REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE SAAD DAHLAB BLIDA

Département INFORMATIQUE

Mémoire de Projet de Fin d'Etudes

de Master en Informatique

Option : Génie Logiciel

Thème :

Streaming Adaptatif De Contenus Multimédia

Proposé par :

- **Mr. Amine BOUABDELLAH**

Encadreur :

- **Mr. Amine BOUABDELLAH**

Promotrice :

- **Mme. Nachida REZOUG**

Présenté par :

- **Ali SENANI**
- **Yassine YAHIA**

Promotion : 2012 - 2013

SOMMAIRE

Remerciements iii

Sommaire iv

Liste des figures viii

Liste des tableaux x

Introduction générale 1

Chapitre I. La Vidéo 4

I.1 Introduction 4

I.2 Vidéo 5

I.3 Codecs 6

I.3.1 H264 7

I.3.2 VP8 7

I.4 Les Conteneurs 7

I.4.1 MP4 7

I.4.2 WebM 8

I.5 Mesure de la qualité vidéo 8

I.6 La transmission de la vidéo sur l'internet 9

I.7 Conclusion 9

Chapitre II. Le streaming 10

II.1 Introduction 10

II.2 Streaming 10

II.3 Les Différentes types de streaming 11

II.3.1 Streaming Server (Serveur de diffusion en continu) 11

II.3.2 Streaming Progressif 12

II.3.3 Streaming Adaptatif 13

II.3.4 La norme DASH 14

II.3.5 Solutions propriétaires 17

II.4 Pourquoi HTTP dans streaming adaptatif 20

II.5 Algorithme proposée Par Konstantin Miller et Adam wolisz 22

II.5.1 Contexte 22

II.5.2 Objectifs d'adaptation 22

II.5.3 Description de l'algorithme (25) 23

II.6 Algorithme présent dans la plateforme ITEC 27

II.6.1 Remplissage de buffer 27

II.6.2 Choix de la représentation Vidéo 28

II.7 Conclusion 29

Chapitre III. Contribution Et Environnement d'expérimentation 30

III.1 Introduction 30

III.2 Plateforme réelle 31

III.2.1 DASH 32

III.2.2 Le contenu : Vidéo 32

III.2.3 Réseaux de transport : Internet 33

III.3 Plateforme d'expérimentation 36

III.3.1 DASH : DASH-JS 36

III.3.2 Le contenu : Vidéo 39

III.3.3 Streaming : Modèle Client /Serveur 40

III.3.4 Réseau : Emulation 41

III.4 Mise en place de la plateforme d'expérimentation: 43

III.4.1 Vidéo 43

III.4.2 Émulation de réseaux 49

III.4.3 Serveur NTP 54

III.5 Conclusion 53

Chapitre IV. Evaluation 54

IV.1 Introduction 54

IV.2 Evaluation du streaming adaptatifs sous différentes contraintes 55

IV.2.1 Scénario 1: Scénario de référence (Original sans modification) 55

IV.2.2 Scénario 2 : retrait du coefficient 0 .9 58

IV.2.3 Scénario 3 : coefficient α β intervertis 60

IV.2.4 Scénario 4 : coefficient α β intervertis et retrait du coefficient 0.9 62

IV.3 Comparaison entre les résultats obtenus 64

IV.4 Analyse de tableau 65

IV.5 Conclusion 66

Conclusion générale 66

Bibliographie 68

Annexe : Outils de développements 70

IV.5.1 **Environnement de programmation**.....70
IV.5.2 Système d'exploitation Ubuntu.....70
IV.5.3 **Émulation de Réseaux**.....72

