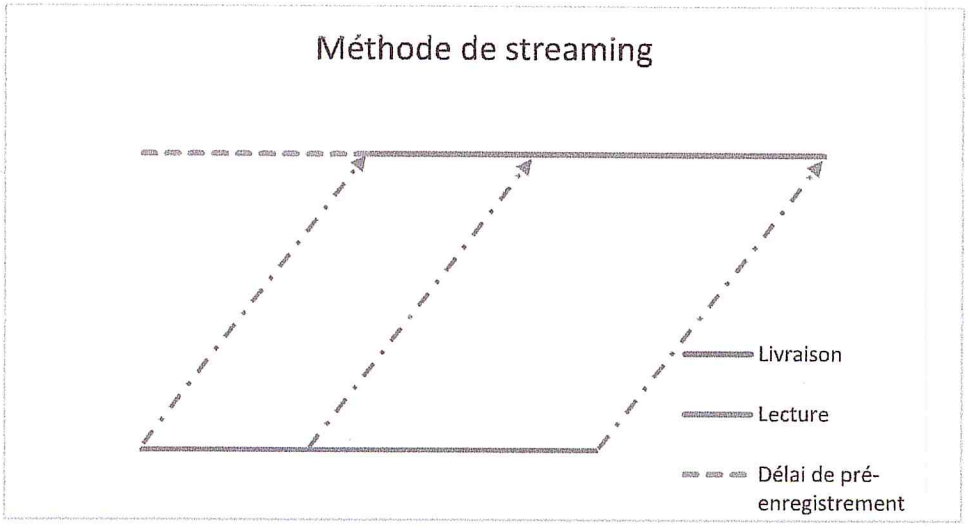



Figure I.3 Méthode de streaming [ATW, 02]

Dans ce qui suit, nous détaillons plus le principe de fonctionnement de cette méthode.

## STREAMING VIDEO

Le streaming (terme anglais, de Stream : « courant », « flux », « flot »), lecture en continu, diffusion en flux, lecture en transit ou diffusion en mode continu, désigne un principe utilisé principalement pour l'envoi de contenu en « direct » (ou en léger différé). Très utilisée sur Internet, elle permet la lecture d'un flux vidéo à mesure qu'il est diffusé. Comme décrit précédemment, elle s'oppose à la diffusion par téléchargement de fichiers qui nécessite de récupérer l'ensemble des données d'un morceau ou d'un extrait vidéo avant de pouvoir le regarder. Néanmoins la lecture en continu est, du point de vue théorique, un téléchargement car il y a un échange de données brutes entre un client et un serveur, mais le stockage est provisoire et n'apparaît pas directement sous forme de fichier sur le disque dur du destinataire. Les données sont téléchargées en continu dans la mémoire vive (RAM), sont analysées à la volée par l'ordinateur et rapidement transférées dans un lecteur multimédia (pour affichage) puis remplacées par de nouvelles données.

### III.1 Principe de Fonctionnement

La lecture en continu fonctionne selon le protocole client-serveur. Le contenu est mis à disposition sur un serveur. Le client souhaitant accéder au contenu envoie une requête pour en récupérer une petite partie, à l'endroit du contenu où il souhaite commencer la lecture. La réponse est placée dans une mémoire tampon. Lorsqu'il y a suffisamment de données

dans cette mémoire pour permettre de lire le début du fichier audio ou vidéo, la lecture démarre. En arrière-plan, le téléchargement du flux se poursuit afin d'alimenter sans cesse la mémoire tampon avec la suite du fichier [WHY, 01]. La Figure I.5 montre le diagramme de processus de streaming vidéo.

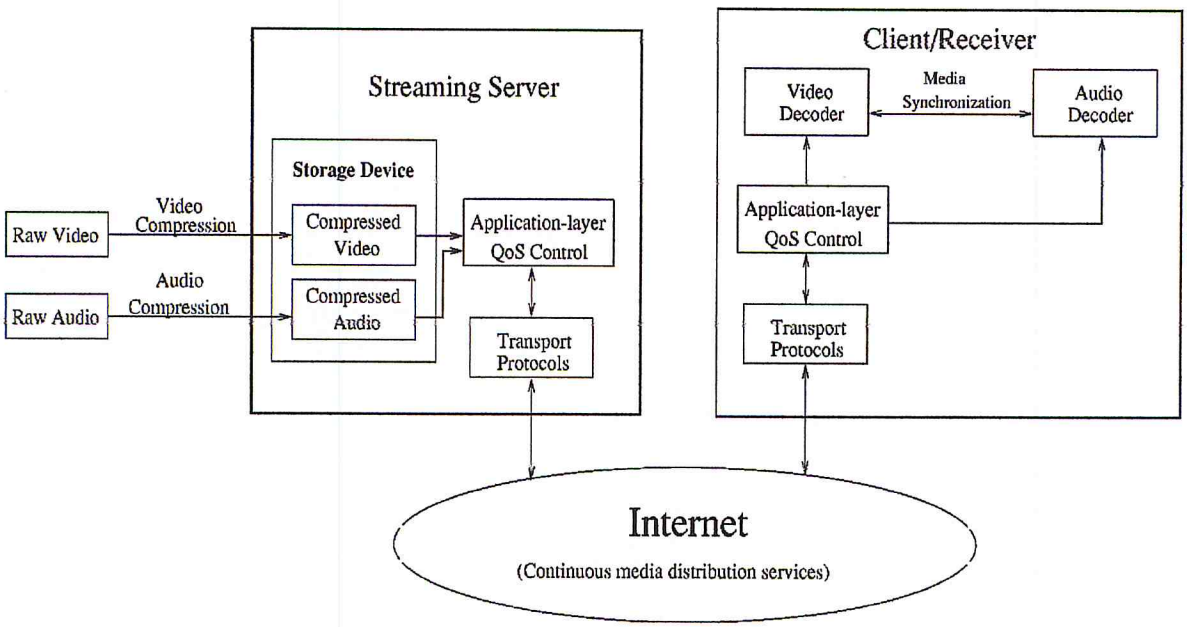



Figure I.4. Diagramme de processus de streaming vidéo [WHY, 01]

## III.2 Approches de Streaming

Le streaming peut être classé en trois grands types d'approches en fonction du type de contenu et de la façon dont il est mis à la disposition du public.

### III.2.1 Streaming Progressif ou « Statique »

C'est la solution la plus couramment utilisée, car elle s'effectue à partir d'un serveur web « standard » et utilise les protocoles HTTP et FTP basés sur TCP. Elle ne nécessite donc pas de serveur spécialisé.

Elle consiste à lire progressivement le fichier multimédia pendant son téléchargement. Après un temps de latence nécessaire au chargement des premières secondes, celles-ci sont lues tandis que la suite du fichier se charge. Le fichier est donc simplement proposé au téléchargement, de la même manière que tout autre type de fichier, et c'est le navigateur ou client (lecteur multimédia) qui se charge d'effectuer le streaming. La copie du fichier téléchargé est détruite par le navigateur à la fin du traitement. Elle permet de délivrer tout

type de médias mais préalablement préenregistrés : elle ne peut pas transmettre de flux en temps réel [BTM, 08].

L'inconvénient de cette solution est l'impossibilité de s'adapter à la qualité de connexion de l'utilisateur. Pour le diffuseur, il devient ainsi souvent nécessaire de proposer plusieurs fichiers avec des résolutions différentes pour permettre à l'internaute de choisir en fonction des capacités de sa connexion. Par ailleurs, elle induit une attente pour avoir les premières images si le fichier est de taille importante. Le progressif est la méthode utilisée par des sites comme Dailymotion ou YouTube [UPD, 13].

### III.2.2 Streaming Continu ou « Dynamique »

Elle nécessite un serveur spécialisé, appelé serveur de streaming, et utilise le protocole RTP / RTCP sur UDP pour diffuser le contenu.

Il n'y a qu'un seul fichier diffusé contenant plusieurs fois les mêmes informations à différents niveaux de qualité, et c'est le serveur de lecture en continu spécialisé qui se charge de diffuser l'information adaptée. En fonction du débit de la connexion de l'internaute, le serveur sélectionne le niveau de qualité maximal pour une diffusion en temps réel. Le serveur est également capable de s'adapter automatiquement aux variations de la bande passante : si la connexion se détériore et que le taux de transfert baisse, le contenu est livré avec une moindre qualité afin d'éviter les interruptions de diffusion. Si en revanche, la connexion devient plus fluide, la qualité s'améliore. Le contenu démarre dès que l'utilisateur demande à y accéder sans délais [BTM, 08].

L'inconvénient de cette solution est de devoir utiliser un serveur spécialisé (Xiph Icecast, Real Helix Streaming Server, Windows Media Services, Adobe Flash Media Server, Quicktime Streaming Server, etc.) et que l'internaute doit avoir une bande passante adaptée au contenu envoyé, le contenu étant diffusé au même rythme que la lecture de l'internaute [UPD, 13].

### III.2.3 Streaming Adaptatif

C'est en fait une méthode de livraison vidéo hybride qui réagit comme du Streaming, mais qui est basée sur le HTTP-Progressive Download. Plus concrètement, la vidéo transmise est découpée en morceaux de 2 à 4 secondes qui sont transportés via le protocole HTTP. L'affichage de la vidéo est donc une récupération d'une longue série de petits bouts de fichiers reçus en téléchargement [BSC, 12].

Chaque morceau peut être hébergé sur un simple serveur Web et peut également tirer profit des caches HTTP existantes. De là, il est téléchargé par le client d'une façon linéaire

et au fur et à mesure, la vidéo est reconstituée en faisant jouer les morceaux de façon continue [WBM, 13].

La partie "adaptive" intervient lorsqu'une vidéo est encodée en plusieurs débits. Ainsi, pour chaque 2 secondes, une multitude de morceaux correspondant au débit associé seront générés [BSC, 12]. Lorsqu'ils sont parfaitement alignés et synchronisés, cette technique permet de faire varier la qualité de la vidéo affichée en fonction de la bande passante disponible, et ce, d'une façon transparente pour l'internaute sans jamais rompre l'expérience de visionnement.

## III.3 Les étapes de Streaming Vidéo

Le streaming est le traitement appliqué à un flux de données en temps réel transitant sur le serveur, ou à un montage vidéo ou à un fichier vidéo installés sur le serveur. Il procède en plusieurs étapes [WHY, 01] :

### III.3.1 Encodage

Afin de réduire le nombre de paquets de données à transmettre (et donc économiser la bande passante nécessaire) et permettre leur lecture en temps réel, les fichiers multimédias doivent être compressés dans un format de streaming du serveur : c'est l'encodage.

Cela suppose deux opérations : la compression à l'aide d'un **codec** et le multiplexage dans un conteneur (muxeur).

#### III.3.1.1 Compression avec un codec

Un **codec** (pour **co**mpression/**dec**ompression) est un algorithme de compression/décompression utilisé pour réduire la taille d'un flux ou d'un fichier (audio ou vidéo)[DMV, 05].

Il existe de nombreux formats de compression. On distingue deux types de compression :

- **La compression sans perte**

La compression sans perte se base sur la fréquence d'apparition de mots binaires dans le flux binaire représentant une image ou une source audio. Elle réduit la quantité d'information à transmettre aux clients en comptabilisant le nombre de fois qu'apparaît continuellement chacun des mots binaires les plus fréquemment rencontrés.

- **La compression avec perte**

La compression avec perte, quant à elle, réduit la quantité d'information à transmettre en dégradant sensiblement les données tout en conservant l'information la plus pertinente du média. Les formats de compression permettent d'obtenir des ratios avoisinant le 1/10 tout en conservant une qualité visuelle ou auditive.

La Figure I.6 montre les normes de compression vidéo et leur chronologie :

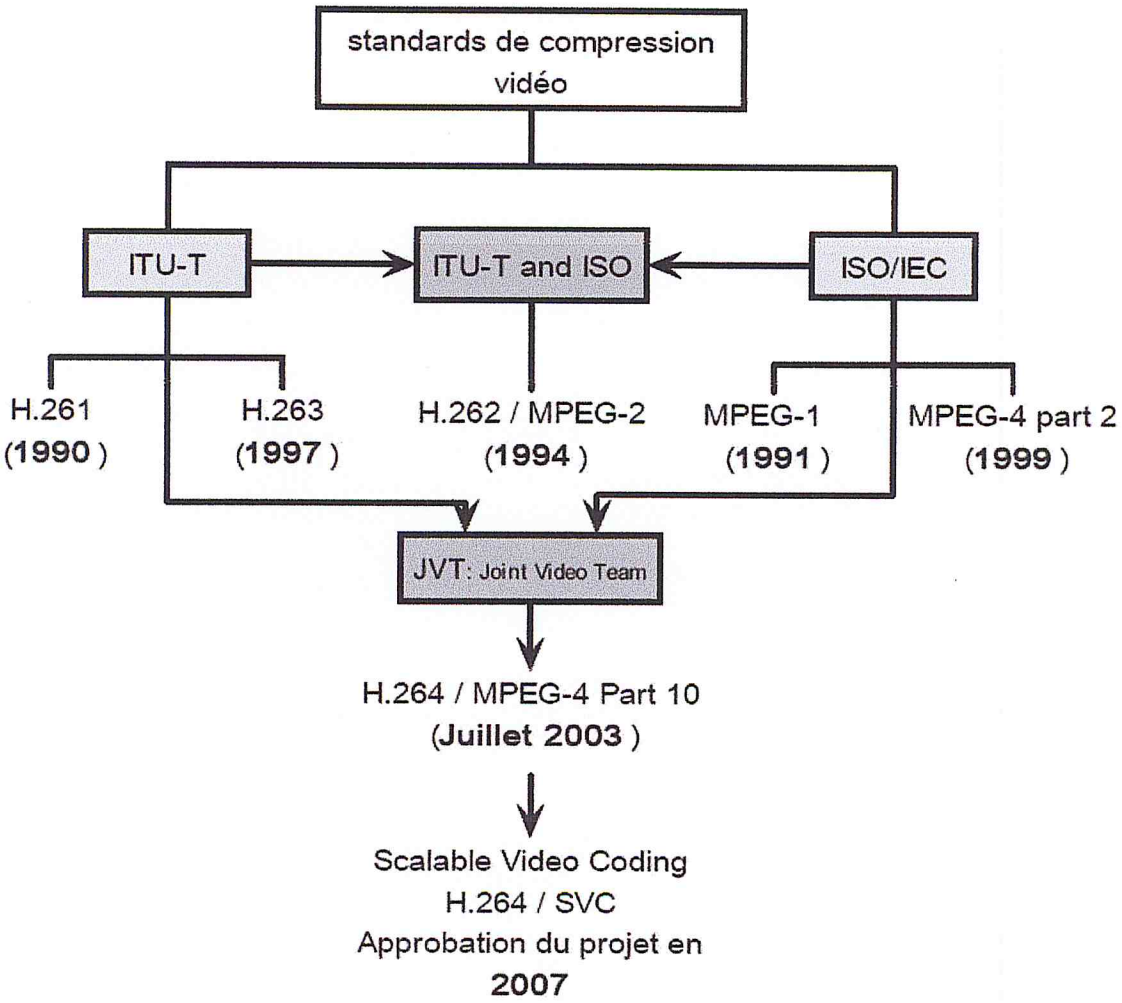



Figure I.5. Principales normes de compression audiovisuelles [AHA, 08]

### III.3.1.2 Multiplexage dans un Conteneur

Le multiplexage consiste à encapsuler (empaqueter) ensemble les différents flux requis dans un même fichier (conteneur) avant que celui-ci ne soit diffusé sur le réseau.

Un conteneur (ou muxeur) contient donc un ou plusieurs flux déjà encodés à l'aide de codecs, mais aussi d'autres informations qui définissent comment lire les données en indiquant le nom du codec nécessaire au décodage des flux audio et vidéo [WBM, 13].

Généralement, il y a un flux vidéo et un flux audio. Les formats conteneur les plus avancés sont capables de gérer de l'audio, de la vidéo, des sous-titres, des chapitres et des métadonnées (ou tags) et de façon synchronisée pour que les différents flux soient bien lus en même temps. WAV, AIFF (etc) sont des conteneurs audio (flux audio seulement). AVI, OGG, Quicktime, ASF, MP4 (etc) sont des conteneurs vidéo (flux audio et vidéo).

### III.3.2 Diffusion des Flux Vidéo sur le Réseau

Le conteneur vidéo est ensuite placé sur le serveur qui, à chaque requête d'un internaute, duplique le fichier demandé et le délivre sous la forme d'un flux continu de données (petits paquets de données marqués temporellement afin d'être réordonnancés de manière cohérente par le client).

A cause des fluctuations réseaux, des différents parcours empruntés par les paquets et des variations de la bande passante, les paquets n'arrivent pas toujours dans le bon ordre. Les paquets sont donc regroupés et agencés dans le bon ordre dans une mémoire tampon (ou buffer) créée par le lecteur média de l'ordinateur de l'utilisateur. Au bout de quelques secondes, une fois que le buffer de réception possède assez d'informations, la lecture du flux commence et les images ou le son sont retransmis. La mémoire tampon a donc pour rôle de fluidifier le flux. Si la connexion réseau est mauvaise, l'arrivé des paquets sera ralentie. Lorsque le buffer de réception est vide, la lecture s'arrête et reprendra lorsqu'elle possèdera assez de données pour continuer. L'image est alors figée.

### III.3.3 Lecture du Média

Les différentes opérations permettant la lecture du média sont assurées par le lecteur multimédia de l'utilisateur. Il dispose de plusieurs codecs à sa disposition pour assurer :

- **Démultiplexage**
  Le conteneur est tout d'abord démultiplexé : les différents flux audio et vidéo sont séparés et sauvegardés dans des fichiers différents.

- **Décompression**
  Chacun de ces flux sont ensuite décodés (décompressés) en temps réel avec les mêmes codecs que ceux utilisés pour les compresser. Ces flux sont alors restitués avec le maximum de qualité possible à l'utilisateur. Dans la plupart des cas, la compression est asymétrique : la compression (en fonction de la qualité voulue) sera plus longue et la décompression assez rapide pour permettre une lecture presque instantanée du flux.

## PILE PROTOCOLAIREDU STREAMING VIDEO

Très peu de protocoles ont été conçus et normalisés pour la communication entre les clients et les serveurs de streaming. Selon leurs fonctionnalités, les protocoles directement liés au streaming vidéo sur internet peuvent être classés dans les trois catégories suivantes [WHY, 01]:

- *Protocole de couche réseau* fournit un soutien de base de services de réseau tels que l'adressage de réseau. L'adresse IP sert comme protocole de couche réseau pour la vidéo en streaming sur Internet.
- *Protocole de couche de transport* fournit des fonctions de transport réseau de bout en bout pour les applications de streaming. Les protocoles de transport comprennent UDP (User Datagram Protocol) [POS, 80], TCP (Transmission Control Protocol), le protocole de transport en temps réel (RTP) [SCF, 03], et protocole de contrôle en temps réel (RTCP) [SCF, 03]. UDP et TCP sont des protocoles de transport de couche inférieure tandis que RTP et RTCP sont les protocoles de transport de couche supérieure, qui sont mises en œuvre sur le dessus de UDP / TCP.
- *Protocole de contrôle de couche session* définit les messages et les procédures pour contrôler la livraison de données vidéo au cours d'une session établie. Le RTSP [SRL, 98] et le protocole d'initiation de session (SIP) [HSR, 99] sont les protocoles de contrôle de session.

Pour illustrer la relation entre les trois types de protocoles, nous décrivons les piles de protocoles pour le streaming sur la Figure I.8. Pour le plan de données, sur le côté d'envoi, la vidéo / données audio compressée est récupéré et mis en paquets à la couche RTP. Les flux de paquets RTP fournissent un timing et des informations de synchronisation, ainsi que des numéros de séquences. Les flux de paquets RTP sont ensuite transmis à la couche TCP/UDP et la couche IP. Les paquets IP obtenus sont transportés sur l'Internet. Sur le côté du récepteur, les flux de vidéo sont traités de la manière inversée avant leurs présentations. Pour le plan de contrôle, les paquets RTCP et paquets RTSP sont multiplexés à la couche UDP/TCP et se déplacent à la couche IP pour la transmission sur Internet.

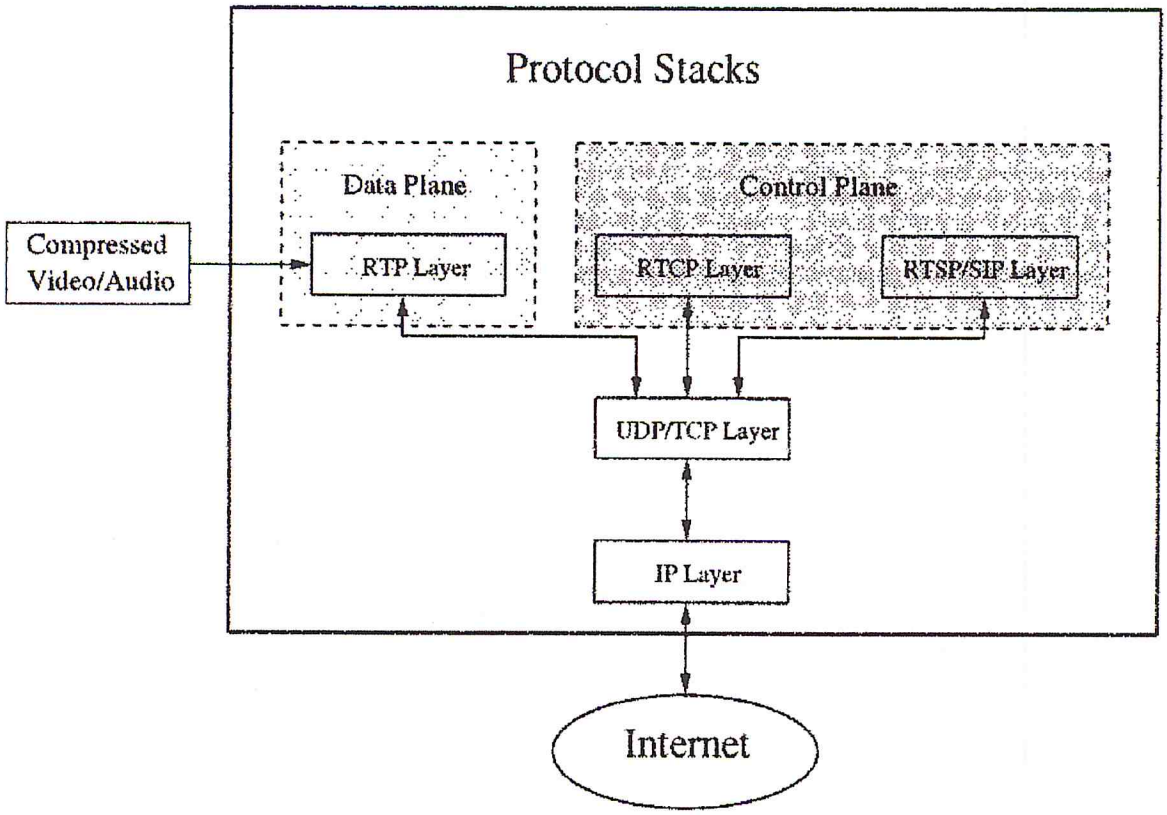



Figure I.6. Protocoles utilisés pour le streaming vidéo [WHY, 01].

Ces protocoles seront décrits dans les sections suivantes.

## IV.1 Le protocole IP

Conçu à l'origine pour des applications qui n'étaient pas multimédia, Internet repose à la base sur deux protocoles IP et TCP. Le protocole IP assure l'acheminement des paquets de point en point, jusqu'au terminal final mais sans se préoccuper du contenu. Il ne gère pas les pertes et les retards. Ce protocole simple mais peu fiable est complété par des protocoles de couches supérieure (comme le protocole TCP) pour assurer la fiabilité de la transmission. Entre autre, il supporte les trois modes de transmission utilisés pour les services streaming :

### IV.1.1 La transmission Unicast

La transmission Unicast ou point à point [PGA, 08] correspond au concept de la vidéo à la demande : à chaque requête d'un utilisateur unique correspond un flux vidéo spécifique qui est délivré par le serveur. C'est une technique de liaison de type point à point. Au niveau du serveur et des axes principaux de diffusion, il y aura sur le réseau autant de flux que de stations ayant envoyé une requête. Il est évident que si tous les utilisateurs visionnent simultanément le même contenu audiovisuel (transmission d'un évènement en direct), tous

les flux sont identiques. Il y a dans ce cas un gâchis manifeste de bande passante. La bande passante totale disponible sur le réseau est partagée entre tous les utilisateurs. La majorité du trafic Internet aujourd'hui est de ce type. Avantage pour l'internaute, il peut visualiser le contenu de son choix au moment de son choix. A la manière d'un magnétoscope, toutes les formes d'interactivité sont possibles (retour arrière, pause…).

### IV.1.2 La transmission Multicast

Une autre approche appelée multicast est de transmettre simultanément le même fichier à tous les internautes qui le souhaitent. A l'opposé de la méthode précédente où pour 1000 connectés il fallait envoyer 1000 fois le même fichier, avec le multicast, on n'assure qu'une seule émission du fichier vidéo. Le multicast s'apparente plus à la diffusion télévisuelle classique telle que nous la connaissons par ailleurs (hertzienne, par satellite…). La diffusion est réalisée à un instant donné, le même pour tous, sans aucune interactivité possible. Pour visualiser la séquence, il suffit de se connecter au moment ad hoc à une adresse IP spécifique. Ce procédé permet d'optimiser la bande passante du réseau, puisque le serveur ne génère qu'un seul flux, qui est ensuite dupliqué si nécessaire au niveau de chacun des nœuds du réseau. La technologie multicast offre une optimisation maximum : un seul flux au départ du serveur, un seul flux par banche jusqu'aux terminaux utilisateurs. L'avantage est évident : on économise de la bande passante sur toutes les artères principales du réseau même si des milliers d'utilisateurs sont connectés puisque l'on ne rediffuse pas à chaque fois toutes les données. Le nombre de connectés est indifférent pour le serveur : il n'émet qu'une seule copie.

Cette méthode convient parfaitement lorsqu'il s'agit de transmettre régulièrement des informations à un grand nombre d'abonnés (bourse, météo...), ou pour la diffusion d'événements en direct… Les utilisateurs finaux n'ont pas le choix du moment où ils veulent visionner la séquence puisque le flux est unique et qu'il sera de toute façon émis à l'horaire prévu. Cette méthode nécessite des équipements spécifiques sur le réseau : tous les routeurs doivent permettre le multicast, ce qui n'est pas le cas actuellement.

### IV.1.3 La transmission Anycast

La transmission Anycast [PMM, 93] où le flux vidéo généré par le serveur ne sera remis qu'à un seul membre du groupe, à la différence de la transmission multicast où le flux est remis à tous les membres du groupe. Ce paradigme de transmission a été conçu spécialement pour supporter la sélection d'un serveur parmi un groupe de serveurs dans lesquels le contenu vidéo est répliqué.

## IV.2 Les protocoles TCP et UDP

TCP fait partie de la couche supérieure du protocole IP, il complète ce dernier en demandant la réémission des paquets perdus ou détruits. Du fait de cette procédure de réémission, TCP est un protocole lent. Cette fiabilité qui est un atout pour la transmission de fichiers « informatiques » devient, du fait de cette relative lenteur, un handicap pour la transmission de la vidéo et de l'audio : lors d'une diffusion en temps réel, il est difficile d'attendre que tous les paquets perdus ou détruits soient réexpédiés.

UDP (User Datagram Protocol) [POS, 80] est un protocole simple permettant de transmettre les données vidéo très rapidement (en temps réel). Pour ce faire, il fonctionne en mode non-connecté, sans contrôle d'erreurs, sans remise en ordre des paquets à l'arrivée, sans réémission des données perdues et sans procédure d'acquittement. Néanmoins, les traitements des erreurs et le ré-ordonnancement peuvent être effectués par d'autres protocoles de haut niveau comme RTP, H.264/MPEG-4 AVC, ….

Les protocoles UDP et TCP supportent des fonctions telles que :

- *Multiplexage.* UDP et TCP peuvent multiplexer des flux de données à partir de différentes applications en cours d'exécution sur la même machine avec la même adresse IP.
- *Contrôle d'erreur.* Si un seul ou plusieurs bits des erreurs sont détectés dans le paquet entrant, la couche UDP/TCP rejette le paquet de telle sorte que la couche supérieure (par exemple, RTP) ne recevra pas le paquet corrompu. D'autre part, contrairement à UDP, TCP utilise la retransmission pour récupérer les paquets perdus. Par conséquent, TCP assure la transmission fiable tandis qu'UDP ne l'assure pas.
- *Contrôle de congestion.* Une autre caractéristique qui distingue TCP de UDP. En effet, TCP utilise le contrôle de congestion pour éviter d'envoyer trop de trafic, ce qui peut causer de la congestion du réseau.
- *Contrôle de flux.* Le protocole TCP emploie le contrôle de flux pour empêcher le débordement du tampon du récepteur alors qu'UDP ne dispose pas d'un mécanisme de contrôle de flux.

Depuis la retransmission TCP a introduit des retards qui ne sont pas acceptables pour les applications de streaming avec des exigences de retard strictes, UDP est généralement utilisé comme protocole de transport pour les flux vidéo. En outre, depuis UDP ne garantit pas la livraison des paquets, le récepteur donc a besoin de s'appuyer sur la couche supérieure (RTP) pour détecter la perte de paquets.

## IV.3 Le protocole RTP

Le protocole RTP (Real-time Transport Protocol) [SCF, 03] est un protocole de transfert des données (vidéo) de bout en bout. RTP fournit les fonctions suivantes à l'appui de streaming vidéo :

- *Time-Stamping* (Horodatage) : RTP fournit horodatage pour synchroniser les différents flux vidéo. On note que RTP en soi n'est pas responsable de la synchronisation, qui est laissée aux applications dédiées. Le Time-Stamp (référence temporelle) indique l'instant exact d'émission du paquet à la source permettant ainsi à l'arrivée d'ordonner les paquets, et de rétablir la régularité temporelle (RTP permet ainsi d'assurer une gigue inférieur à 40 ms).
- *Numérotation de séquence* : Depuis que les paquets arrivant sur le récepteur peuvent être hors de séquence (UDP ne fournit pas de paquets dans l'ordre), RTP emploie la numérotation de séquence pour placer les paquets RTP entrants dans l'ordre correct. Le numéro de séquence est également utilisé pour la détection de perte de paquets.
- *Identification de type de contenu* : Le type de contenu chargé dans un paquet RTP est indiqué par un champ d'en-tête RTP appelé identificateur de type de charge utile (payload type identifier). Le récepteur interprète le contenu du paquet en fonction de l'identifiant de type de contenu. Certains types de contenu courants tels que MPEG-1/2 audio et vidéo ont été attribués à des numéros de type de contenu. Pour d'autres types, cette affectation peut être faite avec des protocoles de contrôle de session.
- *Identification de source* : La source de chaque paquet RTP est identifiée par un champ d'en-tête RTP appelé identificateur de source de synchronisation (Synchronization SouRCe identifier SSRC), qui fournit un moyen au récepteur pour distinguer les différentes sources.

## IV.4 Le protocole RTCP

RTCP (Real-time Transport Control Protocol) [SCF, 03] est le protocole compagnon de RTP, il est conçu pour fournir la qualité de service (Quality of Service, QoS) aux participants d'une session RTP. En d'autres termes, RTP est un protocole de transfert de données pendant que RTCP est un protocole de contrôle.

RTCP permet d'envoyer périodiquement des paquets RTCP contenants des informations sur les participants et sur la qualité de transmission des données. Les informations sur la

qualité de transmission comprennent essentiellement le taux de perte, la gigue et le délai de bout en bout des paquets RTP. RTCP fournit les services suivants [WHY, 01] :

- *Informations sur la qualité de service (QoS feedback)* : C'est la fonction principale de RTCP. RTCP fournit des informations à une demande en ce qui concerne la qualité de la distribution des données. Le retour d'information est sous forme de rapports d'expéditeur (envoyés par la source) et des rapports de réception (envoyés par le récepteur). Les rapports peuvent contenir des informations sur la qualité de réception, tels que :

    1. fraction des paquets RTP perdus, depuis le dernier rapport.
    2. le nombre cumulé de paquets perdus, depuis le début de la réception.
    3. gigue inter-arrivée de paquets.
    4. le délai écoulé depuis la réception du rapport de la dernière expéditeur.

- *Identification du participant* : Une source peut être identifiée par le champ SSRC dans l'en-tête RTP. Malheureusement, l'identificateur SSRC n'est pas pratique pour les utilisateurs humains. Pour remédier à ce problème, le RTCP fournit un mécanisme familier par l'humain pour l'identification de la source. Plus précisément, RTCP SDES (Source DEScription), des paquets qui contiennent des informations textuelles appelées noms canoniques comme des identificateurs uniques globaux des participants à la session. Il peut inclure le nom d'un utilisateur, numéro de téléphone, adresse e-mail et d'autres informations.

- *Mise à l'échelle des paquets de contrôle* : Pour mettre à l'échelle la transmission des paquets de contrôle RTCP avec le nombre des participants, un mécanisme de contrôle est conçu comme suit : Le mécanisme de contrôle maintient le total des paquets de contrôle à 5% de la largeur de bande totale de la session. Parmi les paquets de contrôle, 25% sont alloués à des rapports d'expéditeur et 75% à des rapports de récepteur. Pour éviter la famine des paquets de contrôle, au moins un paquet de contrôle est envoyé dans les 5s à l'expéditeur ou le destinataire.

- *Synchronisation inter-média* : Les rapports d'expéditeur RTCP contiennent une indication de temps réel et de l'horodatage RTP correspondant. Elle peut être utilisée dans la synchronisation inter-média comme la synchronisation labiale en vidéo.

- *Informations minimales de contrôle de session* : Cette fonctionnalité optionnelle peut être utilisée pour transporter des informations de session telles que les noms des participants.

## IV.5 Le protocole RTSP

Le protocole RTSP (Real-Time Streaming Protocol) [SRL, 98] est un protocole de contrôle de session pour le streaming média (vidéo/audio) sur Internet. RTSP fonctionne comme « une commande à distance » du réseau pour les serveurs streaming. L'une des principales fonctions de RTSP est de supporter les opérations de contrôle tout comme le magnétoscope (VCR) tel que stop, pause/reprise, avance rapide et retour rapide. En outre, RTSP fournit également des moyens pour choisir les canaux de distribution (par exemple, UDP, UDP multicast ou TCP), et les mécanismes de livraison basés sur RTP. RTSP fonctionne pour le multicast et unicast. Dans la figure I.11 on peut voir l'utilisation des protocoles pendant un streaming vidéo.

Une autre fonction principale de RTSP est d'établir et de contrôle des flux audio continu et supports vidéo entre les serveurs de médias et les clients. Plus précisément, RTSP fournit les opérations suivantes :

- *Récupération des médias* : Le client peut demander une description de la présentation, et demander au serveur de configurer une session pour envoyer les données vidéo demandées.
- *Ajout des médias à une session existante* : Le serveur ou le client peut aviser l'autre sur tous les supports média supplémentaires qui deviennent disponibles pour la session établie.

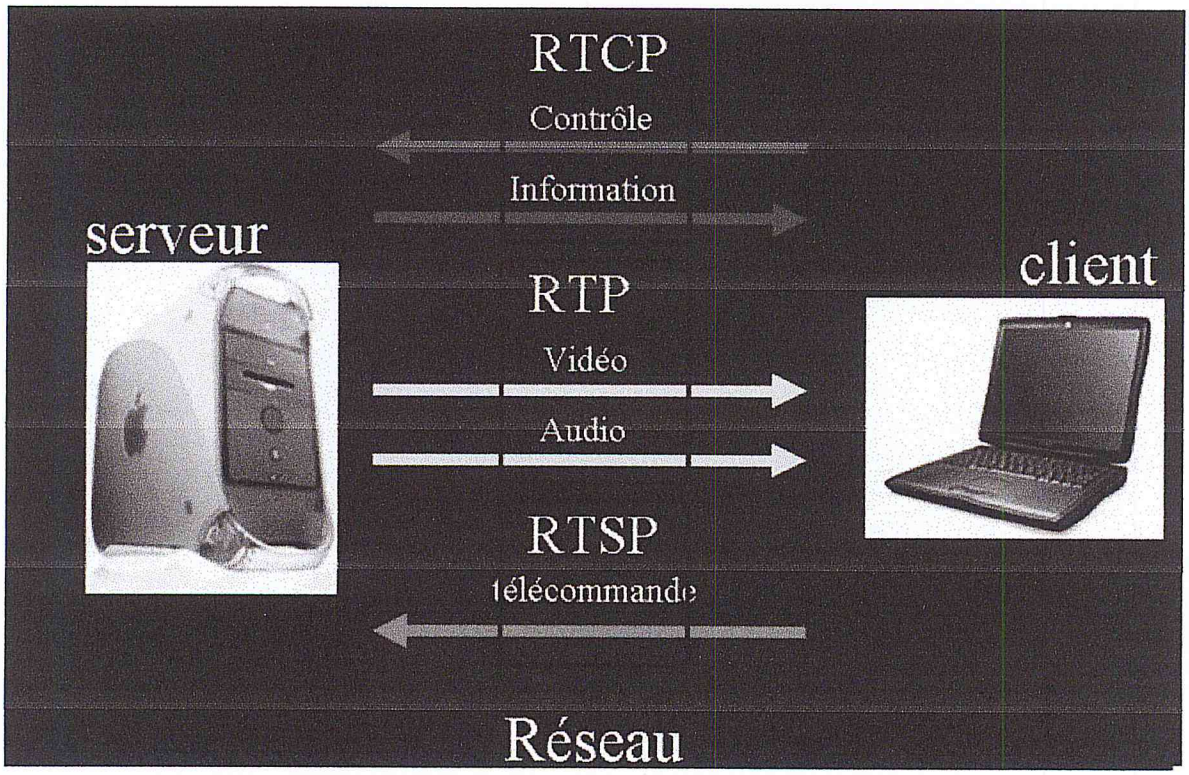


Figure I.7. Utilisation des protocoles pendant la diffusion vidéo [SRL, 98]

En RTSP, chaque flux de présentation et médias est identifié par un localisateur de ressources universel RTSP (URL). La présentation générale et les propriétés des médias sont définies dans un fichier de description de présentation, ce qui peut inclure le codage, le langage, les URL RTSP, adresse de destination, le port, et d'autres paramètres. Le fichier de description de présentation peut être obtenu par le client via le protocole HTTP, email, ou d'autres moyens.

## IV.6 Le protocole SIP

Le protocole SIP (Session Initiation Protocol) [HSR, 99] est un protocole de contrôle de session. Semblable à RTSP, SIP permet également de créer et terminer des sessions avec un ou plusieurs participants. Contrairement à RTSP, SIP prend en charge la mobilité des utilisateurs par mandatement et redirigeant les requêtes à l'emplacement actuel de l'utilisateur.

Pour résumer, RTSP et SIP sont conçus pour déclencher et la livrer directement des flux vidéo à partir de serveurs de streaming. RTP est un protocole de transport de streaming vidéo et RTCP est un protocole de contrôle de livraison de paquets RTP. UDP sont des protocoles de transport de couche inférieure pour les paquets RTP/RTCP/RTSP/SIP et IP fournit une plate-forme commune pour la livraison des paquets UDP sur Internet. La combinaison de ces protocoles fournit un service complet en streaming sur Internet. La Figure I.12 montre les protocoles spécifiques aux streaming vidéo.

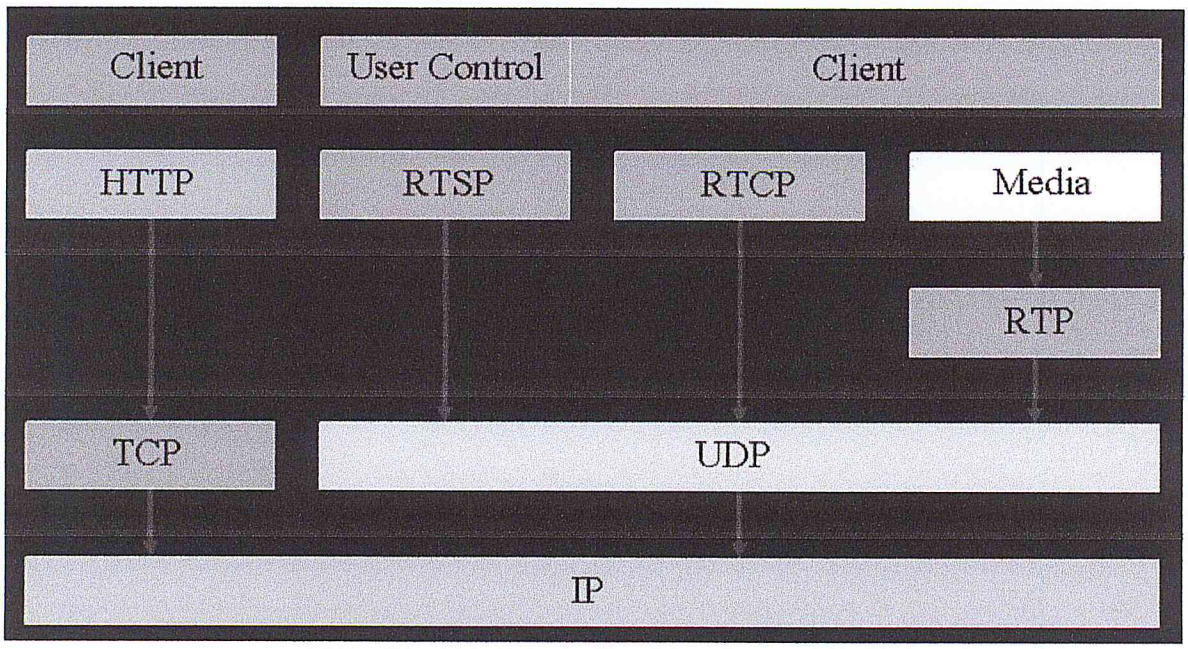



Figure I.8. Protocoles utilisés pour diffuser de la vidéo sur Internet [WHY, 01].

# LES ARCHITECTURES DE STREAMING VIDEO

A l'origine, les services vidéo sur Internet sont exécutés sur l'architecture traditionnelle client/serveur où le client contacte le serveur streaming et demande le flux vidéo approprié. Cependant, cette architecture centralisée présente plusieurs problèmes notamment : le problème de passage à l'échelle face à des milliers d'utilisateurs. Pour remédier à ce problème, plusieurs autres architectures ont été proposées. Parmi celles-ci, nous trouvons les architectures des réseaux de distribution de contenus (Content Distribution Network, CDN), les architectures des réseaux Pair à Pair (Peer-to-Peer, P2P) et les architectures hybrides (P2P/CDN).

## V.1 Les Réseaux de Distribution de Contenu

Dans les CDN, le contenu vidéo est répliqué du serveur d'origine vers plusieurs autres serveurs (appelés serveurs réplicas) situés à proximité des utilisateurs finaux. Ainsi, les demandes des utilisateurs sont dirigées vers le meilleur serveur réplica ayant le contenu désiré. Le meilleur serveur est défini par le serveur le plus proche, le moins chargé et/ou possédant une bonne connexion réseau avec le client en termes de latence ou de taux de perte [PEN, 08].

Une telle architecture à base de réplication de contenus remplace la communication client/serveur par deux flux de communication : l'un entre le client et le serveur réplica, et l'autre entre le serveur réplica et le serveur d'origine [PAV, 06]. Cette distinction en deux flux de communication permet (i) de réduire la consommation de bande passante sur les liaisons réseau, la latence pour les clients et la charge sur les serveurs streaming, et (ii) d'augmenter la disponibilité des contenus vidéo [PEN, 08]. Malgré sa performance, cette architecture souffre néanmoins d'un coût prohibitif et d'une gestion compliquée inhérentes aux opérations consistant à savoir placer les serveurs réplicas, à distribuer des copies du contenu vidéo à des serveurs réplicas et à router les requêtes vers le meilleur serveur réplica.

## V.2 Les Réseaux Pairs-à-Pairs

Dans les réseaux P2P, les utilisateurs s'organisent en groupes, en communautés ou comme des utilisateurs simples. Le concept de cette organisation repose sur le fait que chaque participant au réseau P2P peut jouer au même moment le rôle de client et celui de serveur. Les utilisateurs ne sont plus passifs, consommant simplement des flux vidéo en provenance de quelques serveurs streaming répartis sur Internet, mais ils sont actifs dans la mesure où ils participent à la création, à la génération et à la diffusion de contenus vidéo.

Les réseaux P2P se présentent comme une réelle alternative au modèle client/serveur, au regard de leur capacité à assurer la montée en charge, et leur déploiement rapide à faible coût. Cependant, le streaming vidéo sur les réseaux P2P présente des problèmes spécifiques, comparé au streaming en mode client/serveur, liés à l'organisation des nœuds dans le réseau, à la distribution du contenu entre les nœuds, à la sélection des nœuds, au comportement non déterministe des nœuds et à l'hétérogénéité des nœuds [AHA, 08].

## V.3 Les Architectures Hybrides

Afin de regrouper les avantages des architectures CDN et P2P tout en éliminant leurs limitations, des architectures hybrides sont apparues qui combinent les caractéristiques de ces deux façons de faire. Par exemple, PROP (collaborating and coordinating PROxy and its P2P clients) [GCZ, 06] est un système de streaming hybride proxy/P2P pour la VoD où le proxy sert comme site de cache assurant la disponibilité des segments vidéo qui ne peuvent être servis par aucun pair. Bien qu'une telle architecture hybride semble être très prometteuse, certains aspects doivent encore être pris en compte, y compris l'hétérogénéité des utilisateurs et la dynamique des interactions entre les serveurs et les clients [DON, 08].

## CONCLUSION

En conclusion nous pouvons dire que la vidéo sur internet attend toujours des améliorations, certaines ont été acquises surtout ces dernières années. Des technologies spécifiques ont été développées et aujourd'hui ce marché est en pleine explosion même si de nombreux progrès restent encore à faire.

Dans ce chapitre, nous avons présenté la vidéo et son trafic sur internet.Ensuite,nous avons clarifié la différence entre un système, application et service vidéo. Nous avons détaillé le streaming vidéo, son principe de fonctionnement et ses approches : statique, dynamique et adaptatif. Nous avons expliqué les étapes de streaming avec des exemples pratiques. Nous avons cité les architectures de streaming et nous nous sommes basé sur les protocoles spécifiques au streaming vidéo.

Nous avons vu plusieurs architectures pour le streaming vidéo mais dans notre travail nous nous intéressons à l'architecture CDN. C'est l'architecture la plus utilisée pour le streaming. Le prochain chapitre sera consacré à ce type de réseau CDN.

# CHAPITRE II

## Les Réseaux de Distribution de Contenu

# INTRODUCTION

Internet a évolué d'une façon remarquable les dernières années grâce au succès commercial de l'internet et ses services ce qui a conduit à l'évolution de trafic réseau surtout après l'explosion du contenu media fourni en ligne. Cette évolution a obligé les fournisseurs de services à satisfaire leurs clients (utilisateurs finaux) s'ils veulent rester en course. La plus part de ces services notamment les services vidéo ont souffert de saturation et de blocage à cause de grand nombre de demandes envoyées vers leurs services. Par ailleurs, lorsque les utilisateurs de ces services ne sont pas satisfaits de la qualité offerte, ils peuvent renoncer à l'utilisation du service et passer à une autre offre concurrente. Au fait, lors de l'utilisation des services vidéo, les utilisateurs finaux s'intéressent plus particulièrement à la disponibilité, au temps de réponse et à la qualité de vidéo offerte. Ces besoins de services vidéo ont pavé le chemin pour la naissance de Réseaux de Distribution de Contenu (Content Delivery Networks, CDN).

Dans ce chapitre nous allons essayer d'expliquer le fonctionnement de Réseaux de Distribution de Contenu (CDN).Nous commençons par un aperçu sur les CDN, Nous nous intéressons ensuite au fonctionnement de leur Système de Routage des Requêtes (SRR) dans la section II. La section III sera consacrée au mécanisme de redirection de requêtes. Nous examinons enfin un exemple du SRR réellement déployé par l'entreprise Akamai, leader dans le marché des CDNs.

**Nous tenons à préciser qu'une grande partie de ce chapitre est reprise à partir du chapitre 3 du mémoire [ARO, 12].**

## DEFINITION

Un réseau de distribution de contenu (Content Delivery Network, CDN) est une collection collaborative des éléments de réseau couvrant Internet, où le contenu est répliqué sur plusieurs serveurs Web en miroir afin d'effectuer la livraison transparente et efficace du contenu des utilisateurs finaux. La collaboration entre les composants distribués CDN peut se produire sur des nœuds dans les deux environnements homogènes et hétérogènes [PBV, 08]. Les CDN ont évolué pour surmonter les limitations inhérentes de l'Internet en termes de qualité de service perçue par l'utilisateur lors del'accès au contenu Web. Ils fournissent des services qui améliorent les performances du réseau en optimisant la bande passante, l'amélioration de l'accessibilité, et le maintien de justesse grâce à la réplication de contenu.

Le contenu d'un serveur d'origine est répliqué (ou reproduit) sur plusieurs serveurs, appelés serveurs réplicas, qui sont répartis dans le monde entier à proximité des utilisateurs terminaux. Le contenu peut se référer à toutes les ressources de données numériques. Il se compose de deux parties principales : les données eux-mêmes (documents, page web, images, audio et vidéo) et les données qui les décrivent (les métadonnées) [PBV, 08].

La figure II.1 représente le modèle d'un CDN où les ensemblesde serveurs Web répliqués travers le monde entier sont situées à la périphérie du réseau auquel les utilisateurs finaux sont connectés. Un CDN distribue le contenu à un ensemble de serveurs Web, dispersés dans le monde entier, pour fournir le contenu aux utilisateurs finaux de manière fiable et en temps opportun. Le contenu est répliqué soit à la demande des utilisateurs, ou elle peut être reproduite à l'avance, en mettantle contenu sur les serveurs Web distribués. Un utilisateur est servi avec le contenu du plus proche serveur Web répliqué. Ainsi, l'utilisateur se retrouve sans le savoir communiquer avec un serveur CDN répliqué près de lui et récupère les fichiers de ce serveur.

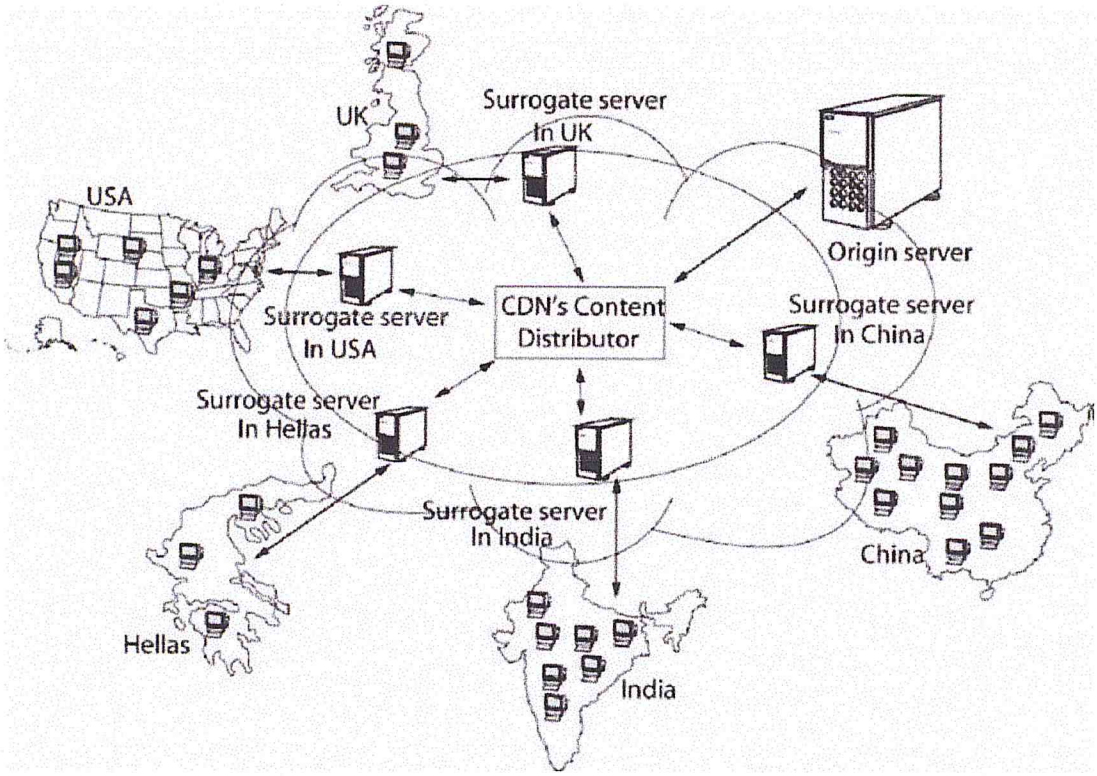



Figure II.1 Modèle d'un réseau de distribution de contenu (CDN) [PBV, 08]

Les trois principaux acteurs dans un CDN sont les suivants : les fournisseurs de contenu, le fournisseur CDN, et les utilisateurs finaux. Un fournisseur de contenu est celui qui délègue l'Uniform Resource Locator (URL) des objets Web à être distribués. Le serveur d'origine de fournisseur de contenu occupe ces objets. Un fournisseur CDN est un organisme propriétaire ou une entreprise qui fournit des installations d'infrastructure aux fournisseurs de contenu afin de fournir du contenu d'une manière rapide et fiable. Les utilisateurs ou les clients finaux sont les entités qui accèdent au contenu du site Web du fournisseur de contenu [PBV, 08].

## L'ARCHITECTURE DES CDNs

L'architecture générale d'un CDN se compose de six éléments ou composants de base : les clients, les serveurs réplica, le serveur d'origine, le Système de Distribution du Contenu (SDC), le Système de Routage des Requêtes (SRR) et le Système de Comptabilité (SC). La figure II.2 présente un schéma très simplifié de cette architecture où les relations entre ses composants peuvent être résumées comme suit [GCT, 01] [KMS, 02] [MRP, 04][ARO, 12] :

- Le serveur d'origine publie le contenu qui doit être distribué et livré par le SDC dans le CDN et délègue l'URL (Uniform Resource Locator) de ce contenu au SRR.
- Le Système de Distribution du Contenu (SDC) réplique le contenu du serveur d'origine vers un ou plusieurs serveurs réplica. Il interagit avec le SRR pour

l'informer de la disponibilité du contenu dans les différents serveurs réplica. De plus, il interagit avec le SC pour l'informer de l'activité de distribution du contenu pour que ce dernier puisse mesurer le volume de distribution de contenu.

- Le client effectue une demande de contenu qu'il perçoit comme étant dans le serveur d'origine. Cette demande (ou requête) est interceptée par le SRR.
- Le Système de Routage des Requêtes (SRR) redirige la requête du client au serveur réplica approprié dans le CDN. Il interagit ainsi avec le SC pour l'informer de la livraison du contenu aux clients. Il arrive aussi qu'il interagisse avec le SDC pour l'informer de la demande du contenu afin de placer le contenu dans les serveurs réplica.
- Le serveur réplica sélectionné fournit alors le contenu demandé au client. Il informe par ailleurs le SC par retour d'information (feedback) du contenu transmis.
- Le Système de Comptabilité (SC) collecte les données à partir du SRR, du SDC et des serveurs réplica pour les formuler ensuite dans des rapports statistiques. Celles-ci sont utilisées comme une base d'informations pour la facturation des biens et d'obligations entre les opérateurs de CDN et les fournisseurs de contenus. Elles servent aussi comme un retour d'informations pour le SRR.