LISTE DES FIGURES

Figure I-1 Le trafic IP mondial selon le type de périphérique (1) 5

Figure II-1 Un diagramme du processus complet de streaming (10) 11

Figure II-2 Structure Simplifié d'un MPD (14) 15

Figure II-3 Example d'un MPD (14) 15

Figure II-4 Exemple d'adaptation du client en MPEG-DASH 17

Figure II-5 L'architecture de HTTP Live Streaming (17) 19

II-6 La pile de protocole 21

II-7 Le principe de cashe 21

Figure IV-2 Algorithme d'adaptation (25) 25

III-1 L'architecture de la plate forme réel 32

Figure III-2 Foncionnalités de Dummynet-NIStNet , et TC/Netem (27) 35

III-3 Plateforme d'expérimentation 36

Figure III-4 L'architecture de DASH-JS (11) 37

Figure III-5 Activation de Media Source API 39

Figure III-6 Structure géneral de EvalVid 40

III-7 Client-Serveur 41

Figure III-8 Expérimentation en environnement réel ou émulé 42

Figure III-9 Mise en place de la plate forme d'expérimentation 43

Figure III-10 Commande de conversion de format y4m vers yuv 44

Figure III-11 Conversion Vidéo de format yuv vers webM 44

Figure III-12 La commande qui fait la conversion de format yuv vers WebM 45

Figure III-13 commande de creationde fichier MPD 47

Figure III-14 création de fichier MPD pour chaque qualité 47

Figure III-15 la commande de création d'un fichier MPD 47

Figure III-16 fichier MPD pour toute les qualités d'une Vidéo 48

Figure III-17 Exemple du nouvelle commande 49

Figure III-18 Test de Réseaux avant la commande TC 49

Figure III-19 commande Tc pour simuler un délai de 20ms sur tout les paquets sortant 50

Figure III-20 resultat de test aprés l'execution de la commande Tc 50

Figure III-21 Commande Iperf pour tester la capacité de notre réseau LAN 51

Figure III-22 Commande Tc pour simuler un trafic sortant a 521 kbps 51

Figure III-23 Résultat de test 51

Figure III-24 Commande Tc pour revenir a la configuration initiale 53

Figure III-25 Script python (coté serveur) 54

Figure III-26 Commande d'installation de NTP 55

Figure III-27 Script python avec l'implémentation de serveur NTP 56

Figure III-28 Script de lancement de client 57

Figure III-29 Représentation de notre réseaux local 57

Figure IV-1 Principe du Streaming adaptatif 54

Figure IV-3 Graphes de représentation d'une vidéo 56

Figure IV-4 Graphe de représentation des débits entre client et le serveur 57

Figure IV-5 Graphes de représentation d'une vidéo 59

Figure IV-6 Graphe représentation d'une vidéo 60

Figure IV-7 Graphe représentation d'une vidéo 61

Figure IV-8 Graphe de représentation des débits entre client et serveur 62

Figure IV-9 Graphe représentation d'une vidéo 63

Figure IV-10 Graphe de représentation des débits entre client et serveur 64

Figure IV-12 La commande pour la fréquence d'interruption 79

LISTE DES TABLEAUX

Tableau 1 Comparaison entre Microsoft-LSS, Apple HLS, et MPEG-DASH.....20

Tableau 2 tableau descriptif de la commande ffmpeg.....45

Tableau 3 description de la nouvelle commande.....48

Tableau 4 Tableau des resultats des tests.....53

Tableau 5 une comparaison entre les différents tests.....65

INTRODUCTION GENERALE

Les services multimédias connaissent actuellement un succès remarquable et sont de plus en plus déployés C'est notamment le cas des services de vidéos. Le trafic Internet d'aujourd'hui est généré en majorité par des applications du streaming vidéo, sur des terminaux fixe ou mobile car la diffusion de vidéos est monnaie courante. Dans le domaine de la vidéo en ligne, l'une des principales questions ces dernières années fût de migrer des technologies classiques de streaming vers du téléchargement via HTTP

les problème principaux de streaming vidéo sur les réseaux IP sont , le manque de fiabilité de ce réseaux (Réseau du type Best Effort), Particulièrement, les fluctuations de débit dans les réseaux IP posent un problème majeur dans la maximisation de la qualité de service observée par l'utilisateur finale et la fluidité n'est pas garantie (longs temps d'attentes, interruptions, blocage partiel ou total) , il y'a aussi le problèmes de compatibilité des navigateurs et des players . Dans ce cadre, une solution récente est le Streaming adaptatif de la vidéo. Cette solution se base sur la génération, au niveau du serveur, de plusieurs descriptions pour une seule vidéo. Les description sont codées a différents débits ou chaque débit correspond à une qualité donnée .

Le challenge est de proposer une solution simple et efficace pour les fournisseurs de services, en garantissant une bonne qualité d'expérience pour les clients, quel que soit le type de terminal utilisé. Une solution prometteuse, « Streaming Dynamique et Adaptatif sur HTTP » (1) (DASH en anglais), consiste à segmenter des différents débits de la même vidéo (ces morceaux de vidéos sont appelées des chunks). On liste ces différents segments dans différents fichiers permettant l'ajustement d'un débit à l'autre par le client . Le lecteur de ce dernier teste ainsi régulièrement l'état du débit et télécharge le meilleur segment suivant dans sa mémoire tampon .

L'introduction de ces technologies a permis d'un coté de réduire le cout en bande passante pour la diffusion et la mise en cache de contenu et de l'autre d'optimiser l'expérience utilisateur quel que soit son débit .En revanche , les couts ont augmentés d'une part en terme de temps de préparation de contenus (notamment en raison des multiples transcodages) ; d'autre part , en terme de stockage de ces mêmes contenus (au lieu d'un seul fichier auparavant, on se retrouve maintenant avec un nombre important de petits fichiers) .

Avant de pouvoir nous focaliser sur l'étude de l'algorithme d'adaptation, le premier obstacle que nous avons rencontré était la mise en place d'une plate-forme d'expérimentation adéquate. La transmission vidéo sur les réseaux de nouvelles générations requiert des compétences diverses et complémentaires sur toute la chaîne de transmission. Ainsi il a été nécessaire de rassembler, d'adapter et d'intégrer un ensemble d'outils hétéroclite afin de structuré notre réseau. Il a fallu mener plusieurs taches dont celle d'assurer la compatibilité entre les composants du réseau (spécification des formats vidéo, choix du protocole de streaming, configuration des différents paramètres) ; développer une application permettant la perturbation du trafic sur le réseau en limitant la bande passante disponible pour les clients (émulation réseau) ; implémenter les scripts de l'adaptation et finalement recueillir les résultats et leur appliquer un traitement .

L'objectif principal à atteindre est l'amélioration de la qualité d'expérience de l'utilisateur final Cette amélioration se fera par le développement d'un algorithme d'adaptation. Les input pouvant avoir une influence sur la qualité vidéo sont: la taille des chunks vidéo, le nombre de descriptions vidéos, longueur du buffeur de lecture, changement de la bande passante et l'ensemble des conditions nécessaire avant d'initier le saut vers une description vidéo différent tout cela pour assurer un niveau de qualité aussi bon que possible pour le client autrement dit améliorer la qualité de service .

Pour réaliser ce projet de fin d'étude qui nous a été proposé au sein de l'équipe telecome de la division Architecture des Systèmes et Multimédia (ASM) au centre de développement des Technologies Avancées (CDTA), dont le thème est le suivant : « Algorithme d'adaptation pour le Streaming adaptatifs de contenus multimédia sur HTTP » , nous avons organisé notre mémoire en quatre chapitres .

Chapitre I. LA VIDEO

I.1 Introduction

Le trafic vidéo sur Internet va représenter 69% de tout le trafic Internet grand public en 2017 (contre 57% en 2012). Ce pourcentage n'inclut pas la vidéo par échange peer-to-peer (P2P) de partage de fichiers. La somme de toutes les formes de vidéo (TV, vidéo à la demande [VoD], Internet et P2P) sera de l'ordre de 80 à 90 % du trafic mondial de consommation d'ici 2017 (1)

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 (1).

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. la Figure I-1 représente le trafic IP mondial selon le type de périphérique