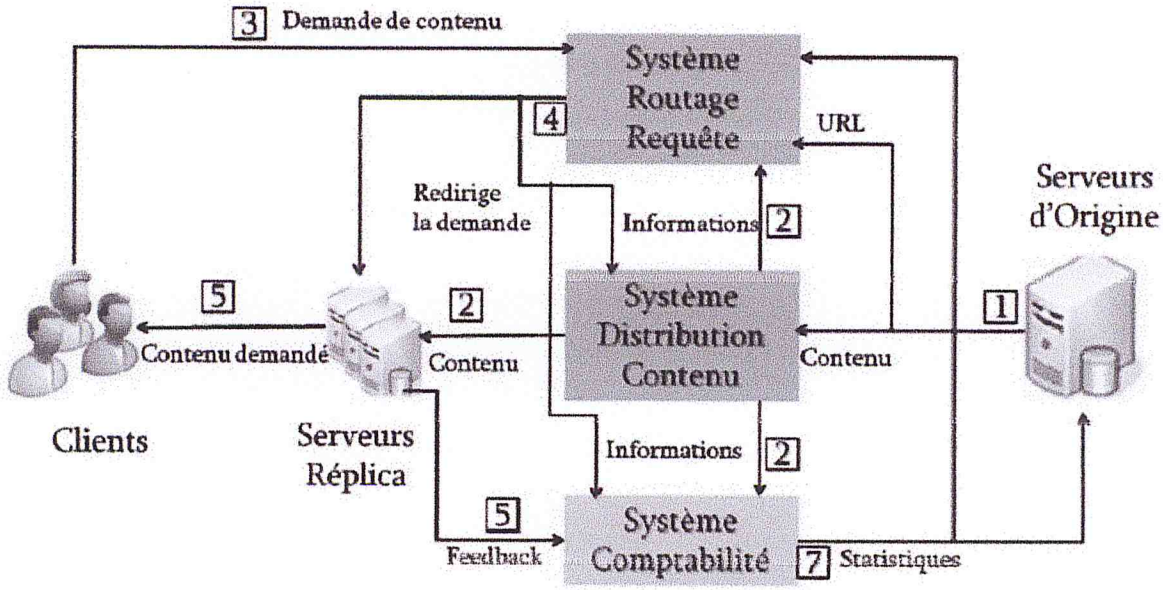



Figure II.2. L'architecture de CDN[ARO, 12]

## CONCEPTION D'UN CDN ET SES PROBLEMES

Selon [BCT, 04], le fonctionnement des CDNs implique la gestion et le contrôle de plusieurs fonction au niveau son architecture. Il exige une gestion des serveurs réplicas, de la distribution de contenu et de la redirection des clients.

Dans la littérature, plusieurs travaux de recherche ont été proposés dans le but de résoudre ces problèmes. Nous allons décrire ces problèmes en citant les principaux travaux dans le domaine.

## III.1 Gestion et placement des serveurs réplicas

Un topique intéressant dans les CDN qui est le placement réplica.Il s'agit d'où et comment les serveurs réplicas doivent être distribué à travers l'internet pour minimiser la latence, le nombre des serveurs et la consommation de la bande passante utilisée pour la gestion des réplicas [VAP, 06].

Plusieurs algorithmes ont été proposés pour résoudre ce problème, comme l'algorithme à base d'arbre [LGI, 99], l'algorithme de Glouton [RGE, 01] qui positionne progressivement des serveurs réplica et l'algorithme Hot Spot [QPV, 01] qui met les serveurs réplica à proximité des clients générant plus de charge. Ces algorithmes spécifient l'emplacement des serveurs réplica afin de réaliser de meilleures performances avec des coûts d'infrastructure faible. L'expérimentation a montré que l'algorithme de Glouton donne les meilleures performances [QPV, 01].

La majorité des schémas présentés dans la littérature attaquent le problème des placements réplicas statiques qui peuvent être formulés comme suit : Etant donné une topologie de réseau donnée, un ensemble de serveurs CDN et un paquet de trafic de requête donné, le but est de décider où répliquer le contenu afin d'optimiser une fonction objectif avec des contraintes de ressources. Les solutions proposées ont essayé de maximiser la qualité de réception du client dans le cas d'une infrastructure donnée, ou de minimiser les coûts de l'infrastructure CDN à la rencontre d'un client spécifié. Par exemple, les contraintes prise en compte sont limitées aux capacités de stockage des serveurs, aux taux de chargement des serveurs, au délai maximum toléré par les clients etc.

Une brève étude des différentes fonctions et des contraintes objectives considérées dans la littérature peut être trouvées dans [KKM, 02]. Pour le cas statique, quelques solutions ont été proposées dans [QPV, 01], [JJK, 01] et [RGE, 01]. Dans [QPV, 01] Qiu et al. Ont formulé le problème des placements réplicas statiques au problème K minimum, tels que les K placements des serveurs réplicas ont été sélectionnés pour que la somme des distances entre les clients et leurs meilleurs serveurs réplicas soit minimisée. Dans [JJK, 01] et [RGE, 01] Jamin et al. Et Radoslavov et al. proposent des heuristiques tels que les réplicas sont placés dans les nœuds avec la plus grande fan-out indépendamment de la fonction du coût réel. La raison est que les nœuds sont susceptibles d'être dans des endroits stratégiques, les plus proches (en moyenne) à tous les autres nœuds, et par conséquent adapté pour un placement réplica. Dans [RGE, 01] une évaluation de performance basée sur une topologie au niveau routeur sur le monde réel montre que l'heuristique basée sur la fan-out a un comportement

proche de l'heuristique Glouton en terme de latence moyenne du client. Dans les deux [CKS, 01] et [LGI, 99] les auteurs considèrent le problème du placement réplicas d'un serveur d'origine sur une topologie en arbre. Dans [KRS, 00] les auteurs aussi considèrent également une topologie très simple comme des anneaux, des lignes et de arbres tout en considérant la mise en place des proxys interceptant à l'intérieur du réseau pour réduire le temps de téléchargement. Dans [KRR, 02] le problème de réplication optimale des objets dans les serveurs CDN est analysé, toutes ces solutions n'ont pas à considérer la dynamique du système (par exemple, des changements dans la structure de la circulation des requêtes, la topologie du réseau, les sites réplicas).

Dans [BLP, 03] une approche différente est proposée et une stratégie d'allocation dynamique est considérée qui prend explicitement en compte la dynamique du système ainsi que les coûts de la modification des placements réplicas. En assumant la dynamique des requêtes des utilisateurs pour obéir au modèle Markovian, une formulation du problème des placements réplicas dynamiques est obtenue. Mais ce modèle ne peut pas saisir avec précision la dynamique des utilisateurs et peut être résolu numériquement que pour les CDNs de taille limitée, il nous permet d'identifier une politique optimale pour le placement réplica dynamique qui peut être utilisé comme un point de référence pour l'évaluation des heuristiques et donne un aperçu sur la formulation d'une heuristique de placement conservatrice.

La solution de placement réplicas de contenu lourds, comme le streaming vidéo, rend impossible le stockage de la totalité du contenu de plusieurs flux longs parce qu'il épuiserait la capacité du cache. Dans ce cas, non seulement le contenu doit être reproduit et distribué aux répliques, mais aussi d'une manière qui évite de surcharger les serveurs. Pour résoudre ce problème, dans [SRT, 99] et [WSA, 02], les auteurs proposent une technique de mise en cache préfixe lequel un proxy stocke les images initiales de clips populaires. À la réception de la demande pour le flux, le proxy initie la transmission de la demande client en même temps et les trames restantes à partir du serveur. En plus de cacher les retards, le débit et la perte des effets d'un modèle de service plus faible entre le serveur et le proxy, cette technique de mise en cache facilite les mandataires à effectuer le travail avant le lissage dans le tampon de lecture du client, en transmettant les grandes trames à l'avance de chaque salve. Les techniques de mise en cache préfixe réduit le pic et la variabilité des besoins en ressources de réseau le long du chemin de la procuration pour le client.

## III.2 Techniques de Distribution de Contenu

Les principaux problèmes liés à la distribution du contenu se résument dans les problèmes de placement et de délivrance du contenu. Le problème de placement consiste à choisir le serveur réplica dans lequel le contenu sera stocké de façon à minimiser la charge du serveur réplica et/ou à équilibrer la charge entre l'ensemble des serveurs réplica. En

- **La distribution statique :** mettre en cache et de reproduire le contenu statique, tel que des pages html, images, documents, fichiers audio / vidéo, etc
- **La distribution dynamique :** mettre en cache et répliquer le contenu généré dynamiquement. Cela inclut la livraison d'applications et de réplication.
- **La distribution de streaming media :** pour stocker des objets de streaming media, ainsi que de servir streaming aux clients. Essentiellement, le cache agit comme un serveur multimédia en continu, stocker des clips multimédias pour une utilisation ultérieure.
- **Fractionnement en direct :** le cache répliqué flux en direct, de sorte que seul un exemplaire est tiré vers le bas à partir du serveur en amont et est ensuite distribué aux clients abonnés.

Nous tenons aussi à préciser que le contenu peut être entièrement ou partiellement reproduit dans les serveurs réplica. En effet, il est préférable parfois de partitionner un contenu de grand volume, comme les fichiers multimédia, sur plusieurs serveurs réplica qui sont limités par leurs capacités de stockage et de traitement. Cependant, la gestion de cette dernière approche (connue sous le terme anglais Partial-Site) est plus complexe que la gestion de l'approche traditionnelle (Full-Site) où le contenu en entier est répliqué dans les serveurs réplica [PAB, 08].

## III.3 Le Système de Routage de Requêtes

Lors du routage des requêtes dans le CDN, il est nécessaire de sélectionner d'abord le meilleur serveur réplica parmi l'ensemble des serveurs réplica disponibles, de rediriger ensuite cette requête vers ce meilleur serveur. La sélection prend en compte différents critères : la distance entre le client et le serveur réplica (nombre de saut, la latence, ...), la charge des serveurs réplica disponibles, etc. Une fois le meilleur serveur choisi, la requête sera redirigée vers lui par un mécanisme approprié de routage : redirection HTTP, redirection DNS, Anycast, .... Dans notre travail, nous nous intéressons au fonctionnement de e SRR. C'est pourquoi, nous allons détailler ces mécanismes ainsi que les algorithmes de sélection existants dans les sections suivantes.

Il est à préciser ici que la conception du système de distribution du contenu (SDC) a un impact direct sur la conception du système de routage des requêtes (SRR). En effet, si le contenu en entier est répliqué sur les serveurs réplica (l'approche Full-Site), le SRR dirige directement les requêtes des clients vers le meilleur serveur réplica. Dans le cas contraire (l'approche Partial-Site), le SRR est conçu de telle façon qu'à la réception de la requête du client, il reçoit, de la part du SDC, la liste contenant toutes les parties du contenu et leurs emplacements dans les serveurs réplica. Le SRR redirige la requête du client vers plusieurs meilleurs serveurs réplica ayant chacun une partie du contenu [PAB, 08]. En outre, dans le mode Pull, le SRR interagit également avec le SDC pour l'informer du résultat de sa sélection afin de placer ou mettre à jour le contenu dans le serveur réplica sélectionné [KMS, 02].

## LES RESEAUX DE DISTRIBUTION DE CONTENU EN STREAMING

Les CDNs ont été originalement conçus pour distribuer des contenus très demandés à partir des serveurs web les plus populaires. Aujourd'hui, la plupart des CDNs supportent le streaming média, i.e. la livraison et la lecture simultanées de contenus média (vidéo/audio). Ces CDNs sont connus sous le nom de SCDNs (Streaming CDNs) pour les réseaux de distribution de contenu en streaming. Certains SCDNs sont explicitement conçus pour offrir le média en streaming tels que PRISM [CGK, 01], VCDN [CAS, 03], MarconiNet [DSY, 99] ou EdgeStream8. D'autres SCDNs à usage général supportent tout type de contenu, y compris le contenu média comme Akamai9, Limelight Networks ou MirrorImage. Comparativement aux CDNs traditionnels (supportant seulement la livraison des pages web), les SCDNs imposent plus de contraintes [ATW, 02] :

- Une page web est un élément relativement léger, de l'ordre de 10 kilooctets. Elle peut donc être répliquée dans son intégralité sur chaque serveur choisi. Par contre, les contenus de type média (comme les films) ont une longue durée et nécessitent une quantité importante de stockage, de l'ordre de quelques mégaoctets à quelques giga-octets. Il n'est donc pas pratique ou souhaitable de répliquer un flux média en entier sur chaque serveur choisi. Le contenu média est alors partitionné en plusieurs parties ou segments, et seulement une portion du média est mise sur un serveur.
- Le transfert des flux streaming sur le réseau Internet imposent des contraintes très fortes en termes de QoS. Par exemple, si la congestion et la perte de paquets conduisent à un délai de quelques secondes dans la livraison d'une page web (ce qui est souvent acceptable), leur effet par contre sur une session de streaming des médias peut poser problème (interruptions et plusieurs artefacts).
- L'interaction client-serveur pour un SCDN implique une session RTP/UDP très longue à l'opposé du CDN conventionnel qui implique une (ou plusieurs) session (s) HTTP / TCP plus courte (s) (de l'ordre d'une fraction de seconde). Il est donc parfois nécessaire d'effectuer une opération de transfert intercellulaire en plein milieu d'une session RTP (midstream handoff). Cela consiste à solliciter un autre serveur sans interruption de service afin d'équilibrer la charge entre les serveurs streaming.

## LA SELECTION DU MEILLEUR SERVEUR REPLICA

Les algorithmes de sélection du serveur réplica permettent de sélectionner le serveur vers lequel un client donné doit être redirigé [SPV, 03]. La sélection peut être aléatoire [DEL, 99] ou faite en fonction de plusieurs métriques liées à la performance du CDN. Ces métriques sont détaillées dans la section IV. Mais avant cela, nous présentons les techniques de mesures utilisées.

## V.1 Techniques de mesures

Les systèmes de routage de requêtes peuvent utiliser une variété de mesures afin de déterminer le meilleur serveur réplica qui peut servir la requête du client. La nature décentralisée de l'internet rend l'évaluation quantitative de la performance du réseau très difficile. La collecte des statistiques de réseau directement sur les périphériques réseau (routeur et serveur) pourrait être plus couteuse en termes de performance du système. Donc, en général, l'acquisition de données statistiques du réseau repose sur l'utilisation d'une combinaison de méthodes de sondage sur le réseau actif, la surveillance du trafic passif et les commentaires des serveurs réplicas. Pour des détails plus profonds de la façon dont les mesures peuvent être déduites voir aussi [BDL, 02] [CHN, 02].

Dans les CDNs, il est possible de combiner plusieurs mesures (paramètres) en utilisant à la fois le concept de proximité et de rétroaction du serveur réplica pour une meilleure sélection. La mesure de performance est souvent une composante des systèmes de gestion de réseau et offre la possibilité de suivre, comprendre et projeter la performance de bout en bout de CDN. De plus, il faudrait mesurer à la fois la performance interne ainsi que la performance du point de vue du client. Les paramètres typiques qui pourraient être utiles pour mesurer sont : la perte de paquets et la latence pour tout type de contenu et en particulier pour la diffusion de contenu de la bande passante moyenne, le temps de démarrage et la cadence. En déployant des sondes à base de matériel ou de logiciel, de manière stratégique dans le réseau, on peut corréler les informations recueillies par les sondes avec le cache et les journaux du serveur pour déterminer les statistiques de livraison et de qualité de service. L'endroit le plus utile pour mettre les sondes est le bout du réseau, en mesurant la performance perçue par les utilisateurs finaux à travers le CDN.

Les mesures de réseau et proximité géographique peuvent être utilisées par le système de routage de requêtes SRR pour diriger les utilisateurs au serveur réplica le plus proche. En outre, des mesures de proximité peuvent être échangées entre les serveurs réplicas et l'entité requérante. Dans nombreux cas, les mesures de proximité sont « à sens unique » en ce qu'ils permettent de mesurer le chemin des paquets vers l'avant ou vers l'arrière depuis les serveurs réplicas jusqu'au système de routage. Ceci est important car de nombreux chemins dans l'internet sont asymétriques. Afin d'obtenir un ensemble de mesures de proximité, un réseau peut employer des techniques de sondage actives et / ou techniques de mesures passives. Le système de routage de requêtes peut aussi utiliser les feedback des serveurs réplicas afin de sélectionner un nœud de livraison moins chargé. Les feedback peuvent être livrés à partir de chaque serveur réplica ou peuvent être agrégées par site ou par emplacement. Nous allons discuter en détails sur les mesures passives, sondage actif et la rétroaction.

### V.1.1Mesure passive sur le réseau

Appelée aussi la surveillance passive du trafic [méta-routage], c'est une technique de mesure dans laquelle le trafic entre le client et le serveur réplica est surveillé en continu pour avoir en retour des mesures de performances réelles [BCT, 04]. Ces mesures peuvent être obtenues lorsqu'un client effectue un transfert de données vers ou à partir d'un serveur réplica. Une fois le client se connecte, la performance réelle du transfert est mesurée (bande passante, délai, taux de perte, ...). Ces données sont ensuite réinjectées dans le système de routage de requêtes. Un exemple de mesure passive est d'observer la perte de paquets à partir du client vers un serveur réplica, ou la latence du client perçue par l'observation du comportement de TCP. On peut l'application de cette technique dans le mode Server Push [FBZ, 98] où le serveur réplica est modifié de telle sorte qu'il permette d'envoyer les performances du transfert seulement lorsque des changements intéressants sont observés. A la base, un bon mécanisme est nécessaire pour assurer que chaque serveur réplica est testé par un client afin d'obtenir les données. Dans [SKS, 01], les auteurs ont proposé un système basé sur des mesures passives de la performance du réseau à utiliser avec des applications adaptatives. Les principaux avantages de cette technique sont l'évolutivité et la précision des mesures du serveur, tandis que, la modification des serveurs et l'absence de mesure de la route en constituent les principaux inconvénients.

### V.1.2Sondage active du réseau

Appelée aussi « enquête active du réseau » [BCT, 04] qui est une technique de mesure où des enquêtes ou des interrogations (Probing) sont envoyées périodiquement aux ou à partir des serveurs réplicas afin de relever leurs performances. Les entités demandées passées ou possibles sont sondées en utilisant une ou plusieurs techniques pour déterminer une ou plusieurs mesures pour chaque ou plusieurs serveurs réplicas. Un exemple typique de cette technique de sondage est la requête d'écho ICMP envoyée périodiquement de chaque serveur réplica ou plusieurs serveurs vers le client désirant le contenu dans l'objectif de mesurer une performance liée au chemin suivi par les flots au travers du réseau. Cette technique est limitée pour certaines raisons. Les mesures ne peuvent être prises que périodiquement et ne devraient pas avoir une charge perceptible car elle ne peut pas influencer les trafics mesurés, les pare-feu et NATs interdisent les sondages, et en dernier, les sondes sont souvent la cause des alarmes de sécurité déclenchées sur les systèmes de détection d'intrusion.

L'avantage principal de cette technique est celle de ne nécessiter aucune modification coté serveur. Cependant, elle introduit une latence supplémentaire qui pourra être importante si les enquêtes sont effectuées fréquemment. De plus, enquêter peut parfois conduire à une métrique inexacte comme par exemple celle inhérente au trafic ICMP qui

peut être ignoré ou serait moins prioritaire en raison des choix du client, d'où un temps erroné.

### V.1.3Rétroaction des serveurs réplicas

La rétroaction « Feedback » des serveurs réplicas [BCT, 04] peut être obtenue en effectuant des sondages « enquêtes » active périodiquement sur les serveurs réplicas en émettant des requêtes spécifiques d'applications (exemple : http) et en prenant des mesures connexe. Les problèmes de sondage pour les informations des serveurs réplicas est qu'il est difficile d'obtenir des informations en temps réel et les informations en temps non réel sont parfois inexactes et obsolètes. Donc, les informations de rétroaction peuvent être obtenues par des agents qui se trouvent dans les serveurs réplicas et qui peuvent communiquer une variété de métriques concernant la performance du réseau et des serveurs. Il existe deux méthodes pour obtenir des informations de rétroaction : statique, dans lequel la route, qui minimise le nombre de sauts ou d'optimiser d'autres paramètres statiques, est sélectionnée [DRJ, 00] [SBS, 98] ; sondage dynamique (en temps réel) permet de calculer le temps d'aller-retour ou d'autres paramètres de qualité de service en « temps réel » [DRJ, 00] [CCD, 95]. Les méthodes hybrides sont également utilisées pour obtenir d'autres informations de rétroaction utiles [DRJ, 00] [EIC, 00] [MTT, 01]. La rétroaction a été souvent utilisée pour surveiller la charge des serveurs réplica [BCN, 03].

## V.2 Métriques pour la redirection des requêtes

La sélection du serveur réplica est effectuée dans le but de minimiser certains paramètres de performance perçus par les utilisateurs finaux. Dans cette section nous donnons une classification des mesures qui peuvent être adoptées pour mesurer la performance du réseau et des systèmes dans un CDN pour décider où rediriger la requête du client.

- *La proximité géographique* est souvent utilisée pour rediriger tous les utilisateurs dans une certaine région de la même POP [BCT, 04].
- La mesure de *la proximité du réseau* est typiquement dérivée du sondage actif par l'intermédiaire de la table de routage BGP [BCT, 04].
- La possibilité de choisir le POP qui montre la latence la plus faible peut être obtenue en améliorant les systèmes de redirection de requêtes avec des mécanismes de sondage actif et mesure passive, afin de maintenir la connaissance du *temps de réponse.*
- *L'état de la charge du serveur* peut être calculé, en utilisant SNMP ou les agents de rétroaction sur le côté serveur, sur la base de l'état de charge des composants du serveur (CPU, disque, mémoire, interfaces réseau) ou sur la base d'une mesure de la performance globale, comme le débit du serveur ou du temps de réponse du serveur.

- Toutes ces mesures peuvent être des informations pertinentes pour alimenter le mécanisme de sélection de serveur, associées à la connaissance de *l'identité de l'utilisateur*, qui est destinée à classer la priorité de l'utilisateur à accéder à des contenus et services. A titre d'exemple, un client payeur peut avoir accès à de meilleurs services que les non-payeurs. L'identité des utilisateurs payeurs peut être révélée par l'intermédiaire d'un cookie récupéré à partir du système client, ou à un processus d'authentification.

Le tableau résume la classification des indicateurs de performance ci-dessus.

<table>
<tr><th>Métriques</th><th>But</th><th>Technique de mesure</th></tr>
<tr><td>Latence</td><td>Choisir un réplica avec un délai le plus petit</td><td>Sondage actif / mesure passive</td></tr>
<tr><td>Perte de paquets</td><td>Sélectionner le chemin avec un taux d'erreur le plus faible (Utile pour le trafic streaming)</td><td>Sondage actif / mesure passive (information d'en-tête TCP)</td></tr>
<tr><td>Proximité réseau</td><td>Sélectionner le chemin le plus court</td><td rowspan="4">Sondage actif</td></tr>
<tr><td>Bande passante moyenne</td><td rowspan="3">Sélectionner le meilleur chemin pour le trafic streaming</td></tr>
<tr><td>Temps de démarrage</td></tr>
<tr><td>Taux d'armature</td></tr>
<tr><td>Proximité géographique</td><td>Rediriger les requêtes provenant d'une région vers la même POP</td><td>Informations d'en-tête IP, informations bind</td></tr>
<tr><td>Charge de CPU, réseau, charge d'interface, connexion active, charge de stockage</td><td>Sélectionner le serveur avec moins charges</td><td>Agents de rétroaction / Sondage actif</td></tr>
</table>

Table II.1. Tableau Métriques utilisées dans la sélection du serveur réplica [BCT, 04]

## V.3 Classification des algorithmes de sélection

Les algorithmes de sélection peuvent être adaptatifs ou non adaptatifs [PAB, 08]. Ceux de la première catégorie considèrent l'état actuel du CDN pour sélectionner un serveur réplica. L'état actuel du CDN est obtenu par l'estimation de certaines mesures comme la charge des serveurs réplica ou la congestion des liaisons du réseau sélectionné. Par ailleurs, les algorithmes adaptatifs ont la capacité de changer leurs comportements pour faire face à une situation durable, ce qui les rendent plus complexes que les algorithmes non-adaptatifs. Ces derniers utilisent des heuristiques pour sélectionner un serveur réplica plutôt que de

considérer l'état du système actuel. Ils sont faciles à mettre en œuvre et fonctionnent efficacement lorsque les hypothèses faites par les heuristiques sont remplies.

Au regard de sa simplicité, l'algorithme non-adaptatif le plus connu est celui fonctionnant avec la technique du round-robin. Cet algorithme distribue toutes les requêtes des clients vers les serveurs réplica du CDN et tente d'équilibrer la charge entre eux [SZY, 02]. Il suppose que tous les serveurs réplica ont la même capacité de traitement et que l'un d'eux peut servir tout type de demande du client. Cet algorithme est efficace pour les clusters, où tous les serveurs réplica sont situés au même endroit [PAB, 98]. Cependant, il est inefficace dans le cas où les serveurs réplica sont situés à des endroits éloignés. En effet, il ne considère pas la distance entre les serveurs réplica et les clients et il peut ainsi rediriger un client vers un serveur réplica plus éloigné entraînant ainsi des performances médiocres perçues par les clients terminaux. De plus, l'équilibrage de charge entre les serveurs réplica n'est pas pleinement atteint, le traitement des différentes demandes peut en effet impliquer significativement différents coûts de calcul [PAB, 08].

DistribuedDirector [DEL, 99] est une solution propriétaire Cisco qui met en œuvre différents types d'algorithmes de sélection adaptatifs et non adaptatifs. Parmi les algorithmes non-adaptatifs, on trouve celui qui considère le pourcentage des requêtes des clients que chaque serveur réplica reçoit. Un serveur recevant davantage de requêtes est supposé être plus puissant. Par conséquent, les demandes des clients sont dirigées vers les serveurs les plus puissants pour atteindre une meilleure utilisation des ressources. Cependant, les serveurs les plus puissants ne sont pas forcément situés dans des endroits proches des utilisateurs. Un autre algorithme non-adaptatif considère la proximité géographique du client pour rediriger les requêtes vers le serveur réplica le plus proche. Cet algorithme souffre du fait que les demandes des clients peuvent être affectées à des serveurs réplica surchargés pouvant ainsi dégrader les performances perçues par le client. La technique DistributedDirector utilise aussi un algorithme de sélection adaptatif qui prend en compte une combinaison pondérée de trois indicateurs, à savoir la distance inter-AS (le nombre des systèmes autonomes (Autonomous System, AS) traversés), la distance intra-AS (le nombre de sauts traversés), et la latence de bout en bout. Bien que cet algorithme soit souple du fait qu'il permet l'utilisation des trois métriques, le déploiement d'un agent dans chaque serveur réplica pour mesurer ces métriques le rend complexe et coûteux. De plus, les techniques actives utilisées par cet algorithme pour mesurer la latence introduisent un trafic supplémentaire [PAB, 08].

Dans [AFN, 01] et [ASS, 02], les auteurs ont proposé des algorithmes de sélection adaptatifs basés sur la latence client-serveur. Cette latence mesurée passivement au niveau de serveur, est utilisée pour sélectionner le serveur réplica ayant la plus petite latence récemment signalée. Toutefois, ces algorithmes exigent le maintien d'une base de données

centrale contenant les différentes mesures de la latence limitant ainsi l'évolutivité des systèmes sur lesquels ces algorithmes sont déployés [SSP, 04].

## MECANISMES DE REDIRECTION DE REQUETES

Un défi majeur dans la conception de CDN est de réaliser un service de redirection de requêtes efficace qui permet de suivre les serveurs où les données sont répliquées et attribue à chaque requête de client un serveur qui peut offrir le « meilleur » service, ce processus est appelé sélection de serveur. Les algorithmes de sélection de serveur comprennent des critères comme la topologie du réseau, la disponibilité des serveurs et la charge du serveur [SPV, 03]. La capacité de trouver rapidement des serveurs réplicas et effectuer la requête de distribution a des implications importantes pour le temps de réponse perçu par l'utilisateur. La figure II.3 illustre une vue de haut niveau du processus de redirection de requête : 1) le client demande le contenu, par exemple un fichier de streaming, qui se trouve à www.site.com ; 2) tant que site.com n'héberge pas le fichier demandé, mais utilise cdn.com comme fournisseur CDN, la requête est redirigée vers le site cdn.com ; 3-3') en utilisant un algorithme de redirection, le client est redirigé vers le serveur réplica le plus approprié. Si le client a un CDN réplica placé à son directement à son réseau de fournisseur de service internet ISP qui est capable de garantir ce streaming, ce réplica est alors sélectionné (3) en premier (CDN cache-2), d'autre part un autre (CDN cache-1) est sélectionné (3'). 4-4') le serveur réplica sélectionné fournit le contenu au client.