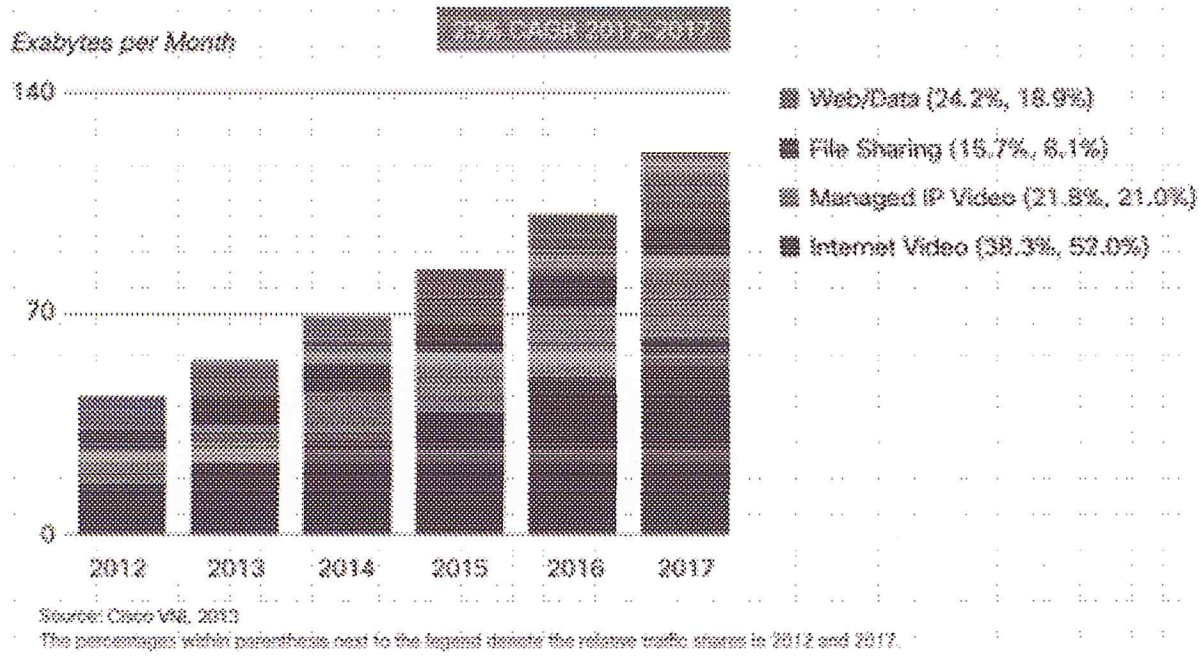


Figure I-1 Le trafic IP mondial selon le type de périphérique (1)

Dans ce chapitre, nous donnons un aperçu sur la vidéo et les différents codecs (codage/décodage) disponibles et comment on mesure la vidéo. Nous présentons ensuite quels sont les différents mode de diffusions de la vidéos qui existent .

I.2 Vidéo

La vidéo regroupe l'ensemble des techniques, technologie, permettant l'enregistrement ainsi que la restitution d'images animées, accompagnées ou non de son. Le mot vidéo vient du latin *vidéo* qui signifie " je vois ". Un flux vidéo est une succession d'images, 25 par seconde en Europe (30 par seconde aux USA) (3),jouée successivement pour créer l'illusion du mouvement . Chaque image est composée de lignes horizontales .Chaque ligne pouvant être considérée comme 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 . La vidéo doit être codé afin d'optimiser son stockage ou sa transmission (2) .

Pour bien comprendre pourquoi on a besoin de compresser les données vidéo, nous allons prendre un moniteur dont la résolution maximum est de 1024x768 ce qui est un standard de nos jours, donc un tel écran représente une matrice de pixels de :

1024x768=786432 pixels

Les cartes graphique supportent aisément le mode couleurs vraies qui possèdent un format de 32bits soit 4 octets par pixel. Donc pour afficher une image en pleine écran avec ce mode il faudra :

$$786432x4 = 3145728 \text{ octets soit } \mathbf{3072\ Ko}$$

Si je voulais regarder une seconde d'animation, je devrais associer 25 images au même format.

Donc pour une seconde de film on obtient :

$$3072x25 = 76800 \text{ Ko soit } \mathbf{75\ Mo/s}$$

Il faudrait pour stocker 1h de film avec un pareil format **263 Go** d'espace disque (3) . Il est important dans ce cas de disposer d'un système permettant de réduire le flux d'informations d'une vidéo en limitant les besoins en termes de données, tout en produisant une vidéo acceptable afin de la stocker sur un ordinateur ou pour la diffuser sur internet. Il existe plusieurs formats de compression vidéo

I.3 Codecs

Codec est un mot qui signifie « codeur/décodeur ». Il désigne un procédé qui compresse ou décompresse un signal analogique ou numérique en format de données pour la transmission, le stockage ou le cryptage de données. Le but principal d'un codec est de pouvoir traiter un maximum de données avec un minimum de ressources. Ils sont utilisés pour la téléphonie, les vidéoconférences et la diffusion sur Internet (2) .

Il existe plusieurs codecs (2) , mais ils peuvent êtres triés en deux parties en fonction de leur façon de compresser les données:

- Les codecs qui compressent de façon non-destructive, sans pertes: cela permet de retrouver un signal tel qu'il était avant le codage

- Les codecs qui compressent de façon destructive : cette façon de compresser supprime des informations jugées les moins importantes d'un signal sonore ou visuel en jouant sur les limites de la perception de l'œil ou de l'oreille humaine: par exemple les fréquences aigües (audio) ou la persistance rétinienne (vidéo) .

Il existe plusieurs formats de compression vidéo

I.3.1 H264

H.264 est protégé par un grand nombre de brevets , est un codec de compression vidéo numérique des images et vidéo haute définition à la norme MPEG-4, développé par le VCRG (Vidéo Coding Experts Group) en partenariat avec le MPEG (Moving Picture Experts Group), aussi connu sous l'appellation AVC (Advanced Vidéo Coding). Le projet H.264/AVC a permis de créer un standard de fourniture de vidéo de qualité avec un 'bit rate' sensiblement inférieur (minimum de moitié) aux précédents standards, sans en augmenter la complexité afin de conserver un niveau de design raisonnable pour un spectre de résolution élargi (4) .

I.3.2 VP8

Est un codec vidéo de On2 (5) technologie qui a remplacé VP7, son prédécesseur. Il a été annoncé le 13/09/2008. Ce format libre était destiné à trouver sa place sur le Web pour les vidéos HTML□5 pouvant être lues directement par le navigateur (6) .

Réalisé à l'origine dans un format propriétaire, il a été racheté par Google qui en a fait un format ouvert le 19 mai 2010 dans le cadre du projet WebM (7) . Il est également utilisé dans le format d'image WebM . en mai 2013 google dévoile son nouveau codec le VP9

I.4 Les Conteneurs

Un format conteneur est un format de fichier qui peut contenir plusieurs types de données compressées avec des codecs normalisés. Les formats conteneurs les plus avancés peuvent contenir de l'audio, de la vidéo, des sous titres, des chapitres et des métadonnées (auteur, date, taille etc…) .

Les conteneurs sont utilisés pour entrelacer différents types de données, par exemple les flux de vidéo, sous-titres, et même des informations de métadonnées. Une grande variété de formats de conteneurs a été mis au point, présentant différentes caractéristiques. Les conteneurs multimédias les plus importants sont brièvement présentées ci-dessous.

I.4.1 MP4

(.Mp4) [3]est un format conteneur populaire défini dans la norme MPEG-4. Il supporte presque tous les types de données multimédia. Typiquement, un conteneur MP4 contient des flux audio et vidéo codés avec H.264 et AAC, respectivement.

I.4.2 WebM

(.webm) est un conteneur de flux audio / vidéo utilisé par certains sites internet. Le format WebM permet d'encapsuler des vidéos compressées à l'aide du codec VP8, les flux audio sont encodés en Vorbis. Il est notamment utilisé par le site internet YouTube .Par conséquent, seuls les codecs open-source sont recommandé (7) .

Dans une vidéo, la question de l'objectivité en matière de qualité vidéo est malheureusement très subjective. Le vrai critère objectif est la qualité visuelle de la vidéo perçue par l'oeil humain . Quelque chose que, malheureusement, on ne peut pas mesurer et quantifier autrement que par des tests humains. Plusieurs outils au fil des années ont été développés pour tenter de comparer la qualité d'une vidéo par rapport à une autre. Le concept de base reste toujours le même, on compare, une par une, une image de la vidéo codée à l'image de la vidéo source. On obtient ainsi une série de valeurs, pour chacune des images qui composent la vidéo. La métrique classique utilisée pour comparer deux images est le **PSNR**

I.5 Mesure de la qualité vidéo

La métrique classique utilisée pour comparer deux images est le **PSNR** (Peak Signal to Noise Ratio) qui tente de déterminer le niveau de distorsion d'une image compressée par rapport à sa source. Surtout utilisée pour juger les formats de compression d'images fixes, le PSNR est considéré comme une mesure très indicative qui dépend grandement du format de compression choisi ou des particularités de l'encodeur.

Le PSNR est défini par (3) :

$$PSNR = 10\log_{10}\left(\frac{d^2}{EQM}\right)$$

où d est la dynamique du signal (la valeur maximum possible pour un pixel). Dans le cas standard d'une image où les composantes d'un pixel sont codées sur 8 bits, d = 255.

EQM est l'erreur quadratique moyenne et est définie pour 2 images Io et Ir de taille m x n comme :

$$EQM = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\|I_0(i,j) - I_r(i,j)\|^2$$

Pour visualiser une vidéo il existe plusieurs technique de diffusion

I.6 La transmission de la vidéo sur l'internet

Téléchargement Direct : Le Téléchargement direct (ou Direct-Download) permet de télécharger un fichier par l'intermédiaire d'un lien, et de le stocker dans un disque dur . Lorsqu'on clique sur un lien DDL, vidéo se télécharge puis il est automatiquement enregistré dans un support. On peut ensuite la visionner à l'aide d'un lecteur vidéo (Windows media player, Winamp, VLC etc...) (8) .