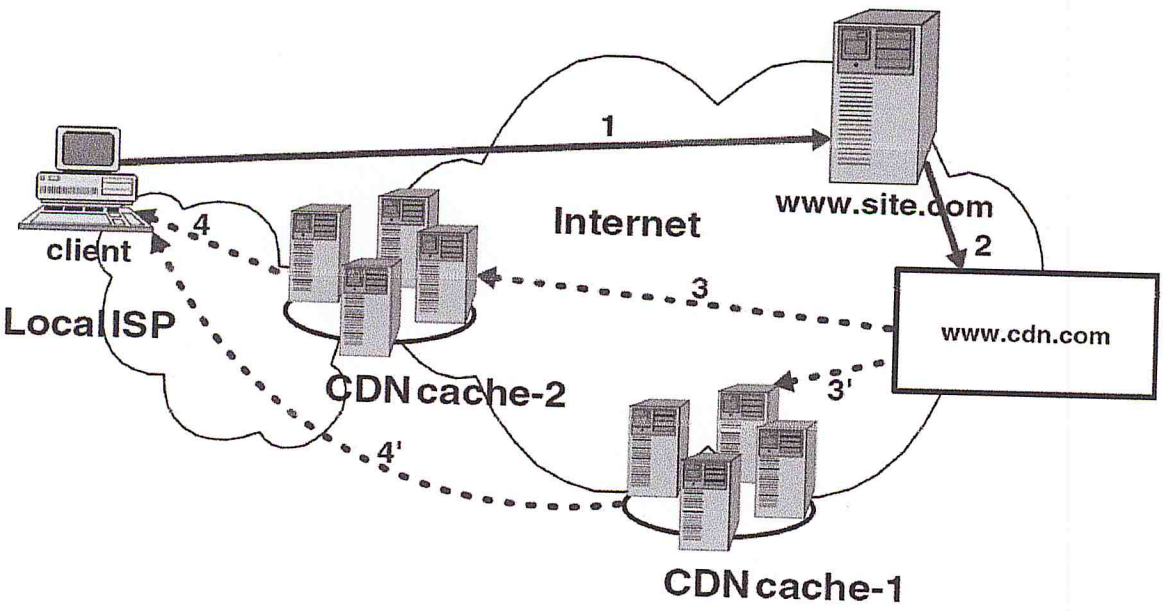



Figure II.3. Processus de redirection de requête[SPV, 03]

Les mécanismes de redirection ou de routage informent le client du résultat de la sélection en indiquant le serveur réplica choisi amené à satisfaire sa requête. Plusieurs mécanismes de redirection de requête existent. Ils peuvent être classés selon plusieurs critères. Dans ce travail, nous avons choisis volontairement de les classer selon la couche TCP\IP où ils opèrent. Cette classification est simple à utiliser et facile à développer. Comme la montre la figure II.4, nous trouvons des mécanismes qui opèrent au niveau de la couche application, de la couche transport ou encore de la couche réseau. Chacun de ces mécanismes, voire une combinaison de ceux-ci, peuvent être utilisés dans un seul CDN. Par la suite, nous décrivons les mécanismes de redirection de requêtes en présentant leurs avantages et leurs limites.

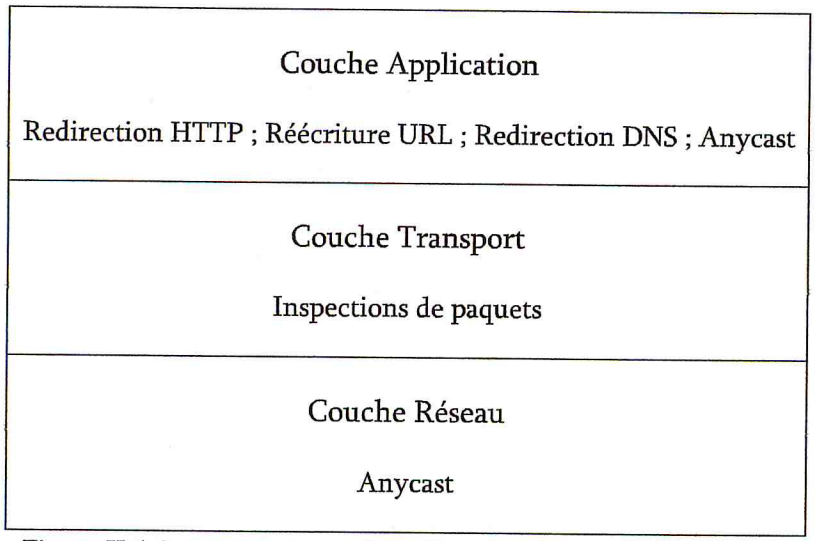



Figure II.4. Les mécanismes de redirection de requête[ARO, 12].

Plusieurs schémas ont été proposés dans la littérature de recherche pour réaliser une redirection au niveau IP par un schéma de certain paquet d'adresse [DKM, 96] [HGK, 98], au niveau DNS grâce à la cartographie des noms URL aux adresses IP de l'un des serveurs d'un cluster [CCD, 14] [AYH, 96] [PYD, 98], ou au niveau Application [WNC, 04] en utilisant les fonctions de redirection du protocole HTTP. Dans [KDK, 03] les auteurs proposent un système hybride basé sur la réplication adaptative des entrées de répertoire qui offre le service de redirection. Le support de niveau réseau permet aux requêtes de client d'être redirigées vers les serveurs appropriés d'une manière rapide et efficace. Dans [CCY, 01]les auteurs se concentrent sur une autre architecture qui intègre le mécanisme de répartition de DNS avec une technique de redirection HTTP effectuées par les serveurs web. Dans les sections suivantes nous décrivons en détails les principaux mécanismes de redirection de requêtes.

## VI.1 Niveau de la couche application

Avec le routage de requêtes au niveau de la couche application un contrôle plus fin, au niveau des différents objets qui composent le contenu multimédia, peut être réalisé. Le processus peut être effectué en temps réel, lorsque la requête d'objet atteint le serveur de contenu ou un élément de commutation. Plusieurs mécanismes de redirection de requêtes opèrent au niveau de la couche d'application. Certains utilisent un protocole spécifié comme http ou DNS et d'autres sont utilisés avec tout protocole comme la réécriture URL et la technique dite anycast.

### VI.1.1 La redirection HTTP

La redirection HTTP propage l'information sur les ensembles des serveurs réplica dans les en-têtes HTTP [PAB, 08]. Les protocoles HTTP permettent à un serveur web de répondre à une demande du client avec un message spécial qui indique au client de soumettre à nouveau sa demande à un autre serveur. La redirection HTTP peut être utilisé à la fois pour la distribution de contenus en Full-Site et en Partial-Site. Elle peut aussi être utilisée pour construire un serveur web spécial, qui accepte les demandes des clients, choisit les serveurs réplica et redirige les clients vers ces serveurs.

Le principal avantage de ce mécanisme est sa flexibilité et sa simplicité du fait qu'il utilise le protocole HTTP. En outre, la réplication peut être gérée avec une granularité fine, chaque page web peut en effet être considérée comme un granule [PEN, 08]. Néanmoins, l'inconvénient le plus important de la redirection HTTP est le manque de transparence. En effet, du fait qu'elle fait une redirection visible pour les clients, ces derniers considèrent ce procédé comme indésirable [SPV, 03]. Par ailleurs, la surcharge générée par cette approche est importante, elle introduit en effet un message aller-retour supplémentaire dans le traitement des requêtes ainsi que sur le protocole HTTP [PAB, 08].

### VI.1.2 La réécriture URL

Cette technique permet à un fournisseur de contenu pour prendre le contrôle direct sur les décisions de routage de requêtes sans avoir besoin de dispositifs spécifique de commutation ou de services d'annuaire dans le chemin entre le client et le serveur d'origine [BCT, 04]. A la base, un fournisseur de contenu peut directement communiquer au client le meilleur serveur réplica qui peut servir la requête. Les décisions concernant le meilleur serveur réplica peuvent être faites sur une base d'objets ou elles peuvent compter sur un ensemble de mesures. En général, la méthode prend avantage des objets contenus qui consistent sur une structure basique qui comprenne des références à des objets supplémentaires et intégrés. Par exemple, la plupart des pages Web, se composent d'un document HTML qui contient du texte brut avec quelques objets incorporés (par exemple

des images GIF, JPEG ou PDF) référencés à l'aide de directives HTML. En général, les objets intégrés sont récupérés à partir du serveur d'origine. Pour automatiser ce processus, des scripts spéciaux analysent de manière transparente le contenu des pages web et remplacent les URL incorporées [KWZ, 01].Un fournisseur de contenu peut maintenant modifier les références aux objets incorporés, tels qu'ils pourraient être récupérées à partir du meilleur serveur réplica.

La réécriture URL, appelée aussi la modification du contenu, est principalement utilisée dans le cas où le contenu est partiellement répliqué sur les serveurs réplica (approche dite Partial-Site) [BCN, 03]. Ainsi, le serveur d'origine redirige les clients vers différents serveurs réplica par la réécriture des liens URL dans les pages générées dynamiquement La réécriture URL peut être proactive ou réactive [PAB, 08]. Dans la réécriture URL proactive, les URL pour les objets incorporés de la page HTML principale sont formulées avant que le contenu ne soit chargé dans le serveur d'origine. L'approche réactive consiste à réécrire l'URL des objets incorporés lorsque la demande du client atteint le serveur d'origine.

- **Réécriture URL Proactive :** un fournisseur de contenu formule les URL incorporés d'une page html principale avant que le contenu est chargé sur le serveur d'origine. Dans ce cas, la réécriture d'URL peut être faite soit manuellement, soit en utilisant des outils logiciels qui analysent le contenu et remplacent URL incorporées. Puisque ce système consiste à réécrire les URL de façon proactive, il ne peut pas prendre en considération des informations spécifiques du client tout en effectuant le routage des demandes. Cependant, il peut être utilisé en combinaison avec le routage de requêtes DNS pour diriger les DNS connexes dans l'espace de noms de domaine du fournisseur de services. Le routage de requêtes dynamique basé sur les détails du client est effectué en utilisant l'approche de DNS.
- **Réécriture URL réactive :** Ce système dynamique consiste à réécrire les URL incorporés d'une page html lorsque la requête du client atteint le serveur d'origine. En dépit de l'ancien régime, celui-ci a la possibilité de considérer l'identité du client lors de la réécriture des URL intégrés. En particulier, un processus automatisé peut déterminer, à la demande, quel serveur pourrait mieux servir le client demandeur. Les URL incorporés peuvent alors être réécrits pour rediriger le client pour récupérer les objets du serveur réplica qui peut mieux répondre à la demande du client que de l'autre, avec la considération de l'emplacement et de la priorité spécifique du client considéré.

Comme les URL réécrites contiennent des noms des meilleurs serveurs réplica, le principal avantage de cette technique réside dans le fait que les clients sont liés à plusieurs serveurs réplica et non pas un seul serveur réplica comme dans les autres mécanismes de redirections. De plus, le niveau de granularité peut être atteint étant donné que les objets incorporés peuvent être considérés comme des granules. Cependant, la réécriture URL peut

surcharger le serveur d'origine et ainsi causer du retard lors de redirection des requêtes [PAB, 08].

### VI.1.3 La redirection DNS

La redirection DNS consiste à utiliser un serveur DNS local personnalisé (modifié) pour répondre aux requêtes de résolution de noms avec l'adresse IP du meilleur serveur réplica plutôt qu'avec celle du serveur d'origine [SPV, 03]. En effet, lorsque l'utilisateur envoie une requête DNS au serveur DNS local, ce dernier la transmet au SRR du CDN (qui peut être un autre serveur DNS) au lieu de déclencher la résolution DNS comme cela se fait habituellement. Ensuite, le SRR interroge chaque serveur réplica, en leur demandant d'examiner la performance de leur chemin avec le serveur DNS local. Chaque serveur réplica reporte les résultats des mesures effectuées à son niveau permettant ainsi au SRR de sélectionner le serveur réplica le plus approprié. Enfin, le SRR envoie une réponse DNS, contenant l'adresse IP du serveur réplica choisi, au serveur DNS local qui la transmet ensuite à l'utilisateur [VAP, 06]. Ce mécanisme fonctionne bien lorsque le contenu en entier est répliqué dans les serveurs réplica (approche Full-Site). En revanche, avec l'approche Partial-Site, la redirection DNS est souvent combinée avec la réécriture URL de telle manière que le serveur DNS local dirige d'abord la requête vers le serveur d'origine, ensuite, vers les meilleurs serveurs réplica contenant les objets incorporés.

Aujourd'hui les services de distribution de contenu commercial s'appuient sur les serveurs Domain Name System modifiés pour rediriger dynamiquement les clients au serveur de contenu approprié. Il s'agit de la forme la plus simple de redirection, selon laquelle un nom de domaine, par exemple, www.cdn.com a plusieurs enregistrements IP qui s'y rattachent. Quand un client demande l'adresse IP du domaine, l'une des enregistrements IP seront sélectionnés sur la base de l'action de DNS.

La popularité des techniques de routage de requête en fonction DNS est principalement due à l'omniprésence de DNS comme un service d'annuaire. Ils consistent principalement en l'insertion d'un serveur DNS spécialisée dans le processus de résolution de nom. Ce serveur spécialisé est capable de retourner un ensemble différent d'enregistrements A, NS ou CNAME basées sur les politiques définies par l'utilisateur, les paramètres, ou une combinaison des deux.

Dans l'approche de **réponse unique**, le serveur DNS renvoie l'adresse IP du meilleur réplica à un enregistrement A au serveur DNS demandé (serveur DNS du site client). L'adresse IP du serveur réplica pourrait également être une adresse IP virtuelle du meilleur ensemble de substituts pour le serveur DNS souhaité. Le meilleur substitut sera choisi, dans une seconde étape, par un mécanisme de changement de serveur.

Dans l'approche de **multiples réponses**, le serveur DNS de routage de requêtes renvoie plusieurs réponses telles que plusieurs enregistrements A pour divers substituts. L'ordre dans lequel les enregistrements sont renvoyés peut être utilisé pour diriger plusieurs clients en utilisant un serveur DNS d'un site client unique.

L'approche de la résolution **multi-niveaux** est également possible selon laquelle plusieurs serveurs de routage de requêtes DNS peuvent être impliqués dans une résolution DNS unique, exigeant ainsi des décisions complexes à partir d'un serveur unique à plusieurs serveurs, plus spécialisés, serveur de routage de requêtes DNS, disséminée dans différents points de l'Internet.

Les mécanismes les plus couramment utilisés pour insérer plusieurs serveurs de routage de requêtes DNS dans une résolution DNS unique est l'utilisation d'enregistrements NS et CNAME: enregistrements NS permettent au serveur DNS pour rediriger l'autorité de la prochaine domaine de premier niveau à un autre serveur de routage de requêtes DNS; les enregistrements CNAME permettent au serveur DNS pour rediriger la demande de résolution pour un domaine entièrement nouveau.

Les limites de base des techniques de routage de requêtes basé sur DNS peuvent être résumées comme suit ci-dessous :

- DNS permet une résolution seulement au niveau du domaine. Cependant, un système idéal demande de résolution devrait servir les requêtes à granularité objet (conservation de sessions si nécessaire).
- Une courte TTL de l'entrée DNS permet de réagir rapidement aux pannes de réseau. Ce retour peut augmenter le volume de requêtes aux serveurs DNS. Par conséquent de nombreuses implémentations DNS ne respectent pas le domaine DNS TTL.
- Routage de requête DNS ne prend pas en compte l'adresse IP des clients. Seule l'adresse Internet du serveur DNS client est connue : cela limite la capacité du système de routage de requêtes pour déterminer la proximité d'un client à la porteuse.
- Les utilisateurs qui partagent un serveur DNS d'un site client unique seront redirigés vers le même ensemble d'adresses IP pendant l'intervalle de TTL. Cela pourrait conduire à une surcharge du serveur réplica pendant une grande demande.

La redirection DNS est simple, transparente et évolutive. En raison de ces avantages, une majorité de réseaux CDN l'utilise comme un mécanisme de routage des requêtes [SPV, 03]. Cependant, cette version basique de la redirection DNS possède plusieurs limites [PAB, 08] [BCN, 03]. D'une part, elle augmente la latence du réseau en raison de l'augmentation du temps de résolution DNS. Pour résoudre ce problème, les administrateurs du CDN

divisent leur DNS en deux niveaux (DNS de bas niveau et DNS de haut niveau) pour répartir la charge [KWZ, 01]. D'autre part, la sélection du meilleur serveur réplica se base sur la proximité des serveurs réplica du serveur DNS local plutôt que sur leur proximité géographique du client terminal. Ainsi, lorsque le client et le serveur DNS local ne se trouvent pas à proximité, la redirection DNS peut rediriger le client vers le serveur réplica plus éloigné entraînant ainsi des performances médiocres perçues par ce client terminal. Par ailleurs, le DNS local peut ne pas être invoqué pour contrôler toutes les requêtes entrantes en raison de la mise en cache de données DNS à la fois au niveau du fournisseur d'accès internet et du client. En effet, le serveur DNS local peut avoir le contrôle sur 5% des demandes dans de nombreux cas [CCC, 02].En dernier, au regard du fait que les clients n'ont pas accès aux noms de domaine réel qui servent leurs demandes, cela peut conduire à l'absence de tout autre serveur pour répondre aux demandes du client en cas de panne du serveur réplica cible. Ainsi, afin de continuer de répondre aux conditions changeantes du réseau ou du serveur, la redirection DNS doit éviter le cache ou les décisions au niveau du client [PAB, 08].

### VI.1.4 Anycast

La technique dite Anycast est un service réseau du protocole IP applicable aux situations dans lesquelles un utilisateur souhaite localiser un hôte (un serveur) qui prend en charge un service particulier et que ce service est répliqué sur plusieurs serveurs [PMM, 93]. Comme nous allons voir dans la sous-section V.3, la technique Anycast IP possède plusieurs lacunes notamment lorsqu'elle route les datagrammes vers le serveur réplica le plus proche sur la base de métriques de routage comme le nombre de sauts. Pour pallier à cela, des chercheurs ont proposé d'opérer le concept Anycast à un plus haut niveau (couche application) afin de pouvoir manipuler d'autres métriques de sélection comme la charge du serveur [PEN, 08]. Dans ce contexte, Fei et al. [FBZ, 98] ont proposé un mécanisme Anycast au niveau de la couche application dans lequel le service est composé d'un ensemble de résolveurs Anycast. Ces résolveurs font la correspondance entre un nom de domaine Anycast (Anycast Domain Name, ADN) et une ou plusieurs adresses IP. En effet, un ADN identifie de manière unique une collection des adresses IP (potentiellement dynamique) qui constitue un groupe Anycast. Les clients interagissent avec les résolveurs Anycast en générant une requête Anycast. Une base de métriques, associée à chaque résolveur Anycast, contient les mesures de performance (la charge et la capacité de traitement des requêtes) des serveurs réplica. Ces données de performance peuvent être utilisées pour sélectionner le meilleur serveur réplica en se basant sur des critères de sélection spécifiées par l'utilisateur. Ce mécanisme de routage donne une meilleure flexibilité. Par contre, il nécessite des modifications au niveau des serveurs réplica ainsi qu'au niveau des clients. Compte tenu du nombre potentiellement important de serveurs et de clients, il peut par

conséquent conduire à l'augmentation des coûts. D'autres mécanismes Anycast au niveau de la couche application se trouvent dans [PEN, 08].

## VI.2 Niveau de la couche transport

A la couche de transport, les niveaux de granularité peuvent être atteints au moyen d'un examen plus approfondi des requêtes du client. Ce niveau fournit des informations sur l'adresse IP du client, le port TCP, et autre couche de quatre informations d'en-tête. Ces données pourraient être utilisées en conjonction avec d'autres mesures de l'état de charge pour sélectionner le serveur réplica qui est mieux adapté pour servir une requête donnée. En général, le flux de trafic vers l'avant (client au serveur réplica sélectionné) s'écoule à travers le serveur de routage de requêtes ou par l'intermédiaire d'une première étape de réplication choisi initialement par le DNS. Le trafic à contre-courant (substitut au client), qui transfère normalement beaucoup plus de données que le flux vers l'avant, va typiquement prendre le chemin direct de serveur réplica.

Dans ces mécanismes [BCN, 03], le premier paquet de demande du client sera envoyé au serveur d'origine qui le retransmet au SRR. Celui-ci inspecte ensuite les informations disponibles dans ce premier paquet (l'adresse IP du client, le numéro du port et le protocole de la couche transport) pour sélectionner le serveur réplica approprié. Toutefois, les mécanismes de routage de la couche transport sont généralement combinés aux mécanismes à base de DNS de telle sorte que le premier paquet de demande du client sera d'abord envoyé au serveur réplica initialement choisi par le DNS. Le SRR décide ensuite s'il raffinera son choix en relançant le processus de sélection avec plus de précision. Ils sont mieux adaptés aux longues sessions comme les sessions FTP et RTSP. A contrario, ils pourraient également rediriger le client vers des serveurs réplica surchargés.

## VI.3 Niveau de la couche réseau

Cette solution vise à résoudre le problème de routage de requêtes au niveau de routage de paquets IP. Le principe de base est qu'un groupe de serveurs fournissant le même service peut être traitée au moyen d'un nom anycast et une adresse anycast. Un utilisateur désirant accéder à certains services, par exemple, un contenu donné, de tous les serveurs (équivalents) émet une requête avec le nom anycast. Ceci est mappé à l'adresse anycast et la demande est envoyée dans le réseau avec l'adresse anycast comme destination. Le rôle de service anycast est de rediriger ces requêtes vers l'un des serveurs, sélectionnant ainsi le serveur qui servir la requête du client. Comme le système de redirection a pour but évident d'améliorer les performances des clients, la redirection est basée sur des critères de performance, par exemple réduire le temps de réponse perçu par l'utilisateur. La redirection est donc réalisée par un routeur d'accès coopératif qui est capable de sélectionner le meilleur serveur réplica adapté à partir de la table anycast.

Anycasting est un mécanisme de localisation plus élaboré qui cible la réplication des serveurs au niveau du réseau sur les plates-formes hétérogènes potentiellement. Un mécanisme de redirection de requête qui est basé sur le concept de l'anycasting doit permettre le maintien des informations sur l'état et la performance des serveurs. Un service anycast peut être réalisé à différents niveaux de la pile de protocoles de réseau. À la couche réseau, les mécanismes de l'anycasting consistent à associer une adresse IP anycast commun avec le groupe de serveurs répliqués. Le protocole de routage route les datagrammes vers le serveur le plus proche, en utilisant la distance métrique de routage. Les protocoles de routage standard intra-domaine unicast peuvent accomplir cela, en supposant que chaque serveur annonce l'adresse IP commune. Dans [PMM, 93] les auteurs proposent une solution du problème de sélection de serveur en introduisant la notion d'anycasting dans la couche réseau. Ce travail a établi la sémantique de service anycasting au sein de l'internet. Au niveau du réseau la mise en œuvre d'un service anycast implique les mécanismes suivants :

- Anycast demande l'interception, qui peut être traitée au niveau du réseau par les routeurs de périphérie. Les routeurs de bord sont paramétrés pour filtrer des paquets avec l'adresse de destination anycast.
- Mécanisme de traduction nom anycast vers adresse anycast.
- Sélection de serveur qui peut être traitée par un module de routeur de bord (ou par une application interagissant avec le routeur). Ce module intéragit avec le module de mesure et conserve et met à jour les indicateurs de performance du serveur anycast.
- Traduction adresse anycast vers adresse IP qui peut être traitée au niveau du réseau par les routeurs de périphérie. Sur réception d'une adresse anycast, les routeurs de périphérie réalisent une redirection en les traduisant en une adresse unicast sélectionnée.
- Collection de mesures utilisée dans le processus de sélection.

La technique Anycast IP vise la réplication à l'échelle du réseau des serveurs réplica sur des plates-formes potentiellement hétérogènes [PAB, 08]. Cependant, il possède certains inconvénients notamment [BCN, 03]:

- des parties de l'espace d'adresses IP sont allouées aux adresses Anycast.
- typiquement, les protocoles de routage ne sont pas sensibles à la charge du réseau. Ainsi, le serveur réplica le plus proche peut-être celui avec une longue latence réseau.
- la charge du serveur n'est pas considérée durant le routage de la requête.

# Exemple d'un protocole du méta-routage [ARO, 12]

Dans cette section, nous essayons d'expliquer le protocole de méta-routage sur lequel nous nous sommes basés tout au long de notre travail.

Dans [ARO, 12], Aroussi a proposé un nouveau protocole de méta-routage qui permet de rediriger en temps réel, et de manière adaptative, les requêtes de l'utilisateur au niveau de la phase résolution des noms (Domain Name System, DNS) vers le meilleur serveur vidéo offrant la meilleure qualité d'expérience (Quality of Experience, QoE) de l'utilisateur final. Cette dernière peut être estimée grâce à un modèle de corrélation avec les paramètres de la qualité de service (Quality of service, QoS) comme le délai, la gigue et le taux de perte. Aussi, dans cette première version du protocole, l'auteur s'intéresse seulement au SRR dans un CDN. Concernant la distribution de contenus vidéo, il suppose que la vidéo est répliquée soit en entier (Full Site) ou à priori (Push) dans les serveurs réplicas. Nous allons détaillerce travaille dans le chapitre suivant.

Ainsi, l'architecture adoptée du CDN comporte trois entités :

- Les clients qui effectuent des requêtes (demandes) de contenus vidéo et reçoivent les flux vidéo en provenance des serveurs réplica.
- Les serveurs réplicas qui envoient les flux vidéo aux clients. Ils fonctionnent sous le contrôle de serveur redirecteur à qui ils envoient parfois des demandes de remplacement.
- Le serveur DNS-CDN (parfois appelé serveur redirecteur) qui désigne le meilleur serveur réplica pouvant répondre à la demande du client. Pour ce faire, il maintient trois tables différentes :
    - La table des contenus disponibles qui contient la liste de tous les contenus vidéo disponibles dans le CDN avec leurs emplacements. Une vidéo est éventuellement présente sur plusieurs serveurs réplica.
    - La table des serveurs réplica connectés qui permet de tenir à jour la liste de tous les serveurs réplica actifs (en marche) dans le CDN. Notons ici que lorsqu'un serveur réplica se connecte ou se déconnecte dans le CDN, il en informe le serveur redirecteur afin de mettre à jour cette table ainsi que la table des contenus disponibles.
    - La table des demandes en cours de traitement qui maintient à jour les données mesurées au niveau du serveur réplica traitant la demande.

## VII.1 Les phases du protocole

Comme illustré dans la Figure V.1, le fonctionnement du protocole de méta-routage passe par cinq phases : initiation, sélection, connexion, rétroaction et adaptation. Ces phases sont décrites et expliquées dans les sous-sections suivantes.

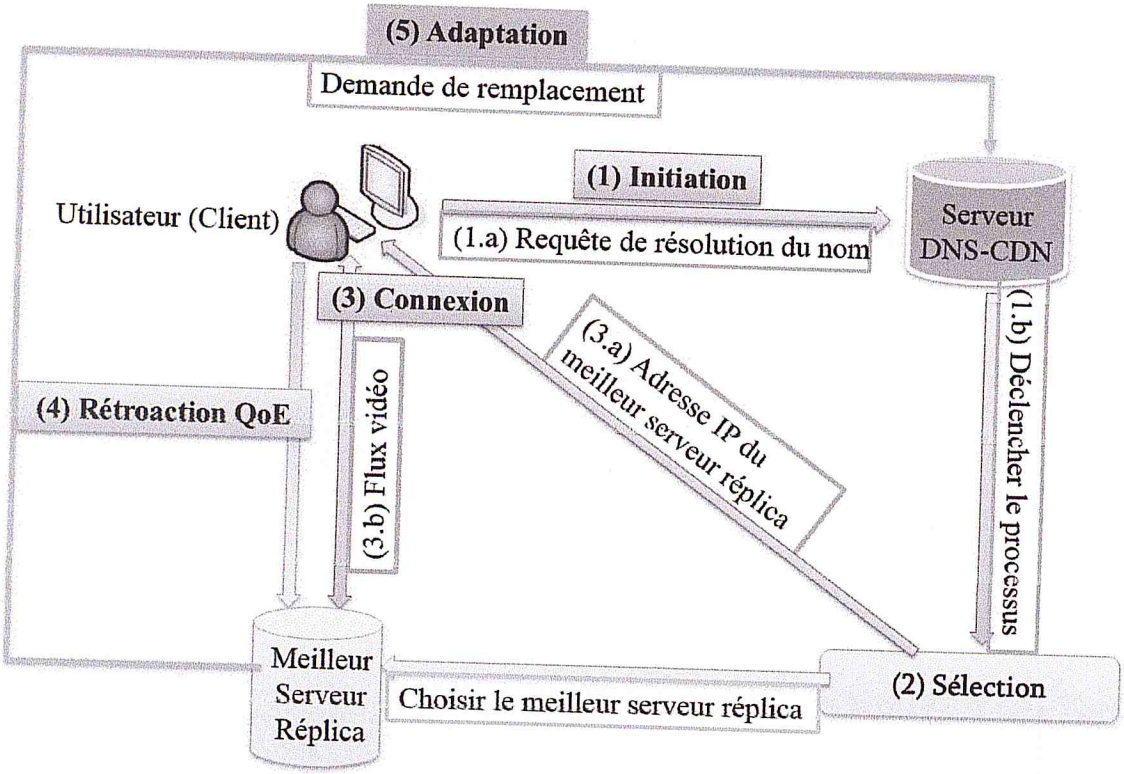



Figure III.1 les phases du protocole de méta-routage [ARO, 12]

### VII.1.1 Phase d'Initiation

Lorsqu'un client initie une demande de vidéo, une requête de résolution de nom est envoyée de manière transparente au serveur DNS-CDN. A partir de la table des contenus disponibles, le serveur DNS-CDN vérifie que le contenu demandé est disponible au moins dans un serveur réplica du CDN. Dans le cas contraire, la demande sera retransmise au serveur DNS local. Sinon le processus est déclenché.