Torrent : Un torrent est un petit fichier de données que l'utilisateur reçoit après avoir cliqué sur un lien de suffixe .torrent. Le contenu décrit une ressource dont le téléchargement est possible en appliquant le P2P (est un modèle de réseau informatique proche du modèle client-serveur mais où chaque client est aussi un serveur.) à l'aide d'un logiciel tel que **BitComet , Azureus, Utorrent**... ou un autre client compatible , ces logiciels sont disponibles gratuitement sur le net. Un fichier .torrent contient l'information afférente à l'emplacement du fichier que nous voulons télécharger et non pas le fichier lui-même. En d'autres mots, c'est ce qui est envoyé à notre ordinateur quand on clique sur un torrent depuis un site Web. nous avons la possibilité de mettre en pause un téléchargement et de le reprendre plus tard (8) .

Streaming : Le streaming désigne un mode de diffusion en "direct" (ou en léger différé) sur internet de fichiers audio ou vidéo. Il permet de commencer la lecture d'un flux audio/vidéo avant la diffusion complète de celui-ci. En effet, le lecteur multimédia va récupérer une partie du contenu qu'il place dans une mémoire tampon, et en démarre la lecture dès que les données sont suffisantes pour en garantir la fluidité. En arrière-plan, le téléchargement du flux se poursuit afin d'alimenter sans cesse la mémoire tampon avec la suite du fichier. Le fichier n'est cependant pas téléchargé et les informations disparaissent dès qu'elles sont lues. De ce fait, aucune copie n'est disponible pour une écoute ultérieure (8).

I.7 Conclusion

Dans ce chapitre, nous avons présenté des généralités sur la vidéo et les différents types de codecs, à savoir H264 et VP8 ont été exposés ainsi les conteneurs qui enveloppent les métadonnées . Le H264 est un codec de compression vidéo il est utilisé par le conteneur MP4, et protégé par un grand nombre de brevet par contre le VP8 est un open source et utilisé par le conteneur webM .Pour mesurer la qualité d'une vidéo nous avons définis le PSNR .

Malgré la compression , la diffusion vidéo sur internet est toujours confrontée au problème de la limitation de la bande passante et du débit .un fichier vidéo ,même compressé , peut

avoir une taille de plusieurs méga octets , ce qui implique de longues minutes ou heures de téléchargement pour un utilisateur relié au réseau a l'aide d'un modem téléphonique de faible capacité . C'est pour pallier ce problème que la solution de streaming a été développée ; plutôt que d'obliger l'utilisateur a attendre le téléchargement complet du fichier avant d'en débuter le visionnage des que la connexion au serveur est établie .

Dans le prochain chapitre , nous présentons la méthode du streaming qui est émergente ainsi ces différentes types et on se base sur le streaming adaptatif , et on expose les différentes produits commerciaux , ainsi la norme DASH , qui utilisent le protocole HTTP .

Chapitre II. LE STREAMING

II.1 Introduction

Avec l'explosion de l'internet et l'apparition des ordinateurs personnels et les Smartphones dans le grand public , des objets multimédias sont de plus en plus diffusés . Pendant longtemps, le mode de diffusion de ces medias a toujours été le même : le fichier est d'abord télécharger puis visionné . Depuis quelques années sont apparus des technique permettant de diffuser les sons et vidéos de manières plus appropriée . Au lieu d'être d'abord téléchargé puis visionné , le fichier contenant le son et la vidéo est affichés par flot : les données sont traitées au fur et a mesure de leur arrivé , en temps réel . Par exemple , pour une vidéo ,les paquets de données reçus de l'internet seront immédiatement décompressés et les images affichées au fur et a mesure de leurs arrivées . cette technique est appelée dans la littérature anglophone (et souvent dans la littérature francophone) technique de streaming

II.2 Streaming

Le streaming est une technologie récente et émergente , elle permet le transfert des ressources média au temps réel ou a la demande a des contenus audio , vidéo et multimédia via internet ou intranet , la technologie permet de transmettre des événements enregistrées sur des support vidéo et/ou audio . Par rapport à la technique de téléchargement il permet de réduire le temps de latence , la taille des ressources a stocker en mémoire et de faire le broadcast des événement en direct (9) .

Le streaming des media est utilisé sur internet par plusieurs applications commerciales comme l'éducation a distance , le télécast , la diffusion (brodcast) des événement en direct , les services de la vidéo à la demande , la radio sur internet , etc. Il est aussi employé par les entreprises pour la téléconférence . Il y a une forte croissance dans la demande des médias en

streaming, celle-ci est à mettre en relation avec la croissance des connexions internet haut débit qui permettent un streaming de plus haute qualité et fiabilité .