### VII.1.2 Phase de Sélection

Le processus de sélection du meilleur serveur réplica se déroule comme suit :

a. Le serveur DNS-CDN diffuse une requête à tous les serveurs réplicas ayant le contenu demandé en leur demandant d'envoyer leur estimation de la QoE. A cette fin, la requête contient :
   - L'adresse IP du client cible afin de permettre aux serveurs réplica de mesurer la performance du réseau entre eux et le client.
   - Le nom du contenu afin de le mettre en cache et/ou le mettre à jour selon la politique du serveur.

- Le délai d'expiration (timeout) de la requête au bout de laquelle le serveur réplica doit répondre.

b. Chaque serveur réplica mesure la performance du réseau, en termes de latence et taux de perte, entre lui et le client afin d'estimer la QoE. Cette estimation est ensuite envoyée au serveur DNS-CDN avant l'expiration du timeout.
c. Une fois le timeout expiré, le serveur DNS-CDN désigne le serveur réplica ayant la meilleure estimation.

### VII.1.3 Phase de Connexion

Une fois le Meilleur Serveur Réplica (MSR) est sélectionné, le serveur DNS-CDN répond à la requête de résolution du nom du client en communiquant l'adresse IP de ce serveur (MSR). Une session RTP/RTCP est ensuite établie entre le MSR et le client. La transmission de la vidéo sur le réseau utilise en effet les protocoles RTP/RTCP [SCF, 03] (Cf. section IV du chapitre I). Ces protocoles utilisent des ports différents. Le protocole RTP utilise un numéro de port pair, et le protocole RTCP le numéro de port impair qui suit directement. Lorsqu'une session RTP est ouverte, une session RTCP est aussi ouverte de manière implicite. Les numéros de port utilisés par les protocoles RTP/RTCP sont compris entre 1025 et 65535. Les ports RTP et RTCP par défaut sont respectivement 5004 et 5005.

### VII.1.4 Phase de Rétroaction

Le protocole RTCP est basé sur des transmissions périodiques des rapports de rétroaction entre le client (récepteur) et le serveur réplica (émetteur). Parmi ces rapports, le rapport RR (Receiver Report) contient des informations statistiques de la qualité de la livraison des paquets RTP reçus au niveau du client. Ces informations permettent de calculer, entre autres, la latence et le taux de perte de paquets qui sont ensuite utilisées pour estimer la QoE. Le serveur réplica décide ensuite d'envoyer une demande de remplacement au serveur DNS-CDN dans le cas où la QoE atteint un seuil critique durant un intervalle donné.

### VII.1.5 Phase d'Adaptation

L'adaptation de notre protocole est basée essentiellement sur les demandes de remplacement envoyées au serveur DNS-CDN. Elle consiste en effet à rediriger le client vers un autre serveur réplica meilleur que le serveur courant sans interruption du service dans l'objectif de maintenir un bon niveau de performance perçue durant toute la session du transfert. Cette opération, connue sous le terme anglais « midstream handoff », doit se faire de manière transparente au client. Nous tenons aussi à préciser que dans le cas où le serveur DNS-CDN ne reçoit aucune demande de remplacement durant un certain intervalle

assez long (qu'il faudra paramétrer), le serveur DNS-CDN lance le processus d'équilibrage de charges entre les serveurs réplicas.

## VII.2 Extension du NS-2

La mise en place de ce nouveau protocole nécessite sa validation par la simulation ou l'expérimentation sur une plate-forme réelle. Ne disposant pas d'une véritable plate-forme CDN de tests, l'auteur a opté pour la simulation. Pour ce faire, il a été nécessaire de pouvoir simuler la redirection DNS ainsi que la transmission vidéo dans un réseau de distribution de contenus de type vidéo. Etant donné qu'il n'existe aucun simulateur offrant cette possibilité, l'auteur s'est intéressé à des solutions partielles proposées dans la littérature. Parmi ces solutions, il distingue principalement des extensions sur le simulateur réseau NS-2 permettant de simuler la redirection DNS dans un CDN comme dans [SHZ, 03], [CFO, 10] et [BHM, 11] ou de simuler la transmission vidéo à travers le protocole RTP/RTCP comme dans [BGK, 08], [ZHJ, 08], [FRI, 09] et [BMV, 10]. Après avoir passé en revue ces principales solutions, Aroussi a opté pour une nouvelle extension sur NS-2 combinant à la fois la solution de Shen et Zhao [SHZ, 03] avec celle de Boronat et al. [BMV, 10] pour simuler respectivement la redirection DNS et la transmission vidéo.

La figure III.2 présente l'architecture de l'ensemble des outils de simulation nécessaires à la validation de ce protocole de méta-routage. Cette architecture passe par deux étapes : la génération du fichier de trace du trafic vidéo et la simulation de réseau avec NS-2. En effet, les modèles du trafic dans NS-2 (CBR, VBR, ...) ne permettent pas de représenter le modèle de trafic vidéo, il faut donc générer un fichier de trace du trafic vidéo compatible avec NS-2 afin de rendre la simulation plus réaliste. Il faut configurer un serveur VLC pour qu'il diffuse, en local et via un stream RTP/MPEG, un fichier vidéo encodé. En parallèle, l'outil rtpdump est lancé en écoute sur le même port afin d'enregistrer les informations des paquets RTP dans un fichier de sortie. Ce dernier fichier sera ensuite converti en un fichier de trace de trafic compatible avec NS-2 contenant des informations utiles telles que le numéro de séquence, la taille et le temps d'émission pour chaque paquet RTP. Dans la deuxième étape, l'auteur a introduit sa propre extension, il a utilisé (i) une version modifiée de RTP_gs [BMV, 10], qu'il l'a appelé RTP-CDN, permettant d'inclure les fonctionnalités supplémentaires caractérisant les clients et les serveurs réplica du CDN, (ii) un nouvel agent DNS_CDN permettant de rediriger les requêtes du client au niveau du DNS vers le serveur réplica ayant la meilleure estimation de la QoE, et (iii) des nouveaux paquets permettant de communiquer entre les clients, les serveurs réplicas et le serveur DNS_CDN

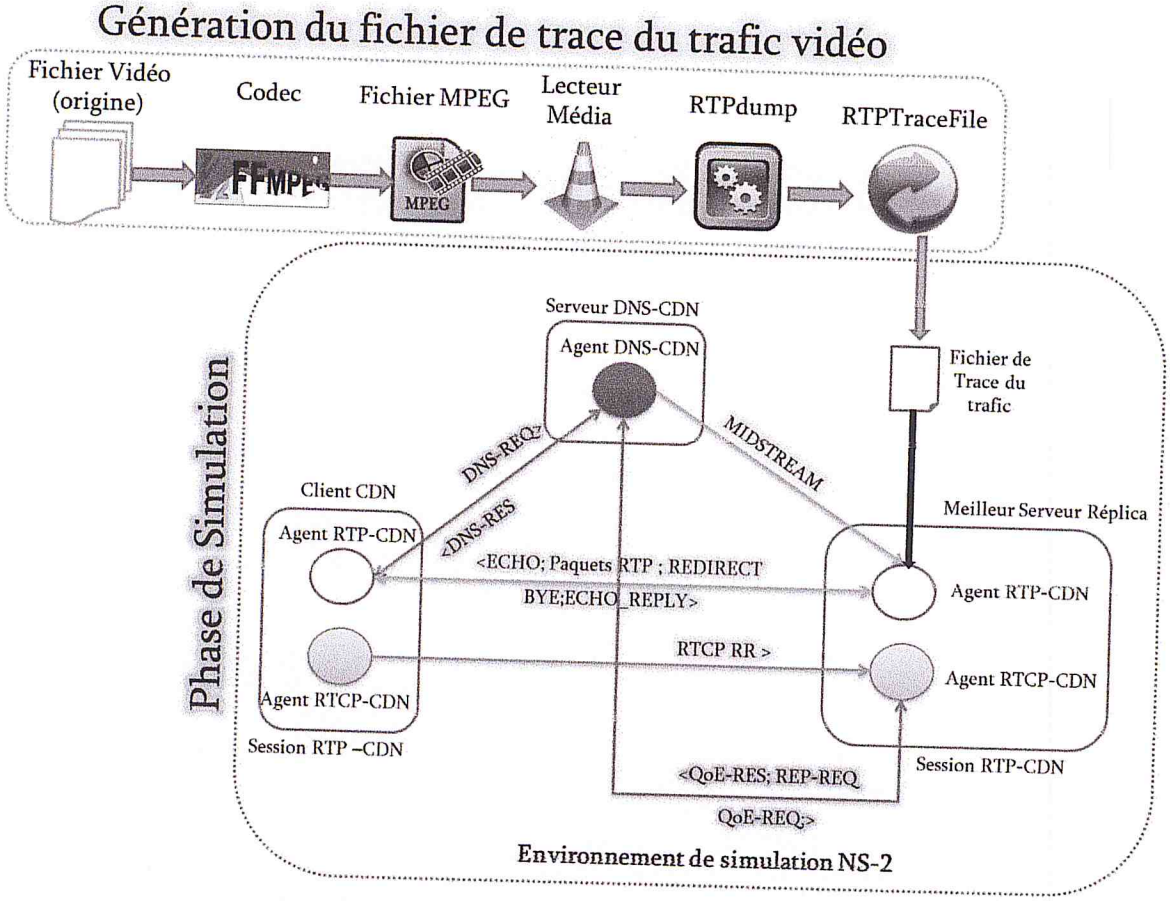



Figure III.2. L'architecture de l'ensemble d'outils de simulation [ARO, 12]

**Dans ce mémoire, nous avons essayé de mettre en œuvre cette extension. Cependant, nous avons rencontré plusieurs problèmes techniques, ce qui nous a poussés à revoir cette proposition et à proposer autre solution plus simple. Cette solution fait l'objet du chapitre IV.**

## CONCLUSION

Nous avons présenté l'état de l'art sur les CDNs. Tout en commençant par des généralités sur les CDNs et les relations entre ses principaux composants : les clients, les serveurs réplicas, le serveur d'origine, le système de distribution de contenu (SDC), le système de routage de requêtes (SRR) et le système de comptabilité (SC). Par la suite nous avons précisé et expliqué le fonctionnement de SRR qui consiste à sélectionner le meilleur serveur réplica et rediriger le client vers ce serveur par un mécanisme de redirection. Et à

la fin nous avons montré un exemple d'un protocole de méta-routage que nous allons implémenter.

# CHAPITRE III

## Simulateur Réseau NS-2

# INTRODUCTION

Simulateur de réseau (Network Simulator 2) NS-2[1] est un simulateur open source utilisé largement dans le milieu universitaire et recherche. Il a été publié en 1989 et profondément mis à jour depuis sa publication. Il est basé sur des évènements ponctuels (discrets) et il est capable de simuler des réseaux câblés et sans-fil, des protocoles unicast et multicast, y compris les protocoles de transport TCP et UDP. NS-2 est un simulateur orienté objet, écrit en C++ et OTCL, il couvre un très grand nombre d'applications (web, ftp, telnet), d'éléments de réseau (Nodes, Link et Queuing), et de modèles de trafic (CBR, VBR, Exponential, ...). Tant que NS-2 est open source (logiciel libre), il permet ainsi d'apporter des modifications à son code source afin d'ajouter des nouveaux protocoles et de nouvelles fonctionnalités. C'est pourquoi il est populaire dans la communauté scientifique et fait l'objet dans plusieurs travaux de recherche.

D'origine, NS-2 ne permet de simuler les principales caractéristiques du CDNcomme la distribution de contenu et le routage des requêtes. Néanmoins, dans le cadre de la redirection DNS des flux de vidéo, il existedes solutions partielles permettant soit de simuler la redirection DNS dans un CDN soit de simuler la transmission vidéo à travers le protocole RTP/RTCP.Nous allons faire une brève présentation de ces solutions.

Nous allons expliquer dans ce chapitre ces travaux de recherches.

## TRANSMISSION VIDEO

RTP fournit un service pour délivrer les données en temps réel comme les données audio et vidéo, RTCP est le compagnon de RTP, les émetteurs et récepteurs envoient des feedbacks RTCP afin de contrôler la qualité des paquets envoyés. RTP/RTCP circule sur le protcole UDP (User Datagram Protocol). Les protocoles sont implémentés comme des Agents dans NS-2, donc RTP et RTCP sont implémentés en utilisant les classes RTPAgent et RTCPAgent respectivement. Les deux classes dérivent de la classe Agent et sont implémentés dans le fichier rtp.cc et rtcp.cc qui se trouvent dans le répertoire de NS-2.

[1]Disponible sur www.isi.edu/nsnam/ns/

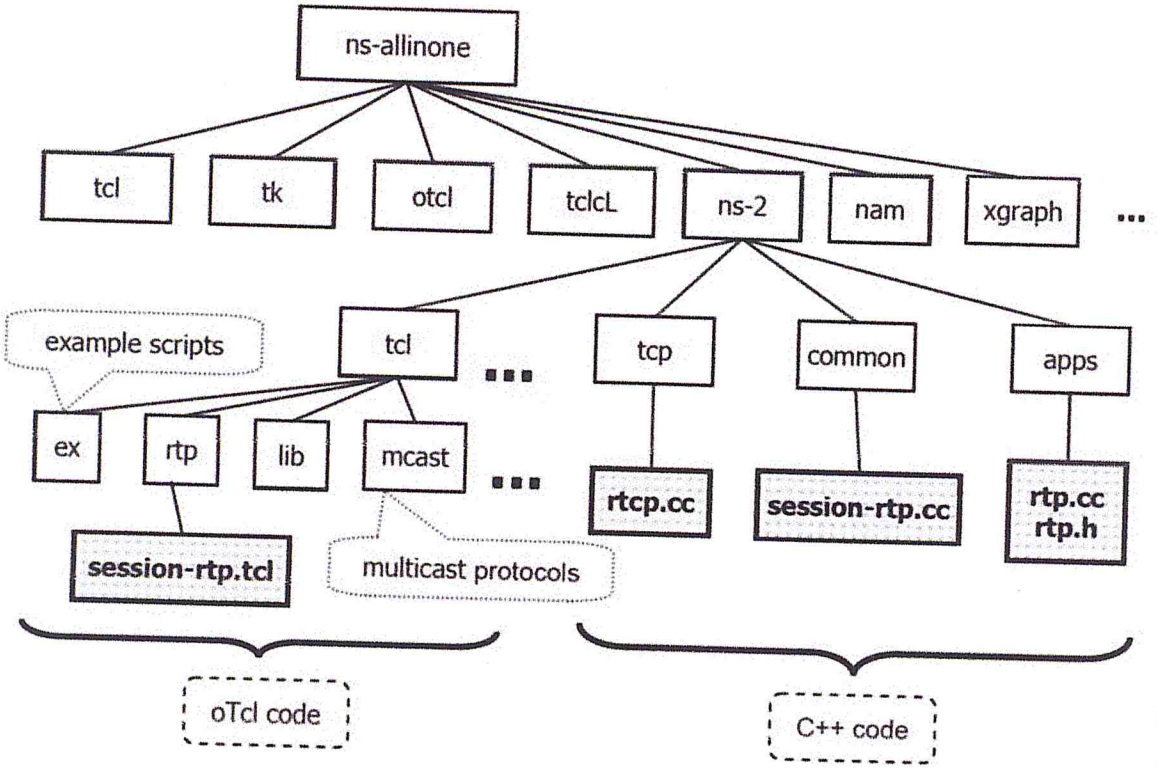



Figure IV.1. Structure du répertoire RTP dans NS-2[BMV, 10].

## II.1 Bouras et al. [BGK, 08]

Les auteurs ont pu mettre une nouvelle extension de la fonctionnalité de l'implémentation RTP/RTCP sur Network Simulator NS-2 pour prendre en charge les contrôles de congestion TCP.Ils ont modifié le code de l'implémentation RTP/RTCP en ajoutant des nouvelles fonctionnalités définies dans RFC 3550 [SCF, 03] et reliées aux métriques de QoS. Ils ont implémenté aussi des algorithmes additionnels et des fonctions afin d'améliorer leur code modifié avec le comportement de la bande passante partagée sur TCP. Ce protocole, nommé RTPUP (UP est l'abréviation de University of Patras), est offert comme un package et totalement documenté pour qu'il soit utilisé dans les simulations et recherches sur NS-2.

Afin de comprendre la structure de RTPUP, les auteurs ont expliqué en détails leur protocole, premièrement renommer l'entête de paquets RTP de « *hdr_rtp* » vers « *hdr_rtpup* » pour reconnaitre leur code et le code orinigal. Aussi définir des nouvelles structures de données nommées « *server_report* » et « *receiver_report* » pour stocker les champs de RTCP, SR et RR respectivement. Une nouvelle classe nommée *RTPUPReceiver* est déclarée pour tenir les champs utilisés par les agents récepteurs pour les mesures de qualité de service. Chaque nouvelle instance de la classe *RTPUPSession* crée deux instances de *RTPUPSource* et une instance de la classe *RTPUPReceiver* la classe RTPUPSession est

appelée par le script TCL et en retour deux nouveaux agents (RTPUPAgent et RTCPUPAgent) sont assignés à chaque nœud participant aux streaming sur le réseau. RTPUPAgent est responsable d'envoyer et recevoir les paquets RTPUP, alors que RTCPUPAgent est responsable de la transmission et la réception des rapports SR et RR.

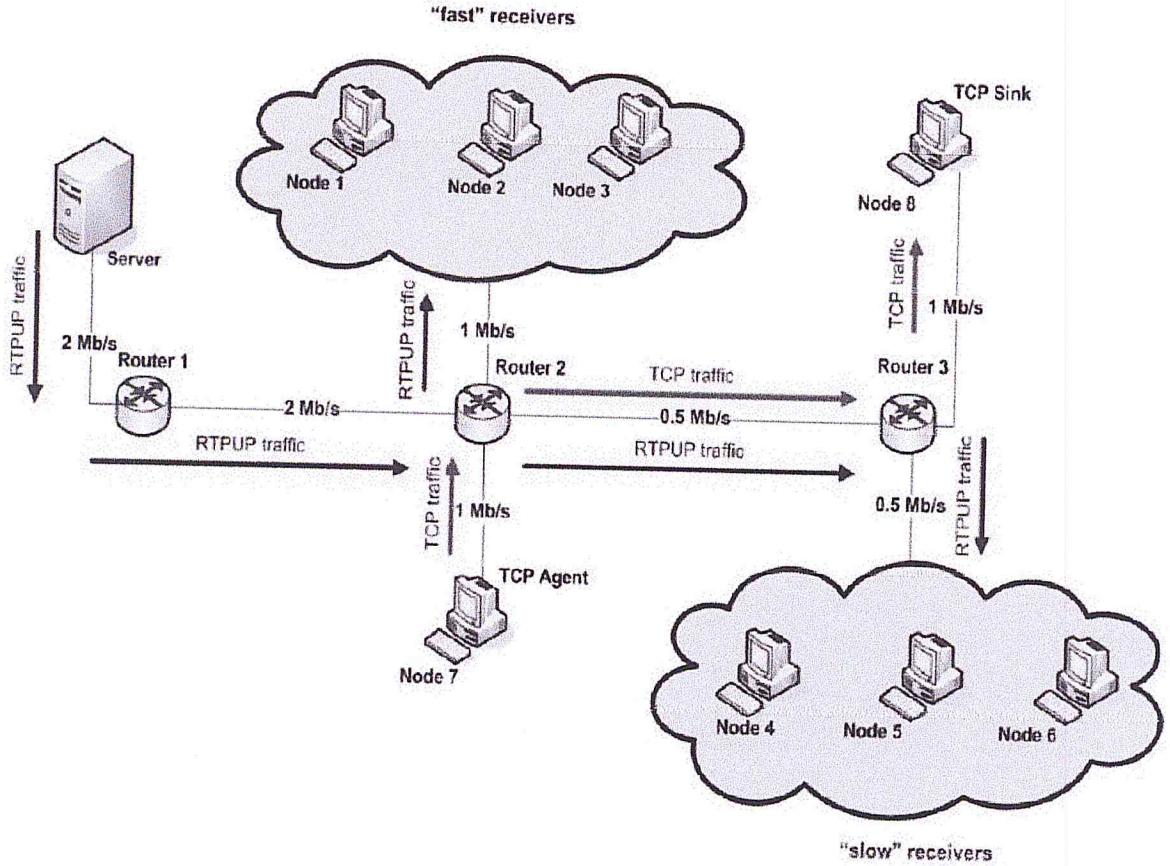



Figure IV.2. Simulation de topologie de réseau [BGK, 08]

Dans cette implémentation RTPUP nous trouvons trois fonctions/modules principales :

- Paquets Send et Receive RTPUP : les paquets sont générés à base d'évènement de délai (timeout) du RTPUPTimer. L'agent RTPUP crée un nouveau paquet RTPUP en appelant la fonction d'envoi :

```
void RTPUPAgent::sendpkt(){}
```

La fonction d'envoi invoque la fonction de production de paquet, qui crée le nouveau paquet RTPUP et ajoute les nouveaux champs dans l'entête du paquet :

```
void RTPUPAgent::makepkt(Packet* p){}
```

Le nombre de séquence du paque RTP :

```
rh->seqno() = seqno_++;
```

L'id source de la source d'envoi :

```
rh->srcid() = session_->srcid();
```

L'horodatage :

```
rh->timestamp()= timestamp_;
```

Les récepteurs dont l'émetteur fait une session :

```
rh->receivers_= session_->receivers_;
```

- Fonction de création de rapports d'envoi et de réception RTPUP : la fonction de création est appellée RTCPUPAgent qui est un résultat de l'évènement timeout de RTCPUPTimer. L'émetteur génère un nouveau SR s'il a envoyé des paquets RTPUP depuis l'ancien SR :

```
//add sender report
sender_report* sr;
//fill in the report
sr = new sender_report;
//assign the sender's id
sr->sender_srcid()= localsrc_->srcid();
//assign the RTPUP packets sent
sr->pkts_sent() = localsrc_->np();
//assign the total bytes sent
sr->octets_sent() = localsrc_->nbytes();
//include the receivers served
sr->rcvr_ = receivers_;
//store the report
rh_->sr_ = sr;
```

- Fonction de contrôle de réception de paquets RTCPUP.

## II.2 Zhou et Jang [ZHJ, 08]

Les auteurs de ce travail ont proposé un système appelé VSS (Video Stream Simulation) pour évaluer la qualité des vidéos délivrées en utilisant les traces trafic dans un environnement de simulation de réseau (NS-2) RTP/UDP/IP. L'architecture de VSS inclut trois étapes :

- Génération de fichier de trace vidéo
- Simulation NS-2
- Evaluation de streaming vidéo.

La première étape consiste à générer le fichier de trace vidéo. Pour rendre la transmission vidéo adaptative, dans cette étape on génère plusieurs fichiers de trace dans N délais différents. La deuxième étape est la simulation de réseau. Elle joue le rôle d'une boite noire dans le système. Les chercheurs peuvent changer la topologie intérieure s'ils le veulent. Dans ce travail, les auteurs ont préféré la structure de réseau RTP/UDP/IP. On génère le fichier de trace émetteur et fichier de trace récepteur pour enregistrer les détails de paquets durant le processus de simulation dans cette étape. A la fin, en utilisant les fichiers de trace générés durant la simulation et les vidéos encodées dans la première étape, on peut générer la vidéo reçue. Après le décodage de la vidéo construite, la transmission peut être évaluée en calculant le Peak Signal Noise Ratio (PSNR) ou autres métriques. Les détails sur l'architecture de VSS sont expliqués dans la figure IV.3.

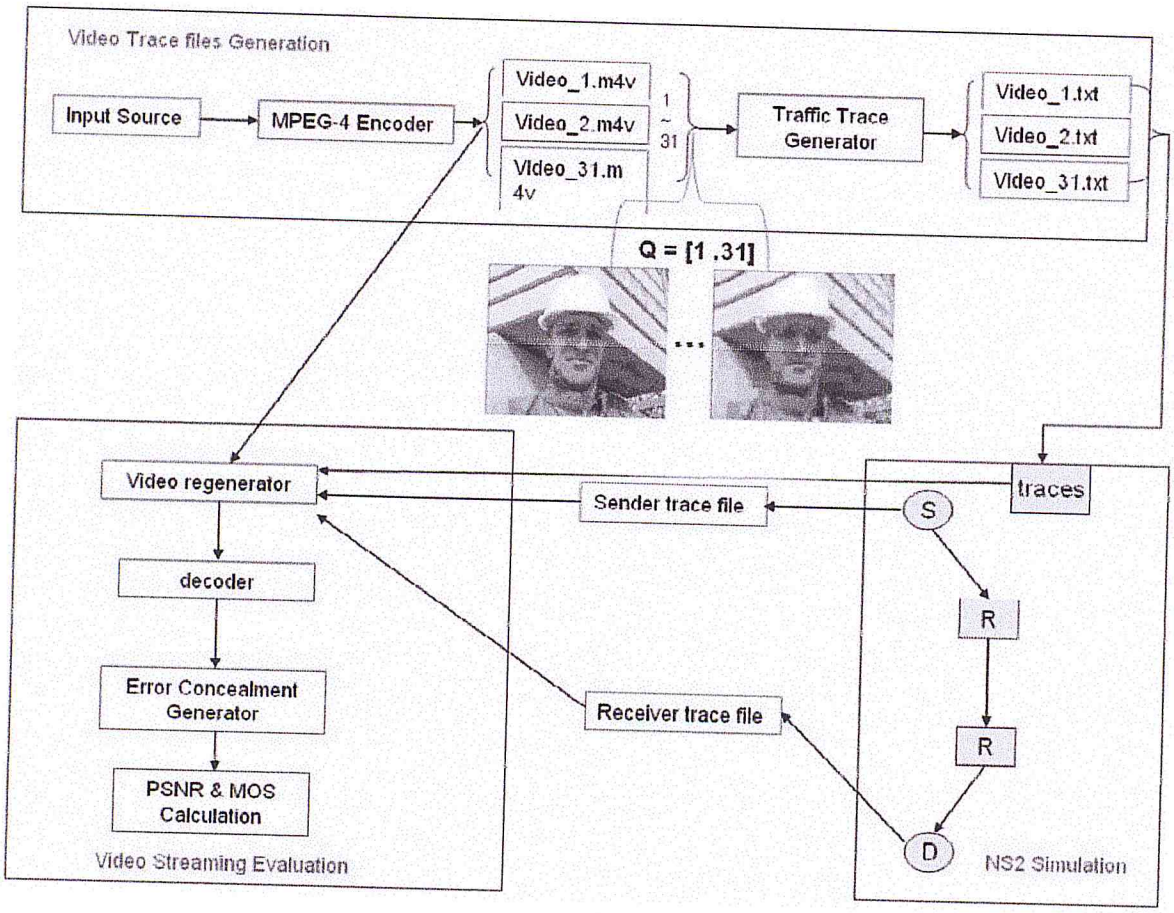



Figure IV.3. L'architecture de VSS [ZHJ, 08].

RTP/UDP/IP sont devenus des standards pour le streaming vidéo, mais la transmission RTP/RTCP est très faible sur NS-2. On ne trouve pas le feedback par exemple donc les auteurs ont amélioré RTP/RTCP en rajoutant une extension qui contient :

- L'ajout de feedback (Sender Report et Receiver Report)

- Ajouter une stratégie TFRC basée sur RTP pour tester les performances et les comparer.
- Ajouter une classe étendue dans le fichier de trace pour l'interface entre les deux premières étapes.
- Générer un fichier de trace de l'émetteur et du récepteur pour les détails de transmission.

Dans la troisième étape, la vidéo reçue est régénérée en utilisant le fichier de trace générélors de la simulation et les vidéos encodées dans la première étape. Cette vidéo est ensuiteévaluée en utilisant la méthode avec référence complète PSNR (Peak Signal to Noise Ratiopour le rapport signal/bruit maximal).