Dans la Figure II-1 on peut voir un diagramme du processus complet de streaming (10) .

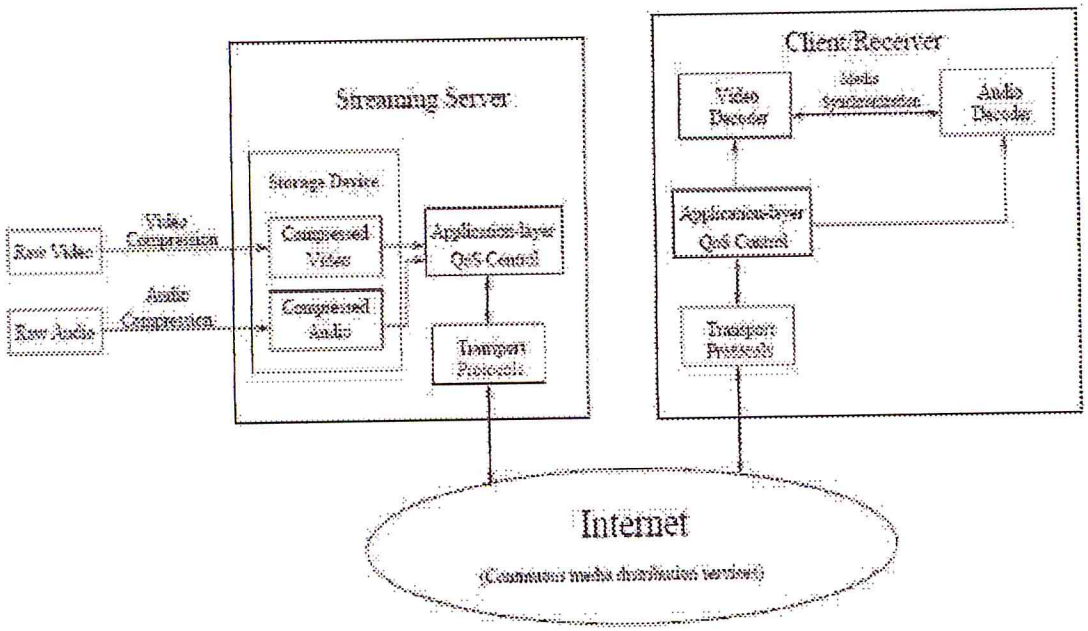


Figure II-1 Un diagramme du processus complet de streaming (10)

Le streaming peut être classé en 3 grandes types d'approches en fonction du type de contenu et de la façon dont il est mis a la disposition du public .

II.3 Les Différentes types de streaming

II.3.1 Streaming Server (Serveur de diffusion en continu)

Un serveur de diffusion en continu (streaming), est un logiciel qui reconnait le format des fichiers, la bande passante disponible, la structure des fichiers et porte attention à comment le fichier est joué sur le lecteur client. Cette méthode transmet les données en fonction du débit du fichier demandé. Ainsi, un fichier encodé à un débit de 300kbps sera livré à 300kbps. Il y aurait des précisions à faire ici, mais à la base c'est ce qui se passe. Étant donné que le serveur de diffusion est en constante communication avec le lecteur client, le débit de livraison peut même s'adapter en suivant les fluctuations du réseau. La technologie Windows Media, entre autre, peut livrer des fichiers encodés en MBR (multiple bitrates) (9) .

C'est un gros fichier dans lequel plusieurs débits du même clip se retrouvent. Ainsi, si la bande passante entre l'Internaute et le serveur de diffusion se dégradait, automatiquement, le

serveur de diffusion compenserait en livrant un des débits inférieurs inclus dans le fichier mère[7].

Le serveur de diffusion permet également de monitorer de façon précise ce que l'Internaute visionne et pendant combien de temps exactement il le regarde. Le fichier ne se retrouve pas dans son entièreté sur l'ordinateur de l'Internaute. Les données envoyées sont affichées, puis éliminées aussitôt. On a ainsi un plus grand contrôle sur le contenu.