## II.3 Frias [FRI, 09]

L'auteur de ce travail a fait une contribution pour tester la QoS dans les réseaux Ad Hoc mobiles pour les services de streaming vidéo basé sur une architecture de couches adaptatives. L'auteur a réalisé une transmission vidéo via RTP/RTCP réel sur NS-2 en modifiant RTP/RTCP native du simulateur, afin de mesurer la QoS de ce processus en utilisant les métriques : délai, temps de réponse, perte de paquets et la bande passante. Son premier but est de tester son protocole de routage appelé MMDSR (MultiMedia Dynamic Source Routing protocl) qui est basé sur la redirection DSR, cette dernière est capable d'improviser la performance des services de streaming vidéo en appliquant les algorithmes de passage de couches réseaux et techniques de routage multi-chemin. La Figure IV.4 montre un schéma de multipath.

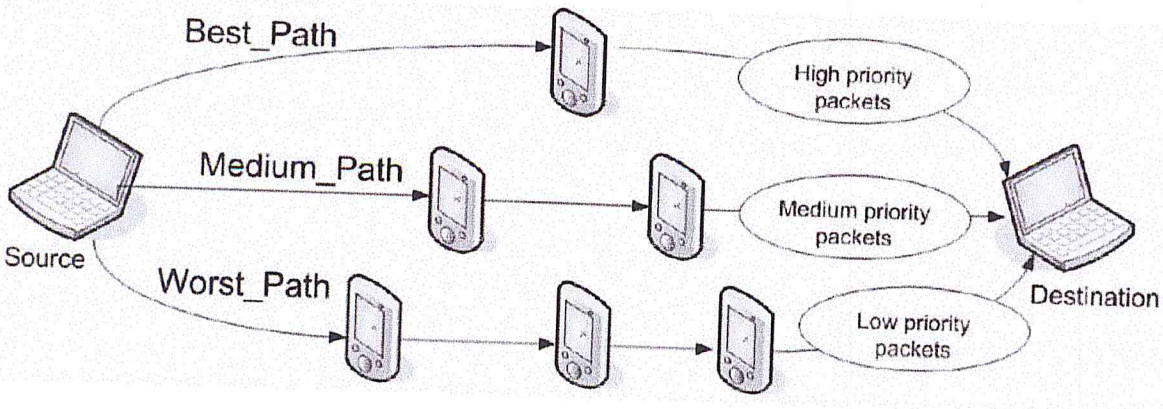



Figure IV.4 Schéma multi-path (multi-chemin) en utilisant trois paths [FRI, 09]

[FRI, 09] a proposé de créer de nouveaux agents RTP et RTCP (nommésrespectivement RTP_v2 et RTCP_v2) avec des fonctionnalités supplémentaires afin defournir le contrôle de la perte et de la gigue dans le streaming vidéo MPEG-2 sur les réseaux sans fil (802.11e). Pour ce faire, l'auteur a défini de nouvelles structures de données

pour générer les paquets RTP et RTCP. En outre, l'auteur a développé deux autres applications pour la distribution vidéo sous NS-2 : (i) un générateur de fichier de trace (appelé mpg2trace) et (ii) une application émetteur/récepteur MPEG-2 capable d'agir virtuellement comme un serveur/client de service streaming vidéo. Enfin, l'auteur évalue la performance en utilisant la méthode PSNR.

L'implémentation de RTP/RTCP inclus dans NS-2.27 est très simple et n'est pas conforme aux spécifications RFC-3550 donc il a été obligé d'améliorer cette implémentation afin de pouvoir mesurer quelques paramètres cette nouvelle implémentation contient :

- Fragmentation RTP : les données sont fragmentées si elles sont plus grandes que la taille maximale à transmettre `Maximum_packet_size`.
- Paquets RTCP : le protocole compagnon de RTP est RTCP donc il est nécessaire d'implémenter les paquets RTCP.
- Période d'envoi de paquets RTCP : la période des paquets RTCP peut être modifiée depuis le fichier TCL de la simulation.
- Mesure de la gigue : les mesures de la gigue sont inclues dans les paquets RTCP
- Calcul de perte de fractions : il est inclus dans le paquet RTCP.
- Identification de numéro de séquence reçu le plus élevé.
- L'horodatage : le temps ou le paquet RTP est créé est inclut dans l'entête du paquet.
- Nombre de paquets de l'émetteur : le nombre total de paquets RTP transmis par l'émetteur est généré depuis le début de la transmission jusqu'à la réception de RR-RTCP.
- Nombre cumulé de paquets perdus : le nombre total de paquets RTP envoyé de la source qui sont perdues depuis le début de la réception.
- Paquets RR : nombre de paquets reçus depuis le dernier RR envoyé.
- Identification de source : chaque source est identifiée par un ID. ce numéro d'identification est inclus dans chaque paquet RTCP.
- Identification de flux : chaque flux de vidéo a un ID.

L'auteur a fait des contributions sur la transmission vidéo pour pouvoir simuler une transmission vidéo en temps réel pour cela il a modifié l'implémenation RTP/RTCP vu précédemment, il a rajouté un outil appelé mpg2trace qui est un générateur de fichier de trace pour qu'il soit compatible avec NS-2. Une application appelée MPEG-2 Sender pour l'envoie de flux vidéo et appelée MPEG-2 Receiver pour la réception. Vu qu'on a choisi cette solution pour la transmission vidéo alors cette solution sera vu en détails dans le chapitre suivant.

## II.4 Monagaud et Boronat et Vidal [BMV, 10]

Les auteurs ont créé un nouveau module RTP/RTCP afin de simuler la transmission vidéo sur NS-2, ce module, appelé RTP_gs, est une extension du protocole natif et

correspond aux spécifications RFC 3550 [SCF, 03]. Cette modification a impliqué l'ajout des nouveaux messages RTCP comme les paquets RTCP SDES, RTCP BYE et RTCP APP.

Les auteurs ont pu intégrer leur nouveau protocole sans remplacer le protocole natif, comme on peut le voir dans la Figure IV.5 le code C++ se trouve dans le répertoire .../ns/rtp_gs et le code OTcl se trouve dans le réperoire .../ns/tcl/rtp_gs.

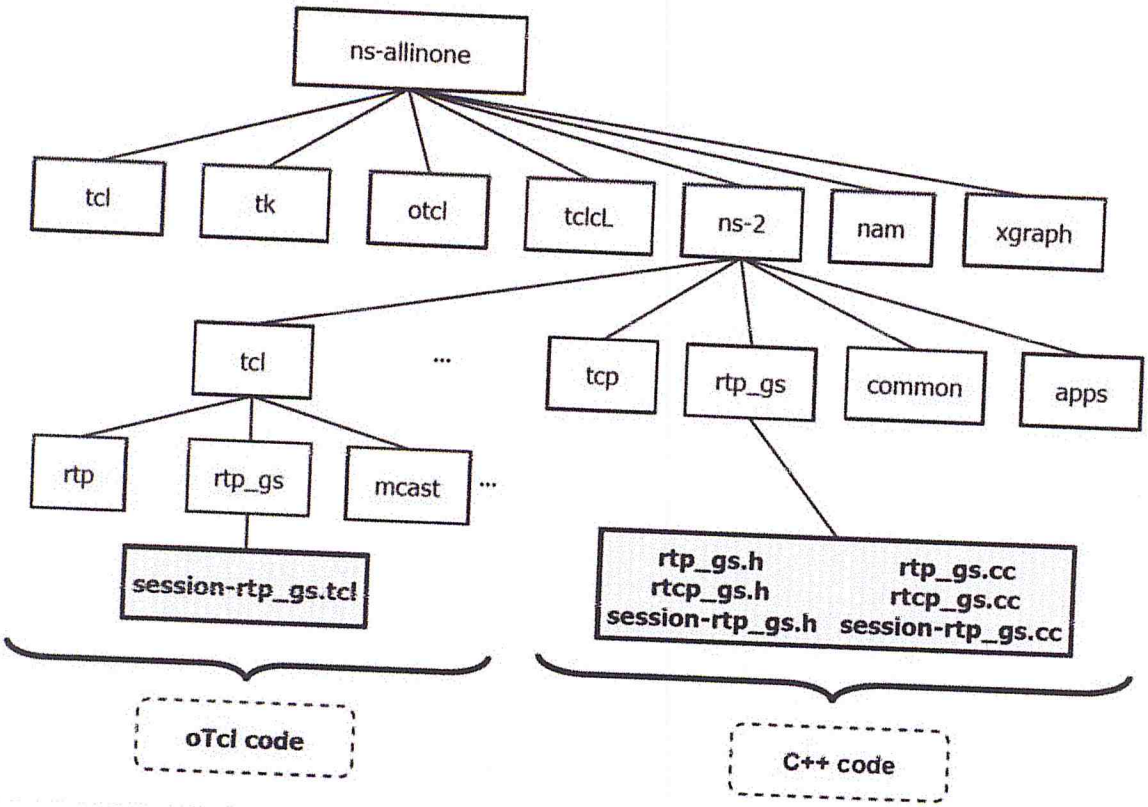



Figure IV.5. Structure de répertoire NS-2 après l'installation de module RTP_gs/RTCP_gs

Boronat et al. [BMV, 10] ont récemment proposé un cadre de simulation précis pourévaluer la QoV en streaming. Ils ont combiné : (i) NS-2 y compris une nouvelleimplémentation du protocole RTP/RTCP (nommé RTP group synchronization, RTP_gs),(ii) une version modifiée de MyEvalvid-RTP [YKC, 08] contenant les outils RTP(RTPdump, RTPplay)33, le lecteur média VLC, le logiciel ffmpeg34 pour le codage et ledécodage MPEG-4 et d'autres programmes pour les mesures de la QoV comme l'afficheurYUV35 et l'outil VQMT36. Comme le montre la figure B.2, l'utilisateur doit suivre troisphases différentes, à savoir la phase de prétraitement, la phase de simulation et la phase depost-traitement pour évaluer la QoV. La première phase a pour objectif de générer unfichier de trace du trafic vidéo compatible avec NS-2 contenant des informations utiles,telles que le numéro de séquence, la taille et le temps d'émission pour chaque paquet RTP.Dans la deuxième phase, les auteurs ont utilisé leur extension du module RTP/RTCP quiinclut les améliorations suivantes: (i) la définition de tous les types de paquets RTCP

(SR,RR, SDES, APP et BYE) avec leur format exact; (ii) la surveillance, la notification etl'enregistrement des métriques QoS au cours de la simulation; (iii) la capacité de traitern'importe quel modèle de trafic d'application; (iv) le support des flux multicast multiplesur le même noeud, v) la compatibilité avec le code NS-2 existant, et (vi) l'inclusion d'unefonctionnalité optionnelle avec l'approche de la synchronisation des groupes dans [MOB,10] . Enfin, la dernière phase a pour objectif de reconstruire le flux vidéo reçu en vued'évaluer sa qualité en utilisant par exemple l'outil VQMT. Le code source, le guided'installation ainsi qu'un exemple de simulation sont disponibles sur :

http://personales.gan.upv.es/~fboronat/Research/NS2_RTP/NS2_RTP_RTCP_module.htm

## II.5 Discussion

Dans [ARO, 12], l'auteur a choisi la solution de Boronat et al [BMV, 10], mais nous avons choisi la solution du Frias [FRI, 09].Ce choix peut être justifié par la facilité de manipuler ce dernier en termes de définition de paquets et génération du fichier de trace.

En effet, Monagaud et Boronat [BMV, 10] ont créé un nouveau module RTP/RTCP afin de simuler la transmission vidéo sur NS-2.Ce module, appelé RTP_gs, est une extension du protocole natif et correspond aux spécifications RFC 3550 [SCF, 03]. Cette modification a impliqué l'ajout des nouveaux messages RTCP comme les paquets RTCP SDES, RTCP BYE et RTCP APP. Dans les fichiers de *rtp_gs*, les auteurs ont modifié l'entête de paquet natif de « hdr_rtp » vers « hdr_rtp_gs », incluant tous les champs spécifiés dans RFC 3550. Ils ont redéfinit l'agent RTP, qui maintient tous les fonctionnalités de l'envoi et réception de données RTP, cet agent appelé *RTP_gs_Agent*. Dans les fichiers de rtcp_gs, ils ont modifié l'agent RTCP qui est responsable de la transmission et la réception des paquets RTCP, appelé « RTCP_gs_Agent ». Ils ont aussi créé un nouvel entête commun pour tous les paquets RTCP appelé « hdr_rtcp_gs ». Comme spécifié dans [SCF, 03], ils ont défini un constant numérique pour chaque paquet RTCP pour l'inclure dans le champ de Playload Type dans l'entête RTCP :

```
typedef enum {
        RTCP_SR = 200,
        RTCP_RR = 201,
        RTCP_SDES = 202,
        RTCP_BYE = 203,
        RTCP_APP = 204
} rtcp_type_t;
```

Les auteurs ont combiné la nouvelle implémentation du protocole RTP/RTCP et une version modifiée de MyEvalvid-RTP [YKC, 08] contenant les outils RTP (RTPdump, RTPplay), le lecteur média VLC, le logiciel ffmpeg pour le codage et le décodage MPEG-4.

Par contre, Frias [FRI, 09] a réalisé une transmission vidéo via RTP/RTCP sur NS-2 en modifiant le protocole natif RTP/RTCP dans l'objectif de mesurer la QoS de ce processus en utilisant des métriques. Les paquets sont définis et conformes aux spécifications RFC 3550 [SCF, 03]. Il a implémenté un générateur de fichier de trace automatique et un émetteur récepteur vidéo via RTP/RTCP.C'est pour ça qu'on a choisi cette solution simplifiée.

## LE ROUTAGE A BASE DE DNS

Dans cette sous-section nous expliquons quelques travaux concernant le routage DNS dans les réseaux de distribution de contenu car notre plateforme NS-2 ne contient pas cette fonctionnalité.

### III.1 Shen et Zhao [SHZ, 03]

Les auteurs ont fait une simulation très proche d'un CDN réel en proposant une topologie de réseau sur NS-2 pour effectuer une redirection DNS. Afin d'aboutir cette implémentation, les auteurs ont introduit quatre nouveaux agents :

- ***cdn_client*** : envoi une requête au *cdn_dns*, une fois qu'il reçoit une réponse du cdn_dns sous forme de paquet, il récupère l'adresse de *cdn_server* (ou *cdn_router*) ensuite il envoie une requête vers ce *cdn_server* (*cdn_router*). lors de la réception des données du *cdn_server*, il suffit de libérer les paquets reçus.
- ***cdn_dns*** : attend les requêtes des *cdn_clients*, dès la réception d'une requête, il effectue une recherche à partir de sa table de hachage, ensuite il retourne l'adresse de *cdn_server* (*cdn_router*) au *cdn_client*, en se basant sur des informations comme l'adresse source du *cdn_client* et le type de contenu demandé.
- ***cdn_server*** : attend la requête du *cdn_client* (ou *cdn_router*), à la réception de la requête il envoie les données demandées au *cdn_client*, en se basant sur l'adresse source du *cdn_client* récupérée du paquet. Dans le cas de la réception de requête du *cdn_router*, après l'envoi de données vers le *cdn_client*, il envoie un paquet de notification au *cdn_router* pour l'indiquer que sa tâche a été faite.
- ***cdn_router***: attend la requête du *cdn_client*, une fois reçue, il cherche dans son parc de serveurs et redirige la requête (avec l'adresse source du *cdn_client*) vers le *cdn_server* approprié en se basant sur le critère de « moins connexion », en même temps il incrémente le nombre de connexion de ce *cdn_server*. A la réception de paquet de notification de *cdn_server*, il décrémente le nombre de connexion de *cdn_server*.

Ces quatre types d'agents sont des sous-classes de la classe *Agent*. Leurs fonctionnalités son implémentées principalement par les méthodes « command » et « recv »

définies dans la superclasse *Agent*. La redirection DNS est assurée par *cdn_dns*, et l'équilibrage de charge locale grâce à *cdn_router*. En correspondance avec les quatre nouveaux agents créés, ils ont également conçu quatre nouveaux types de paquets : PT_CDN_CLIENT, PT_CDN_DNS, PT_CDN_SERVER et PT_CDN_ROUTER.

Pour l'agent cdn_dns il utilise une table de hachage pour pouvoir élaborer ses routages, la clé de cette table de hachage est calculée à base des adresses sources, qui est l'adresse du client qui envoi la requete et le type de contenu que le client recherche, la fonction de hachage est index = ((source – 10) * 10 + content_type ) % length of table

## III.2 Cece et al. [CFO, 10]

Les auteurs ont conçu un système de routage afin d'évaluer les nouveaux approches d'équilibrage de charge au niveau de la couche application. Cette nouvelle extension avec des nouveaux types de datas HTTP et des nouveaux composants d'applications qui sont responsables de traitement de données. Les auteurs ont créé un nouvel agent qui permet la simulation de transfert de données. Pour cela ils ont introduit à l'architecture CDN : (i) un nouveau client HTTP et un serveur HTTP qui représentent des nœuds CDN, (ii) un nouveau mécanisme pour introduire les algorithmes d'équilibrage de charge dans les serveurs. La figure IV.6 le diagramme de séquence d'équilibrage de charge.

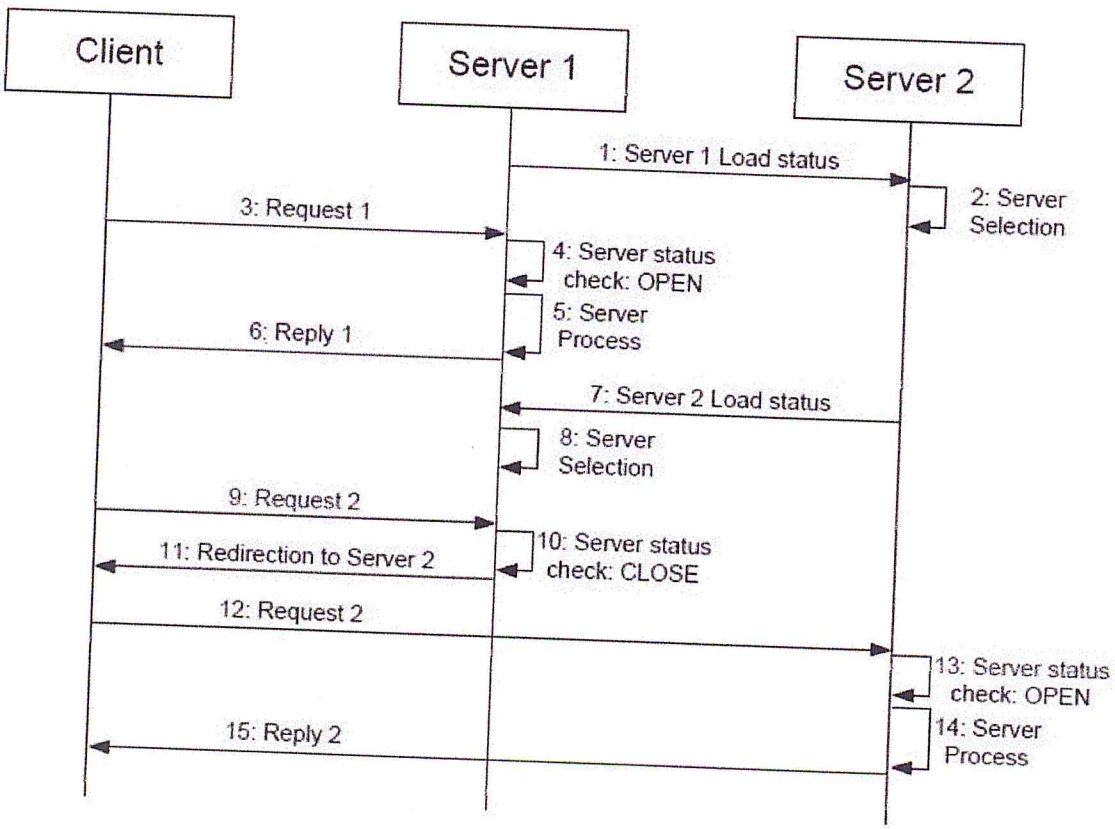


Figure IV.6. Diagramme de séquence de l'équilibrage de charge[CFO, 10]

Pour intégrer cette extension, les auteurs ont introduit plusieurs classes C++ (figure IV.7) :

- *CdnData* : pour les données échangées entre le client et le serveur, c'est une sous classe de *AppData* et elle se compose d'un descripteur de type d'application, identificateur de source, identificateur de la page demandée, identificateur de serveur pour la redirection, les méthodes de messages HTTP, et l'information sur l'état du serveur.
- *CdnAgent* : pour l'envoi et la réception des messages aux nœuds. C'est une sous classe Agent et peut être attachée au nœud par les commandes OTcl.
- *CdnClient* : qui représente le client CDN, elle doit être attachée à un agent, c'est pour ça elle doit contenir un pointeur d'agent, initialisé au *CdnAgent*. le rôle de cette classe est de générer (GET) les requêtes et rediriger (REDIRECT) les messages.
- *CdnServer* : pour traiter les requêtes arrivées et exécuter les algorithmes d'équilibrage de charge et envoyer les messages de redirection aux clients.

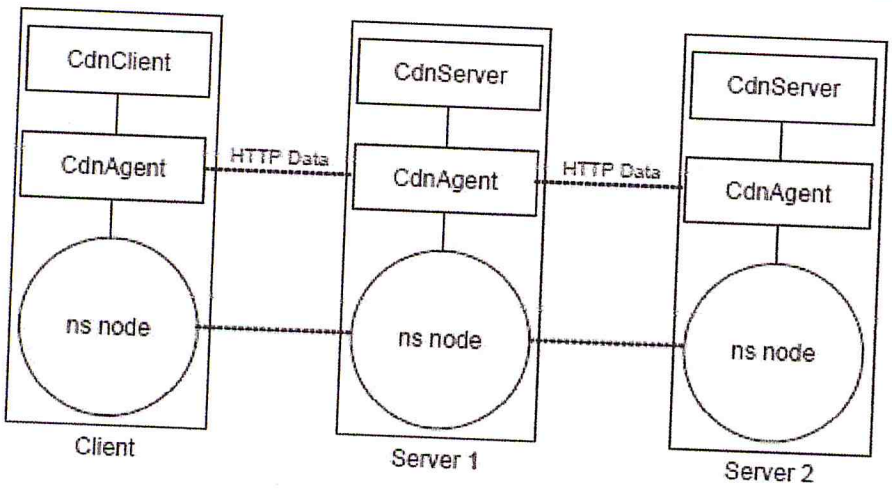



Figure IV.7. L'architecture CDN dans NS-2[CFO, 10]

## III.3 Bhardwaj et Malhotra [BHM, 11]

Les auteurs ont utilisé des techniques de hachage pour la redirection DNS dans les réseaux CDN, leur but était de comparer les différentes techniques et les évaluer. Les auteurs ont utilisé trois techniques de hachage :

- **Modulo hachage** : la fonction de hachage est simplement $h(k) = k \bmod n$ tel que n est le nombre de serveurs. Dans cette approche, l'URL est haché en un nombre modulo le nombre de serveurs i.e k % n. la figure III.8 montre le fonctionnement de cette technique.

- Number servers from 1 to N
- For any URL

  compute h ← HASH("URL")
- Redirect clients to CDN server # p = h mod N

Figure IV.8. Modulo Hashing [BHM, 11]

- **Hachage consistent** : avec cette technique, la sélection de chaque réplica est faite en hachant l'URL. Dans cette approche les adresse URL et IP des serveurs réplicas sont hachées en une longue liste circulaire. URL est assignée au serveur le plus proche dans la liste. Un schéma de cette approche est montré dans la figure IV.9.

- Consider numbers from 0 to n − 1 to be points on a circle
- Map each proxy to several randomly-chosen points..
- Map each URL to a point on circle.
- To map URL to proxy, just find successor proxy along circle.

Figure IV.9. Consistent Hashing [BHM, 11]

- Poids le plus élevée : (Highest Random Weight) cette approche consiste à générer une liste de valeur d'hachage en hachant l'URL et chaque nom de serveur par un poids au hasard, et sortant les résultats. HRW est une méthode similaire de modulo hachage avec des régions de taille non fixée. Chaque URL a un ordre déterministe pour accéder à l'ensemble de serveurs, et cette liste est traversée pour trouver le serveur approprié. On peut résumer ça dans la figure IV.10.

Let list of server addresses be $S_1, S_2, \ldots S_n$

- For URL u, compute:

  $h_1 \leftarrow HASH(S_1, u)$, $h_2 \leftarrow HASH(S_2, u)$, . . .
- Sort $h_1, \ldots h_n$. If hi is minimum, route request to pi.
- If hi overloaded, spill over to next smallest h

Figure IV.10. HRW Hashing [BHM, 11]

Les auteurs ont proposé une simpleimplémentation de la redirection DNS dans le but de comparer entre certaines techniquesde hachage (Modulo Hashing, Consistent Hashing et Highest Random Weight). Cetteimplémentation introduit seulement trois agents dans NS-2: (i) « CDN_CLIENT » pourenvoyer la requête de résolution du nom au « CDN_DNS » et recevoir le contenu demandéà partir du « CDN_SERVER », (ii) « CDN_DNS » pour rediriger la requête du« CDN_CLIENT » au « CDN_SERVER » approprié après avoir exécuté une technique dehachage donnée et (iii) « CDN_SERVER » pour servir le « CDN_CLIENT » avec le contenudemandé en se basant sur les informations reçues de la part de « CDN_DNS ».

## III.4 Discussion

Parmi ces trois solutions, nous avons opté de la même manière que l'auteur de [ARO, 12] et nous avons choisi la solution de Shen et Zhao [SHZ, 03]. Cette solution qui propose quatre nouveaux agents héritant de la classe *Agent*.

# CONCLUSION

Dans ce chapitre nous avons vu les différentesextensions trouvées dans la littérature qui permettent de simuler la redirection DNS ou la transmission des flux vidéo. Ces extensions nous a permis de proposer une autre solution plus simple permettant la direction DNS et la transmission des vidéos à la fois. Cette solution fait l'objet du chapitre suivant.

# CHAPITRE IV

# IMPLEMENTATION DU PROTOCOLE

# INTRODUCTION

L'objectif de notre travail consistait à mettre en place la solution proposée dans [ARO, 12]. Cette dernière se base sur RTP_gs qui nous a posé plusieurs problèmes techniques. C'est pourquoi, nous avons revu cette solution et choisi une autre implémentation de RTP qui est RTP_v2. Ainsi, nous avons proposé une autre solution plus simple. Cette solution fait l'objet de ce chapitre.

# NOTRE EXTENSION SUR NS-2

La figure IV.11présente notre propre implémentation sur NS-2. Nous avons décomposé l'implémentation selon nos objectifs en deux parties : la transmission vidéo via RTP/RTCP et la redirection DNS. Dans cette implémentation, nous utilisons (i) une version modifiée de RTP_v2[FRI, 09] permettant d'inclure les fonctionnalités supplémentaires caractérisant les clients et les serveurs réplica du CDN, et (ii) un nouvel agent DNS-CDN permettant de rediriger les requêtes du client au niveau du DNS vers les serveurs réplicas,

De plus, afin de rendre la simulation plus réaliste, nous générons dans notre architecture un fichier de trace du trafic vidéo. En effet, les modèles du trafic dans NS-2 (CBR, VBR, ...) ne permettent pas de représenter le modèle de trafic vidéo, il faut donc générer un fichier de trace du trafic vidéo compatible avec NS-2. Pour ce faire, nous avons utilisé le générateur de fichier de trace mpg2trace développé par [FRI, 09] qui converti un fichier vidéo en un fichier de trace de trafic compatible avec NS-2 contenant des informations utiles telles que le numéro de séquence, la taille et le temps d'émission pour chaque paquet RTP.

Nous tenons aussi à préciser que notre extension est assez simple par rapport à celle proposée par [ARO, 12]. Au fait, nous ne prenons pas plusieurs points en considération :

- La mesure de la QoE lors de la sélection du meilleur serveur réplica. Nous allons considérer la distance entre le client et le serveur réplica ainsi le critère du nombre de connexion qui représente la charge des serveurs réplicas.
- Le processus du remplacement du serveur réplica. Nous supposons que le client ne peut pas demander d'être réorienté vers un autre serveur réplica meilleur, il reste connecter avec le serveur réplica choisi jusqu'à la fin de la session.
- Le processus du midstream qui désigne la redirection du client vers un autre serveur réplica dans le cas de saturation du serveur réplica en cours.

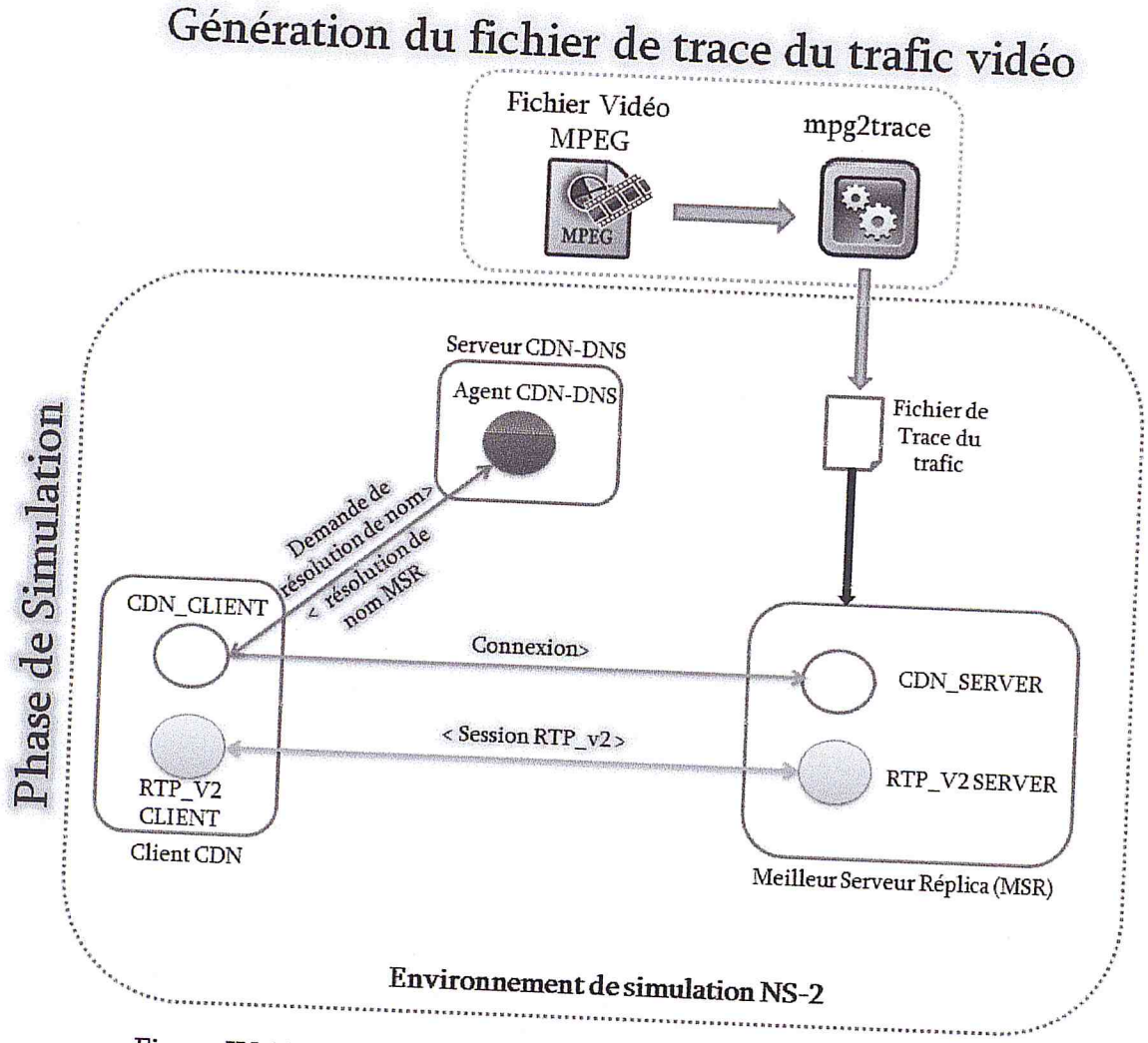



Figure IV.11. L'architecture de l'ensemble d'outils de simulation

## IV.1 Client CDN

C'est l'entité qui joue le rôle de client dans l'architecture de ce protocole, il contient un agent rtp_v2_client héritant de la classe rtcp, il contient aussi l'agent cdn_client qui hérite de la classe Agent, ce dernier envoie une demande au serveur DNS (CDN_DNS), cette demande est une requête qui contient le nom du contenu (REQ). Après avoir reçu une réponse du serveur DNS (RES) qui contient l'adresse du meilleur serveur choisi, le client CDN envoie à ce dernier une demande d'ouverture de session RTP_V2. Une session RTP_V2 va être établie et le client est servi.

## IV.2 Serveur CDN

C'est le meilleur serveur réplica dans l'architecture, il est choisi à chaque fois par le serveur CDN_DNS. Il contient le contenu désiré par les clients, dans notre cas il contient le fichier de trace de trafic généré précédemment. Il inclut l'agent cdn_server héritant de la

classe Agent, Après avoir reçu une demande d'ouverture de session RTP_V2 de la part du client CDN, il lance la session souhaitée entre son agent rtp_v2_server (qui hérite de la classe rtp) et le client CDN.

## IV.3 Serveur CDN_DNS

C'est l'entité plus importante dans toute l'architecture. Elle s'occupe de rediriger le client vers le meilleur serveur réplica contenant le contenu désiré en exécutant son algorithme de sélection de serveur. Dans notre cas on choisit le meilleur serveur selon sa charge et selon sa proximité. Elle se compose d'un agent cdn_dns héritant de la classe Agent, après avoir reçu une requête de la part du client CDN, le serveur CDN_DNS choisi le meilleur serveur et envoie une réponse sous forme de paquet contenant l'adresse du meilleur serveur réplica.

La figure IV.12 présente les principales interactions entre les entités du CDN à l'aide d'un diagramme de séquence.

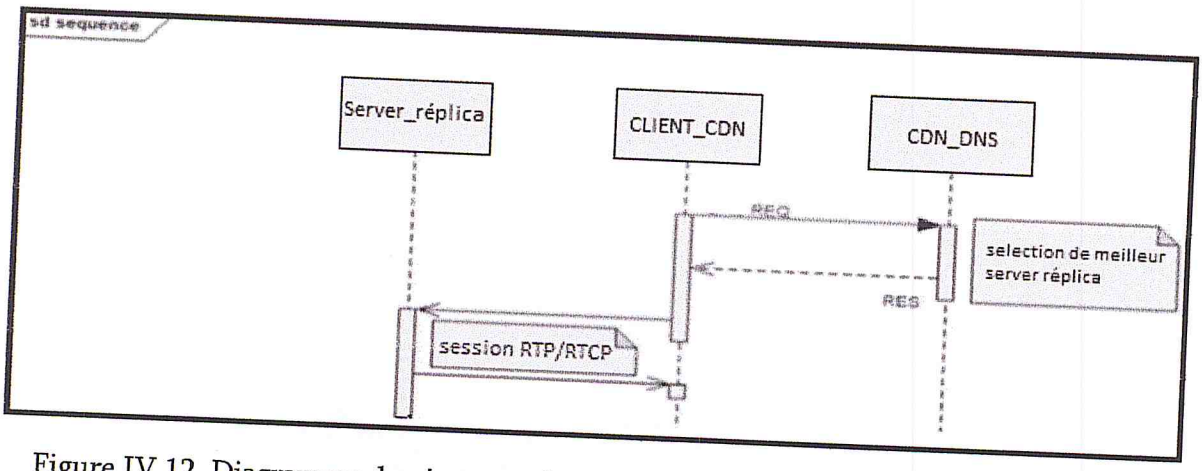



Figure IV.12. Diagramme de séquence des principales interactions des entités du CDN

# ETAPES D'IMPLEMENTATION

Dans notre travail, nous avons décomposé l'implémentation en plusieurs étapes afin de faciliter la compréhension du travail. Nous allons expliquer les détails de l'implémentation des différents agents et protocoles modifiés afin d'aboutir à notre extensions.

## V.1 Installation RTP_V2

Ce protocole est déjà vu dans la section I.1 de ce chapitre, il est déjà implémenté donc il nécessite seulement une installation. Il contient deux classes principales *rtp_v2AgentClass* et *rtcp_v2AgentClass*qui héritent de la classe *Agent,* la première est

responsable d'ouverture de session RTP et la deuxième responsable d'ouverture de session RTCP.Les principales fonctions de ce protocole sont :

```
start() ; stop() ; sendpkt() ; makepkt() ;
```

Dans le cas ordinaire, une session RTP/RTCP s'ouvre automatiquement entre un client et un serveur, mais dans notre travail la session est lancé après avoir reçu une demande d'ouverture de session de la part du client. Nous avons modifié sur ce protocole au niveau de la classe RTP_v2 pour pouvoir prendre l'adresse du client demandant l'ouverture de session. Il fallait ajouter deux fonctions pour cet objectif.

```
void RTP_v2Agent::sendpkt(u_int32_t srcid)
{
    Packet* p = allocpkt();
    lastpkttime_ = Scheduler::instance().clock();
    makepkt(p,srcid);
    target_->recv(p, (Handler*)0);
}
void RTP_v2Agent::makepkt(Packet* p,u_int32_t srcid)
{
    hdr_rtp_v2 *rtph2 = hdr_rtp_v2::access(p);
    rtph2->seqnoh() = seqno_++;
    rtph2->srcid() = session_ ? session_->srcid() : 0;
    //ouvrir une session avec le nouveau srcid
    rtph2->srcid() = srcid;
}
```

De la même manière pour RTCP, nous avons ajouté une variable externe globale `srcid` qui représente l'adresse du serveur et la modifié au niveau de la classe client.

```
extern const u_int32_t srcid_
```

Après avoir fait toutes ces modifications, le protocole RTP_V2 est prêt à être installé, après la décompression du fichier « rtp.tar.gz »[2], un dossier nommé « rtp_v2 » sera créé. On doit déplacer ce dernier vers le répertoire ou la source du simulateur est stockée i.e « ns-2.xx » (xx est la version de ns-2 installé, dans notre cas c'est ns-2.33). Ensuite, on doit

[2] Disponible sur http://gridnet.upc.es/%257Emaguilar/ns2_code_victor/ns2codevictor

modifier le fichier « Makefile.in » afin d'ajouter les fichiers sources de cette implémentation RTP :

- Ajouter à la fin de la section « `OBJ_CC` » (mais avant la ligne « `@V_STLOBJ@` ») :
  rtp v2/rtp v2.o \
  rtp v2/rtcp v2.o \
- Ajouter « `-I./rtp_v2` » à la section « `INCLUDES =` » au début de « Makefile.in ».
- Sauvegarder et quitter.
- Copier le fichier « session-rtp.cc » au dossier « ns-2.xx/common/ ».
- Copier les fichiers « rtp.cc » et « rtp.h » au dossier « ns-2.xx/apps/ ».
- Ajouter à la fin du fichier « ns-2.xx/tcl/lib/ns-default.tcl » :

```
Agent/RTP_v2 set seqno_ 0
Agent/RTP_v2 set interval_ 3.75ms
Agent/RTP_v2 set random_ 0
Agent/RTP_v2 set packetSize_ 210
Agent/RTP_v2 set maxpkts_ 0x10000000
Agent/RTP_v2 instproc done {} { }
Agent/RTP_v2 set rtcp_in_use_ 0
Agent/RTP_v2 set codification_ -1
Agent/RTP_v2 set lostcounter_ 0
Agent/RTP_v2 set losses_ 0
Agent/RTP_v2 set multipath_ 1
Agent/RTP_v2 set flow_id_ 0

Agent/RTCP_v2 set seqno_ 0
Agent/RTCP_v2 set interval_ 375ms
Agent/RTCP_v2 set random_ 0
Agent/RTCP_v2 set flow_id_ 0

Session/RTP set debug_ 0
```

- Ajouter le type de paquet « RTP_V2 » au fichier « common/packet.h », ceci est fait en ajoutant les lignes suivantes :
  - Dans la structure « `enum packet_t` » on ajoute les deux lignes suivantes avant la ligne « `PT_NTYPE // This MUST be the LAST one` » :

```
PT_RTP_v2,
PT_RTCP_v2,
```

  - Dans la classe « `p_info public : p_info()` » on ajoute les deux lignes suivantes mais avant « `name_[PT_NTYPE]= "undefined" ;` »

```
name_[PT_RTP_v2]="rtp_v2";
```

```
name_[PT_RTCP_v2]="rtcp_v2";
```

- Sauvegarder et quitter.
- Exécuter la commande « make ».

A la fin un nouveau fichier nommé « ns » sera disponible avec la nouvelle implémentation RTP/RTCP.

## V.2 Générateur duFichier de Trace

Cet outil génère un fichier de trace pour une vidéo codée MPEG2. NS-2 a besoin d'un fichier de trace pour une vidéo codée MPEG2, tel que ce fichier rassemble les informations nécessaires pour transmettre le flux vidéo. Après la décompression de fichier « mpeg2_tace_ns2.tar.gz »[3] on obtient le code source dans le dossier « src » et le fichier binaire exécutable « mpeg2_trace_ns2 » dans le dossier « bin ». Ceci est l'output du programme quand aucun fichier n'est spécifié, montrant les différents arguments :

```
Running program =./mpg2trace
Number of Arguments = 0
Arguments:

1) Mpeg Video Stream
2) Ouput Text Data file
3) Output Metadata file
4) Output Slice info file
```

Si on veut créer un fichier de trace pour une séquence vidéo on tape juste les commandes suivantes dans le terminal :

```
./mpg2trace video-mpeg2.m2v video-mpeg2.txt video-mpeg2.metadata
video-mpeg2.sli
```

Tel que « video-mpeg2.m2v » est le fichier d'entrée qui contient la séquence vidéo. Trois fichiers de sortie sont fournis : « video-mpeg2.txt » et « video-mpeg2.sli » qui sont générés par mpg2trace avec les informations de la structure de la vidéo, le fichier « video-mpeg2.metadata » est le fichier de trace qui sera lié à la configuration de l'application émetteur-récepteur MPEG2.

## V.3 CLIENT CDN

Comme nous avons vu dans la section I.2 de ce chapitre, il existe plusieurs solutions partielles pour la redirection DNS. Parmi ces solutions vues, nous avons opté pour la solution de Shen et Zhao [SHZ, 03]. Le problème que cette solution n'est pas implémentée

[3] Disponible sur http://gridnet.upc.es/%257Emaguilar/ns2_code_victor/ns2codevictor

et n'est pas complète, alors, il fallait la compléter et implémenter tous les agents, donc nous nous sommes inspirés de ce travail seulement. Nous avons commencé par l'agent *cdn_client*, il faut définir le type d'agent et le type de ces paquets.

- L'agent cdn_client hérite de la classe Agent définie par NS-2.

```
Class CDN_ClientAgent : public Agent {
....
....
```

- Les champs du paquet contenu dans l'agent sont :

```
int content_type ; //type de contenu
int pac_type ; //type de paquet
int dst_addr ; //adresse de destination
int src_addr ; //adresse source
int file_size ; //taille de fichier
```

- Afin de faire l'appel vers les classes de protocole RTP_V2 on a ajouté les lignes suivantes :

```
Const u_int32_t srcid_ ; //variable globale
RTCP_v2Agent rtcp ; //appel de la classe rtcp_v2
void send_rtcp(u_int32_t srcid) ; //fonction de l'envoi
```

- Le client envoi une requête au serveur CDN_DNS demandant un contenu vidéo

```
la fonction command(..) {
....
{
Packet* pkt = allocpkt(); //création d'un nouveau paquet
..
hdr->content_type=content_type; //type de contenu
hdr->ptype=1; //type de paquet du client est 1
...
send(pkt, 0); //l'envoi du paquet
```

- Lors de la réception d'une réponse de la part du serveur CDN_DNS, le client demande au serveur réplica choisi de lancer une session rtp_v2/rtcp_v2

```
la function recv(..){
```

```
    ....
    if(hdr->ptype==2)//si le paquet reçu vient du CDN_DNS
    {
    x=hdrip->daddr(); //adresse du serveur réplica
    ..
    senddest(..) ; //l'envoie du paquet au serveur réplica
    send_rtcp(x); //fonction de session rtp/rtcp
    }
    ....
    rtcp.sendpkt(); //fonction de l'envoi de paquets rtp_v2
}
```

## V.4 Serveur CDN (MSR)

C'est le meilleur serveur réplica, il contient la vidéo désirée par le client. Son rôle est de fournir la vidéo, après avoir reçu une demande du client il commence à transmettre la vidéo via le protocole RTP_v2/RTCP_v2.

- Il contient l'agent *cdn_server* héritant de la classe Agent.

```
Class CDN_Server : public Agent {
....
```

- Il contient les mêmes champs du paquet que celui du client.
- Afin de faire l'appel vers les classes de protocole RTP_V2 on a ajouté les lignes suivantes :

```
u_int32_t srcid_ ;
RTP_v2Agent rtp ; //appel de la classe rtp_v2
Void send_rtp(u_int32_t srcid) ; //fonction de l'envoie
```

- Après avoir reçu une demande d'ouverture de sesssion rtp_v2/rtcp_v2 le serveur envoi les paquets rtp_v2

```
la function recv(..)
{    u_int32_t x; //pour sauvegarder l'adresse
     ....
     senddest(..); //l'envoie de paquet
```

```
        x = hdrip->daddr(); //l'adresse du client
        send_rtp(x); //la session RTP_V2
    }
```

## V.5 Serveur CDN_DNS

C'est la partie la plus importante de l'architecture de notre travail, il contient un agent *cdn_dns* héritant de la classe Agent. Il est responsable de rediriger le client vers le meilleur serveur réplica. Il exécute un algorithme de sélection de serveur selon sa proximité et la charge de ce dernier. On aévité de travailleravec les listes chainées vu que si on dépasse les 100 maillons ça sera difficile de parcourir pour chercher, ajouter ou supprimer les maillons mais la table de hachage nous donne une bonne maitrise et rapidité d'extraction des données. Cette table de hachage inclut toutes les adresses des serveurs réplicas de la CDN et leurs charges.

- Le serveur CDN_DNS contient l'agent *cdn_dns* qui hérite de la classe *Agent*.

```
class CDN_DNS : public Agent {
....
```

- Les champs du paquet contenu dans l'agent sont

```
int content; //type de contenu
int pac_type ; //type de paquet
int dst_addr ; //adresse de destination
int src_addr ; //adresse source
```

- Il contient une fonction de hachage `createhashtable()` ;et fait l'appel à la classe `HashTable`.
- Nous avons utilisé une table de hachage **statique**pour maintenir les informations de routage comme suit :

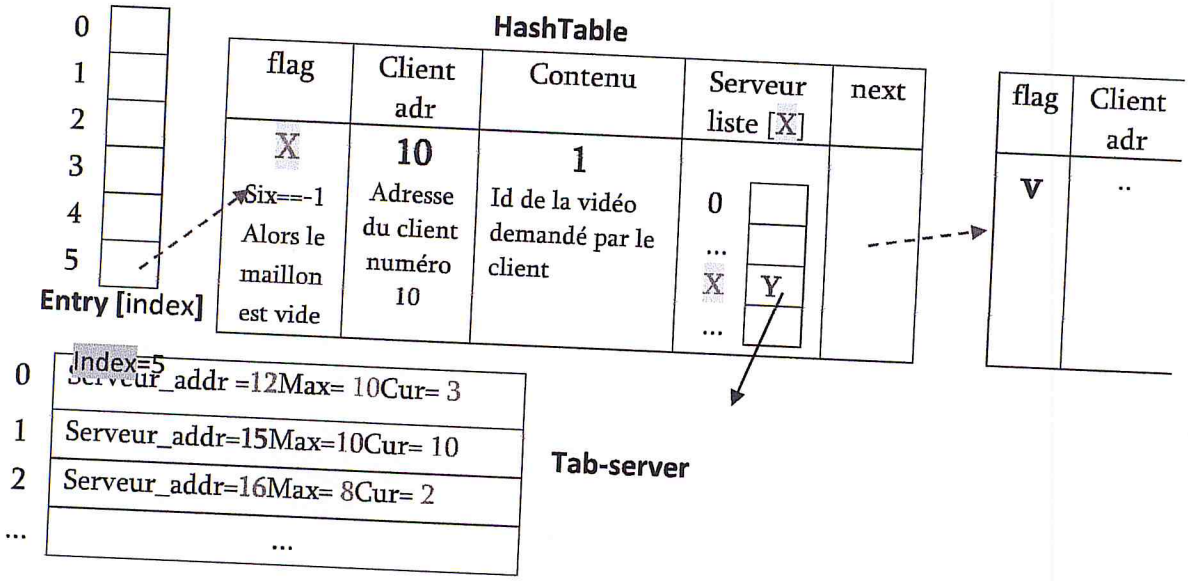



Tableau IV.1. La table de hachage et ses entrées

- La clé de cette table est calculée en se basant sur l'adresse source (client) et le type de contenu demandé, la fonction de hachage est comme suit :

**index = ((source – 10) * 10 + content) % length of table=**

Adresse source du client qui a fait la demande Ex : 10

Contenu-id Ex : 1

La langueur de la table par Ex : 8

Résultat de la clé de la table Ex : 5

- A la réception d'une requête de la part du client, le serveur CDN_DNS calcule la valeur de l'index pour avoir la clé de la table de hachage, et de cette manière une case du vecteur Entry [] pointe sur la liste chainé qui contient le maillon recherché, comme le montre la figure ci-dessus.
- Un maillon dans la table de hachage contient 5 cellules :
  - **Le flag :** c'est une variable type entier qui a une double utilisation :
    - Si elle est égale à (-1) elle nous informe que le maillon est vide et il ne contient aucune information, elle est utiliser dans la fonction de recherche pour passer au maillon suivant.
    - elle peut prendre n'importe quelle valeur (**x**) qui référence (indice) dans le tableau de Serveur-liste [**x**]

- **clientadr** :on garde le numéro de client dans le but de faire un classement des serveurs selon leur proximité, donc chaque client a une liste des serveurs les plus proche au plus loin.
- **contenu**:une variable qui simule l'identifiant de contenu que le client peut demander
- **server-liste** [] : unvecteur qui contient des références vers les adresses des serveurs classé selon leur proximité.du serveur le plus proche au serveur le plus loin par rapport au client.
    - le flag si sa valeur est différente à (-1) il va être utilisé comme indice dans le vecteur (server-liste [flag]) et de cette manière on obtient une référence (y)qui pointe sur d'autres informations sur ce serveur, comme la figure Au-dessus le montre.
    - ce pointeur(y) pointe sur un tableau nommé **Tab-server** [] ce dernier contient tous les informations sur le serveur (l'adresse du serveur, le nombre de connexion courante, le nombremaximal des connexions que le serveur peut supporter)
    - au final le serveur qui a été choisie selon son rapprochement du client va être aussi testé au niveau de sa charge de connexion, sa valeur des connexions courante doit être inférieure à sa valeur des connexions maximales supportées (Cur<Max).si c'est bon ce serveur va être retourné comme étant le meilleur serveur réplica (MSR)
    - après avoir retourné le MSR le nombre des connexions courante va être incrémenté et si cette valeur va être égale à la valeur des connexions maximal (Cur=Max) alors la valeur de flag va être incrémenté pour changer un autre serveur.
- **next** :un pointeur qui pointe sur le maillon suivant de la chaine.

## V.6 Compilation

Après la décompression du fichier « CDN_DNS_V1.tar.gz »,un nouveau dossier sera créé.Il contient tous les classes nécessaires pour incorporer ce module dans NS-2. Ensuite, on doit modifier le fichier « Makefile.in » comme suit :

- Ajouter à la fin de la section « `OBJ_CC` » (mais avant la ligne « `@V_STLOBJ@` ») :

```
MR_V1/cdn.o MR_V1/cdn_dns.o MR_V1/cdn_client.o \
MR_V1/cdn_server.o MR_V1/hash.o \
```

- Ajouter « `-I./MR_V1` » à la section « `INCLUDES =` » au début de « Makefile.in ».
- Sauvegarder et quitter.
- Ajouter à la fin du fichier « ns-2.xx/tcl/lib/ns-default.tcl » :

```
Agent/CDN_Client set content_type_ 0
Agent/CDN_Client set packetSize_ 1000
Agent/CDN_Client set file_size_ 0

Agent/CDN_DNS set packetSize_ 0

Agent/CDN_Server set packetSize_ 1000
```

- Ajouter le type de paquet « CDN_DNS » au fichier « common/packet.h », ceci est fait en ajoutant les lignes suivantes :
  - Dans la structure « `packet_t` » on ajoute les deux lignes suivantes avant la ligne « `PT_NTYPE // This MUST be the LAST one` » :

```
PT_CDN_DNS,
PT_CDN_Client,
PT_CDN_Server,
PT_CDN,
```

  - Dans la classe « `p_info public : p_info()` » on ajoute les deux lignes suivantes mais avant « `name_[PT_NTYPE]= "undefined" ;` »

```
name_[PT_CDN_DNS]="cdn_dns";
name_[PT_CDN_Client]="cdn_client";
name_[PT_CDN_Server]="cdn_server";
name_[PT_CDN]="cdn";
```

- Sauvegarder et quitter.
- Exécuter la commande « make ».

Pour faciliter le travail d'emplacement de fichiers, nous proposons une architecture de répertoire pour NS-2 afin d'intégrer l'implémentation sur le simulateur NS-2.

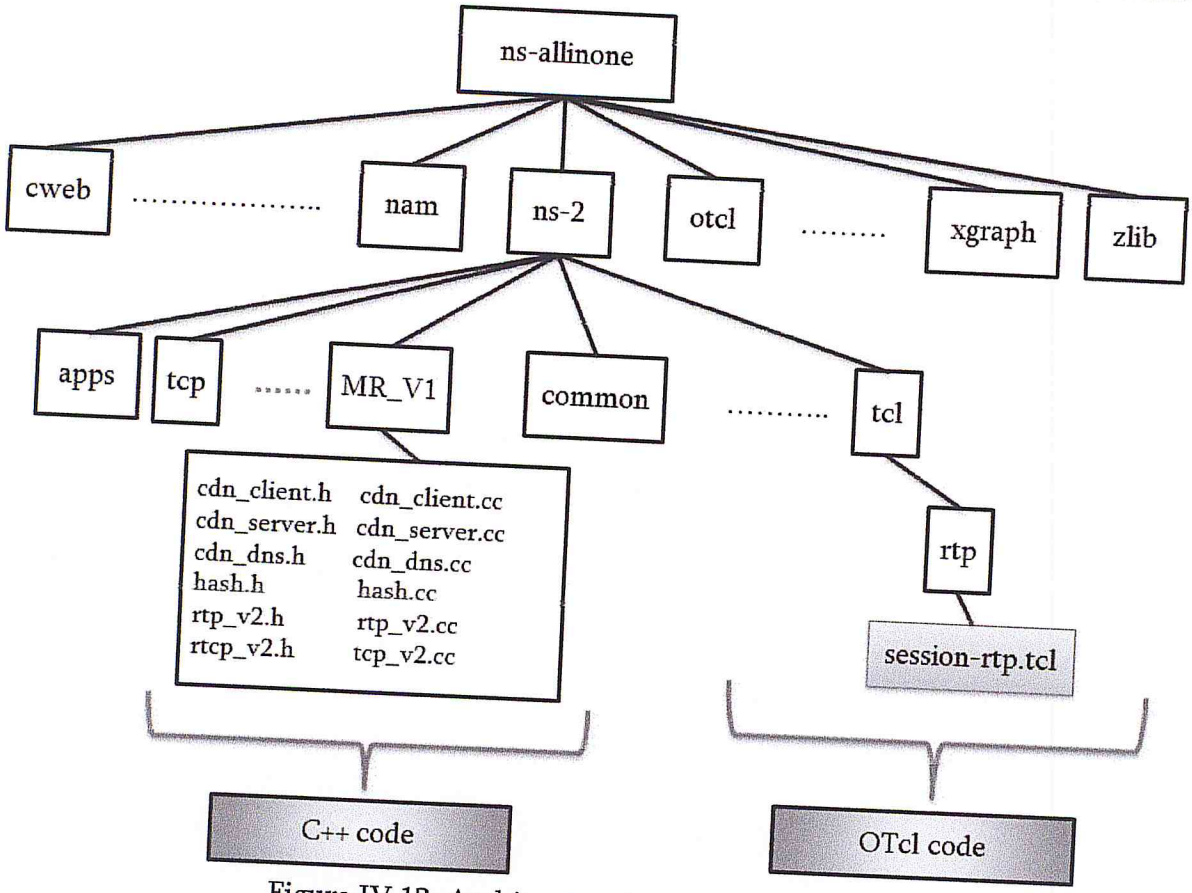



Figure IV.13. Architecture de répertoire NS-2

## SIMULATION

Afin de tester cette implémentation, nous avons fait des simulations avec le simulateur réseau NS-2. Sur ce dernier, on exécute le script OTCL, la simulation génère un fichier de sortie appelé fichier de trace. Nous avons créé un scénario de simulation afin de tester notre protocole proposé.

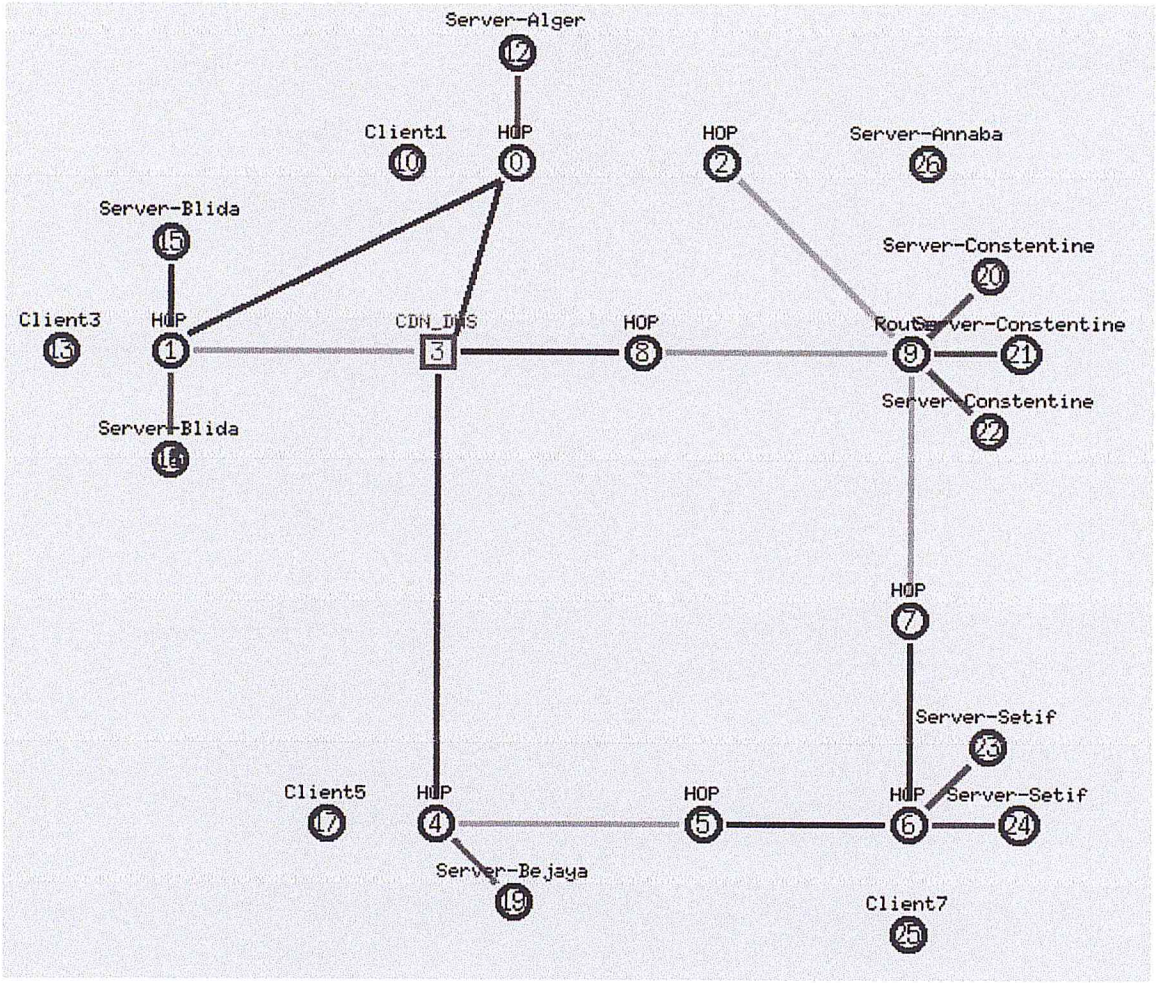



Figure IV.14. Topologie du scénario final

Dans la figure ci-dessus, nous pouvons voir la topologie de scénario final, nous remarquons des lignes de différentes couleurs, ces lignes indiquent les liens duplexes avec différentes bandes passantes et délais :

| Lien | Bande passante | Délai |
|---|---|---|
| Vert | 10Mbps | 70ms |
| Bleu | 6Mbps | 50ms |
| Jaune | 1.25Mbps | 20ms |

Tableau IV.2. Différents liens entre les nœuds

Chaque nœud connecté avec la ligne rouge a un agent *cdn_server* attaché à lui ; chaque nœud connecté à la ligne jaune a un *cdn_client* attaché à lui ; le nœud 3 a un agent *cdn_dns* attaché à lui ; Les nœuds 20, 21 et 22 ont des agents *cdn_server* attachés à eux et ils forment une ferme de serveur.

Le code source de la table de hachage du serveur DNS

```
createhashtable() // création de la table de hachage

{

//liste de serveur

int li01[] = {12,15,16,26,9,19,23,24};  liste des serveurs
pour les clients d'Alger de plus proche au plus loin

int li31[] = {15,16,12,9,19,26,23,24};  liste des serveurs
pour les clients de blida de plus proche au plus loin

.....

//création de table de hachage

hash.makeHashTableEntry(10,1,li01); création de table pour les
clients de Alger

hash.makeHashTableEntry(13,1,li31); création de table pour les
clients de Blida

.....

//creation de table de server

hash.makeServerTableEntry(12,10,3); //serveur Alger (12), max
(10) , curant(3)

hash.makeServerTableEntry(15,8,8);//serveur Blida(15),
max(8),curant(8)

hash.makeServerTableEntry(16,10,4);//serveur Blida(16),
max(10),curant(4)


....

}
```

Dans notre expérience, le client 1 et le client 3 vont demander une résolution du nom auprès le serveur DNS :

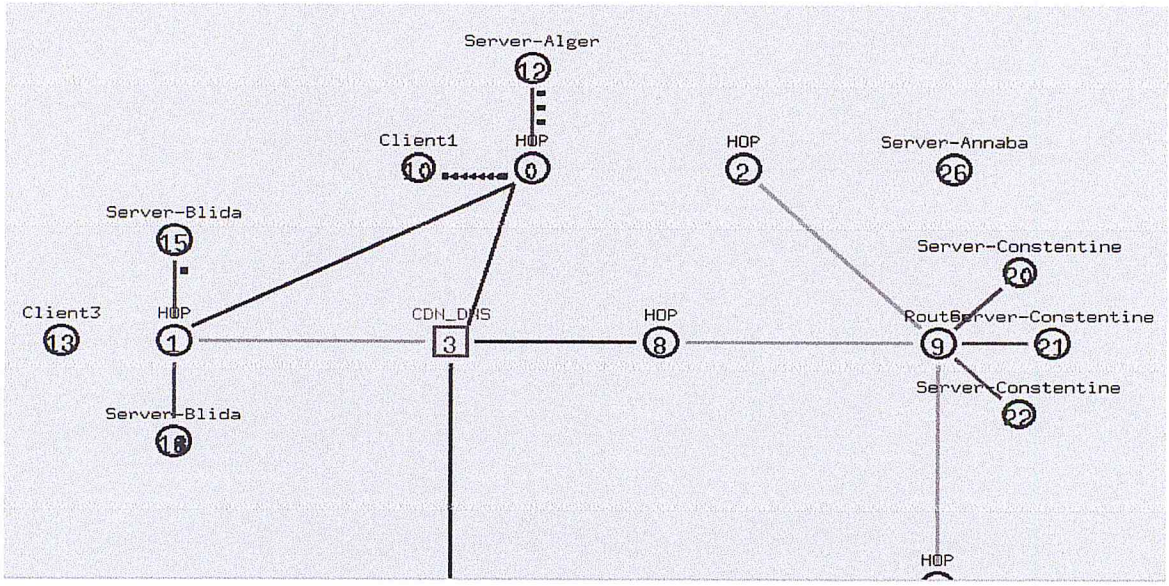
```
$ns at 0.1 "$ns trace-annotate

//"Client 1,3 envois une résolution du nom auprès de DNS\""

$ns at 0.1 "$c1 send"
```

```
$ns at 0.1 "$c3 send"
```

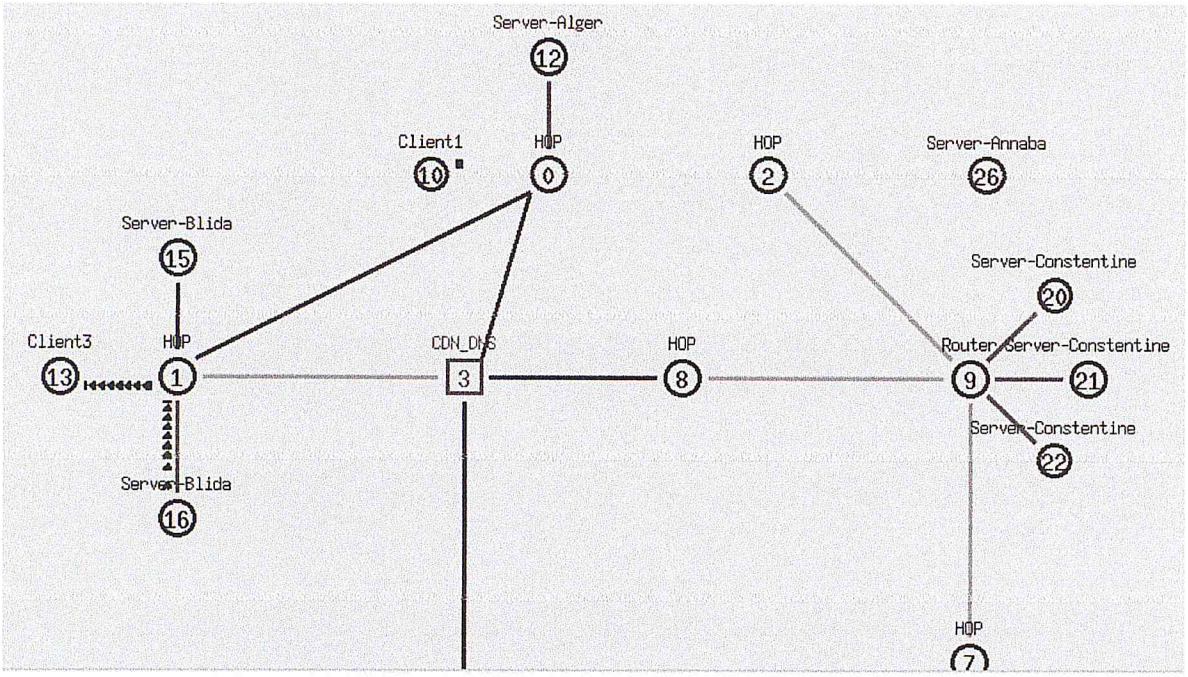
Nous avons eu le résultat suivant :




Client 1 servi

Le client 1 demande au serveur DNS de lui désigner le MSR, le DNS consulte sa table de hachage et cherche le meilleur serveur pour les clients d'Alger vus qu'il a déjà un classement des serveurs les plus proche dans une liste( li01[] ) et il vérifie en même temps la disponibilité de ces serveur selon leur charge (nombre de connexion),

Dans le premier cas, le client1 va avoir comme réponse le serveur 12 (Server-Alger) vu que il est le premier de la liste des serveurs les plus proche et il est disponible parce qu'il n'a pas atteint le nombre maximal de connexion (max (10) >curant (3))




Client 3 servi

Figure IV.15. Différents scénarios de simulation

Dans le deuxième cas, le client 3 va avoir comme réponse le serveur 16 au lieu de serveur15 qui est le premier dansle classement mais qu'il est chargé (max(8)=curant(8)) ,alors le DNS choisit le prochain sur la liste des serveurs les plus proche ( li31[] ) et le moins chargé et c'est le serveur 16 (max(10)>curant(4) )

## CONCLUSION

Dans ce chapitre nous avons vu notre propre solution qui combine à la fois celle de Shen et Zhao [SHZ, 03] (redirection DNS) avec celle de Victor Carrascal Frias [FRI, 09] (transmission vidéo).Nous avons vu la conception du cette solution et son implémentation à travers les trois agents créés : CDN_DNS, CDN_client et CDN_serveur.Le code source de ces agents est fourni ainsi qu'un manuel d'installation.

# CONCLUSION GENERALE& PERSPECTIVES

L'objectif de notre travail fut d'implémenter une extension sur le simulateur réseau NS-2 qui permet à la fois la redirection DNS et la transmission vidéo dans le CDN. Pour ce faire, il a été nécessaire, dans un premier temps, de faire une étude bibliographique sur la vidéo sur Internet (architectures et protocoles), les CDNs (notamment le SRR) et le simulateur réseau NS-2. Dans un second temps, nous nous présenté le protocole de méta-routage à implémenter et qui fait partie du [ARO, 12]. Ce protocole permet de rediriger en temps réel les requêtes de l'utilisateur dans la phase de résolution de noms (DNS) vers le meilleur serveur réplica. Pour le mettre en place, nous avons introduit une extension qui combine à la fois celle de Shen et Zhao [SHZ, 03] pour la redirection DNS avec celle de Victor Carrascal Frias [FRI, 09] pour la transmission vidéo. Nous avons ainsi créé trois nouveaux agents : CDN_DNS, CDN_client et CDN_serveur. Toute l'implémentation du protocole se trouve dans ce mémoire avec le test de simulation. Le code source des fichiers implémentés est fourni ainsi qu'un manuel d'installation.

Comme perspectives, cette native implémentation peut être complétée et/ou améliorée concernant les points suivants :

- Prendre en charge autres paramètres de sélection du meilleur serveur réplica comme : la disponibilité du serveur, le temps de réponse du serveur, les paramètres de QoS du réseau (bande passante, délai, taux de perte, etc) et enfin la notion de la QoE.
- Prendre en compte le mécanisme de redirection si le client demande une redirection vers autre meilleur serveur.
- Rajouter le mécanisme « midstream » qui désigne la redirection du client vers un autre serveur réplica dans le cas de saturation du serveur réplica en cours.
- Effectuer des tests plus élaborés
- Comparaison avec d'autres algorithmes de sélection du meilleur serveur réplica.

# ANNEXE : CODE SOURCE sur NS-2

Afin de mettre en œuvre notre extension sur NS-2 (version 2.34) nous avons développé plusieurs agents etclasses. Les noms de tous les fichiers implémentés sont inclus ici :

1. Les fichiers C++ pour implémenter les nouveaux agents :
   - cdn_client.h
   - cdn_client.cc
   - cdn_dns.h
   - cdn_dns.cc
   - cdn_server.h
   - cdn_server.cc
   - rtp_v2.cc
   - rtp_v2.h
   - rtcp_v2.cc
   - rtcp_v2.h
2. les fichiers C++ pour les fonctions de table de hachage
   - hashtable.h
   - hashtable.cc
3. le script de test
   - cdn.tcl

# REFERENCES BIBLIOGRAPHIQUES

## A

[AHA, 03] : T. Ahmed. Adaptive Packet Video Streaming Over IP Networks: A Cross Layer Approach. Thèse de doctorat, Université de Versailles-Saint-Quentin-en-Yvelines,Versailles, France, 2003, p. 191.

[AHA, 08] : T. Ahmed. Transport Adaptatif et Contrôle de la Qualité des Services Vidéo sur les Réseaux IP Filaires, Sans-fil et sur les Architectures P2P. Thèse de HDR, Université de BORDEAUX I, Talence, France, 2008, p. 151.

[ARO, 12] : S. Aroussi. Méta-Routage basé sur la Qualité d'Expérience dansle cadre des Applications Vidéo. Mémoire de Magistère. Ecole nationale Supérieure d'Informatique (E.S.I)Oued-Smar, Alger, 2012, p. 151.

[ATW, 02] : J.G. Apostolopoulos, W. Tan et S.J. Wee. Video Streaming: Concepts, Algorithms, and Systems, HP Labs Technical Report, 2002, p. 35.

[AYH, 96] : D. Andersen, T. Yang, V. Holmedahl, and O. Ibarra. Sweb: Toward a scalable world wide web server on multicomputers. In Proceedings of International Parallel Processing Symposium, April 1996.

## B

[BDL, 02] : T. Bu, N. Duffield, F. Lo Presti, and D. Towsley. Network tomography on general topologies. In Proc of ACM SIGMETRICS, Marina del Rey, California, June 2002.

[BGK, 08] : C. Bouras, A. Gkamas & G. Kioumourtzis. Extending the Functionality of RTP RTCP Implementation in Network Simulator (NS-2) to support TCP friendly, International ICST Conference on Simulation Tools and Techniques (SIMUTools), Marseille, France, 2008, p. 6.

[BHM, 11] : S. K. Bhardwaj & J.S. Malhotra. Simulation and Comparison of Hashing Techniques in CDN DNS, International Journal of Engineering Science and Technology, Vol. 3, Issue. 4, 2011, pp. 3039-3044.

[BLP, 03] : N. Bartolini, F. Lo Presti, and C. Petrioli. Optimal dynamic replica placement in content delivery networks. In Proc of the 11th IEEE Int. Conference on Networks (ICON), Sydney, Australia, Sept. 2003.

[BMV, 10] : F. Boronat, M. Montagud & V. Vidal. Enhanced RTP-based Tool-Set for Video Streaming Simulation using NS-2, Short Papers, International Conference on Mobile Computing and Multimedia Proceedings, Paris, France, 2010, pp. 382-386.

[BSC, 12] : Benjamin Rainer, Stefan Lederer, Christopher Müller, and Christian Timmerer. A Seamless Web Integration of Adaptive HTTP Streaming. Boucharest - Romania : s.n., aout 2012.

[BTM, 08] : Ali C. Begen, Tankut Akgul, and Mark Baugher. Watching video over the web . San Francisco – USA, 2008.

## C

[CCD, 95] : M. E. Crovella and R. L. Carter. Dynamic server selection in the internet. In Proceedings of the Third IEEE Workshop on the Architecture and implementation of High Performance Communication Subsystems, HPCS, 1995.

[CCD, 14] : Cisco. Cisco's distributed director. http://www.cisco.com/warp/public/cc/pd/cxsr/dd/index.shtml.

[CCY, 01] : V. Cardellini, M. Colajanni, and P. Yu. Redirection algorithms for load sharing in distributed web server systems. In Proc. IEEE 19th Int'l Conf. on Distributed Computing Systems (ICDCS), 1999.

[CDN, 96] : A. Chankhunthod, P. B. Dansig, C. Neerdaels, M. F. Schwartz, and K. J. Worrel. A hierarchical internet object cache. In *Proceedings of USENIX*, January 1996.

[CFO, 10] : F. Cece, V. Formicola, F. Oliviero & S. P. Romano. An Extended ns-2 for Validation of Load Balancing Algorithms in Content Delivery Networks, International ICST Conference on Simulation Tools and Techniques (SIMUTools), Malaga, Spain, 2010, p. 6.

[CHN, 02] : M. Coates, A. O. Hero, R. Novak, and B. Yu. Internet tomography. IEEE Signal Processing Magazine, May 2002.

[CKS, 01] : I. Cidon, S. Kutten, and R. Soffer. Optimal allocation of electronic content. In Proceedings of IEEE Infocom, 2001.

[CLH, 01] : K. S. Candan, W.-S. Li, Q. Luo, W.-P. Hsiung, and D. Agrawal. Enabling dynamic content caching for database-driven web sites. In *Proc. of ACM/SIGMODConference 2001, Santa Barbare, California, USA*, May 2001.

## D

[DKM, 96] : D. M. Dias, W. Kish, R. Mukherhee, and R. Tewari. A scalable and high available web server. In *Proceedings of the 41st IEEE Computer Society International Conference*, 1996.

[DMV, 05] : Djordje Mitrovic, Video Compression, University of Edinburgh, 2005.

[DOK, 01] : F. Douglis and M.F. Kaashoek, "Scalable Internet Services," *IEEE Internet Computing*, vol. 5, no. 4, 2001, pp. 36–37.

[DON, 08] : Y. Dony. Video-on-Demand over Internet: A Survey of Existing Systems and Solutions, Master Thesis, Universite Notre-Dame de la Paix, Namur, Belgique, p. 60, 2008.

[DRJ, 00] : S. G. Dykes, K. A. Robbins, and C. L. Jeffery. An empirical evaluation of client-side server selection algorithm. In Proceedings of IEEE Infocom, 2000.

## E

[EIC, 00] : EICE. Internet qos measurement for the server selection. Technical report, Technical Report of EICE, CQ2000-48, 2000.

## F

[FPV, 06] : Future Performance of Video Codecs. University of Essex ,United Kingdom : s.n., 13 juillet 2006.

[FRI, 09] : V. C. Frias. Contribution to provide QoS over Mobile Ad Hoc Networks for Video-Streaming Services based on Adaptive Cross-Layer Architecture, Doctor Thesis, Technical University Of Catalonia, Catalonia, Spain, 2009, p. 311.

## G

[GCZ, 06] : L. Guo, S. Chen et X. Zhang. Design and Evaluation of a Scalable and Reliable P2P Assisted Proxy for on-Demand Streaming Media Delivery, IEEE Transactions on Knowledge and Data Engineering, Vol. 18, Issue. 5, 2006. pp. 669-682.

[GSK, 08] : Y. Guo, K. Suh, J. Kurose et D. Towsley. A Directory-based Peer-to-Peer Video, Journal of Computer Communications (COMCOM), Vol. 31, Issue 3, 2008, pp. 520–536.

## H

[HGK, 98] : G. Hunt, G. Goldzmidt, R.P.King, and R. Mukherjee. Network dispatcher: A connection router for scalable internet services. In *Proceedings of 7th International World Wide Web Conference*, April 1998.

[HSR, 99] : M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. "SIP: Session Initiation Protocol," Internet Engineering Task Force, RFC 2543, Mar. 1999.

## J

[JCF, 07] : Jean-Charles Fouché, *Comprendre la vidéo numérique*, Éditions Baie des Anges, 2007.

[JJK, 01] : S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt. Constrained mirror placementon the internet. In INFOCOM, pages 31–40, 2001.

[JPG, 96] : Jean-Paul Guillois, Techniques de compression des images, Éditions Hermes, 1996

## K

[KDK, 03] : K. Dasgupta and K. Kalpakis. Maintaining replicated redirection services in webbased information systems. In Proceedings of the 2nd IEEE Workshop on Internet Applications, 2001.

[KKM, 02] : M. Karlsson, C. Karamanolis, and M. Mahalingam. A unified framework for evaluating replica placement algorithms. *Technical Report HPL-2002, Hewlett PackardLaboratories.*

[KRR, 02] : J. Kangasharju, J. Roberts, and K. W. Ross. Object replication strategies in content distribution networks. Computer Communications, 25(4):367–383, March 2002.

[KRS, 00] : P. Krishan, D. Raz, and Y. Shavitt. The cache location problem. IEEE/ACM Transactions on Networking, October 2000.

## M

[MNR, 98] : S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson. Adaptive web caching: Towards a new caching architecture. Computer Network andISDN Systems, November 1998.

[MTT, 01] : K. Mase, A. Tsuno, Y. Toyama, and N. Karasawa. A web server selection algorithm using qos measurement. In Proceedings of International Conference on Communication, 2001.

## N

[NSE, 13] : Niklas Pollard & Sven Nordenstam, Ericsson to buy Microsoft IPTV business, Reuters, [En ligne] 2013. http://www.reuters.com/article/2013/04/08/us-ericsson-microsoft-idUSBRE9370HL20130408.

## P

[PAV, 06] : G. Pallis et A. Vakali. Insight and Perspectives for Content Delivery Networks, Communications of the ACM, Vol. 49, Issue 1, 2006, pp. 101-106.

[PBV, 08] : M. Pathan, R. Buyya et A. Vakali. Content Delivery Networks: State of the Art, Insights, and Imperatives, Chapitre 1 du livre, Content Delivery Networks, R. Buyya, M. Pathan et A. Vakali, 2008, pp. 3-32.

[PEN, 08] : G. Peng. CDN: Content Distribution Network, Technical Report TR-125 of Experimental Computer Systems Lab in Stony Brook University, 2008, p. 26.

[PGA, 06] : Phillipe Gasser, Télévision et vidéo sur IP, MSH Paris nord – Plate-forme Arts, Sciences, Technologies, Paris – France, p. 64, 2006.

[PGM, 06] : T. Plagemann, V. Goebel, A. Mauthe, L. Mathy, T. Turletti et G. Urvoy-Keller. From Content Distribution Networks to Content Networks Issues and Challenges, Computer Communications, Vol. 29, No. 5, 2006, pp. 551-562.

[PHB, 06] : Philippe Bellaïche, Les Secrets de l'image vidéo, Paris, Eyrolle, 6e édition, 2006.

[PMM, 93] : C. Partridge, T. Mendez et W. Milliken. Host Anycasting Service, Internet Engineering Task Force, RFC 1546, 1993, p.9.

[POS, 80] : J. Postel. User Datagram Protocol, Information Sciences Institute, RFC 768, 1980, p.9.

[PYD, 98] : P. Yu and D. M. Dias. Analysis of task assignment policies in scalable distributed web-server systems. IEEE Transaction on Parallel and Distributed Systems, June 1998.

## S

[SBS, 98] : M. Sayal, Y. Breithart, P. Scheuermann, and R. Vingralek. Selection algorithms for replicated web servers. 26(3), December 1998.

[SCF, 03] : H. Schulzrinne, S. Casner, R. Frederick et V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, Internet Engineering Task Force, RFC 3550, 2003, p. 89.

[SHZ, 03] : S. W. Shen & S. X. Zhao. Simulation and Analysis of Content Delivery Network, Project Final Report, Special Topics: High Performance Networks, CMPT 885, Simon Fraser University, Colombie-Britannique , Canada, 2003, p. 26.

[SKS, 01] : M. Stemm, R. Katz, and S. Seshan. A network measurement architecture for adaptive applications. In Proceedings of IEEE Infocom, 2001.

[SRL, 98] : H. Schulzrinne, A. Rao et R. Lanphier. Real Time Streaming Protocol (RTSP), Internet Engineering Task Force, RFC 2326, 1998. p. 93.

[SRT, 99] : S. Sen, J. Rexford, and D. Towsley. Proxy prefix caching for multimedia streams. In Proceedings of IEEE Infocom, 1999.

## T

[TMP, 13]: H. TRAN, A. MELLOUK, J. PEREZ, S. HOCEINI, S. ZEADALLY. QoE-based Server Selection for Content Distribution Networks. University of Paris-Est Creteil Val de Marne (UPEC), France, 2013, p. 14.

## U

[UPD, 13] : Généralités sur la lecture en continu. Université Paris Descartes [En ligne] 2013. http://wiki.univ-paris5.fr/wiki/Streaming.

## V

[VGL, 06] : L. Vu, I. Gupta, J. Liang et K. Nahrstedt. Mapping the PPLive Network: Studying the Impacts of Media Streaming on P2P Overlays, Technical Report, University of Illinois at Urbana-Champaign, 2006, p. 9.

[VNI, 13] : cisco visual network index. Cisco VNI :Forcast and Methodology , 2012- 2017. USA : s.n., 29 mai 2013.

## W

[WBM, 13] : WebM. WebM An Open Web Media Project. [En ligne] 2013. http://www.webmproject.org.

[WHY, 01] : Dapeng Wu, Yiwei Thomas Hou ,Wenwu Zhu ,Ya-Qin Zhang. Streaming Video over the Internet: Approaches and Directions. fevrier 2001.

[WNC, 04] : Winddance networks corporation. http://www.winddancenet.com.

[WSA, 02] : B. Wang, S. Sen, M. Adler, and D. Towsley. Optimal proxy cache allocation for efficient streaming media distribution. In Proceedings of Infocom, 2002.

## X

[XKR, 06] : D. Xu, S.S. Kulkarni, C. Rosenberg et H.K. Chai. D. Xu et al. Analysis of a CDN-P2P Hybrid Architecture for Cost-Effective Streaming Media Distribution, Multimedia System, Vol. 11, Issue. 4, 2006. pp. 383-399.

[XLK, 07] : S. Xie, B. Li, G.Y Keung et X. Zhang. Coolstreaming: Design, Theory, and Practice. IEEE Transactions On Multimedia, Vol. 9, Issue 8, 2007, pp. 1661-1671.

## Z

[ZHJ, 08] : R. Zhou & K. Jang. Adaptive MPEG-4 Video Streaming Over IP Networks, International Technical Conference on Circuits/Systems, Computers and Communications (ITCCSCC), 2008, pp. 637-640.