Cette méthode est donc avantageuse dans les cas suivants (9) :

- Effectuer une diffusion en direct
- Recueillir des données statistiques extrêmement précises et détaillées sur le visionnements
- Plus grand contrôle sur l'usage de la bande passante
- Utilisée (seules les données de la portion visionnée sont transférées) Plus grand contrôle sur le contenu
- Les clips à diffuser sont de longue durée (30, 45, 60 min)

II.3.2 Streaming Progressif

Cette méthode de livraison utilise à la base un simple serveur web. Parfois, on le réfère à du "http streaming", mais en réalité, ce n'en est pas vraiment. Le serveur web ne fait que pousser les données le plus rapidement que possible sans se soucier en quel débit le fichier vidéo a été compressé. Sur la plupart des plateformes de livraison (Windows Media, Flash, Real) , le lecteur permet de démarrer le visionnement du clip avant que le fichier ne soit entièrement transféré. Aussitôt qu'il y a assez de données pour procéder à une lecture continue, le lecteur se mettra à jouer le clip (9).

Avec cette méthode, vous visionnerez le clip avec la même qualité. Seul le temps de téléchargement variera. Vous ne pourrez par contre pas avancer à un endroit du clip qui n'a pas encore été téléchargé.

Cette méthode est avantageuse dans les cas suivants (9) :

- Les vidéos à livrer sont courts (2, 3, 5, 6 minutes) et nécessitent donc un temps de téléchargement raisonnable, même avec une connexion plus lente.
- Les clips à diffuser sont avant tout des clips sur demande

- Minimiser les coûts de déploiement (simples serveurs web)
- Vous n'avez pas besoin de faire du direct

De plus, une autre méthodes autre que les deux mentionnées est également disponible c'est le streaming adaptatif

II.3.3 Streaming Adaptatif

À ces deux principales méthodes de livraison vidéo sur le Web, s'ajoute l'Adaptative Streaming ,

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 (11) .

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 [7], la vidéo est reconstituée en faisant jouer les morceaux de façon continue.

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. 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 (11).

En résumé, voici quelques avantages du Streaming Adaptative face aux autres méthodes de livraison (11):

- Coûts moins élevés pour le déploiement
- Le streaming adaptative prend avantage des serveurs Web et des caches/proxies déjà en place .
- Expérience utilisateur beaucoup plus fluide, même lors de conditions changeantes dans la qualité du réseau .

• Démarrage pratiquement immédiat du contenu vidéo en présentant à l'utilisateur le débit le moins élevé au départ et progressant graduellement au débit approprié à sa bande passante disponible.

• Aucun buffering et déconnexion .

Il existe plusieurs solutions commerciales (propriétaires) d'Adaptive Streaming par exemple Microsoft (Smooth Streaming + client Silverlight), Apple (HTTP Adaptative Streaming) et Adobe (HTTP Dynamic Streaming), et aussi la norme DASH(Dynamic Adaptative Streaming over HTTP)

II.3.4 La norme DASH

MPEG Dynamic Adaptive streaming sur HTTP (MPEG-DASH) est un protocole présenté par un groupe de travail conjoint (12) du projet de partenariat de troisième génération (3GPP) et MPEG. Ce protocole a été récemment considéré pour devenir une norme ISO (13) . MPEG-DASH définit une structure similaire à Microsoft-LSS pour diffusion adaptative, néanmoins il propose des changements dans les formats de fichiers, et definit un fichier XML.

MPEG-DASH introduit le concept de présentation par les médias. Une présentation multimédia est un collection de contenus vidéo / audio structurés

• Une présentation de média est constituée d'une séquence d'une ou de plusieurs périodes qui sont consécutives et ne se chevauchent pas.

• Chaque période est constituée d'une ou plusieurs représentations du même contenu multimédia. Les périodes ont un temps de démarrage qui est attribué par rapport au début de la présentation des médias.

• Chaque représentation spécifie un profil de qualité de la vidéo se composant de plusieurs paramètres tels que la largeur de bande, le codage, et la résolution . Les résentations contiennent un ou plusieurs segments, représentée par une adresse URL (Universal Ressource Locators)

• Les segments contiennent des fragments du contenu vidéo réelle .

Le fichier Media Présentation Description (MPD) est un fichier XML qui contient l'ensemble de la structure d'une présentation multimédia présenté ci-dessus. Une version simplifiée est

- La taille et la durée des segments (ceux-ci peuvent être sélectionnés individuellement pour chaque représentation),
- Le nombre de représentations
- Le profil de chaque représentation (bitrate, CODEC, format de conteneur, etc.)

En ce qui concerne le comportement du client, il peut avec souplesse /

- Décider quand et comment télécharger les segments.
- Choisir une représentation appropriée.
- Changer les représentations.
- Sélectionnez le transport du fichier MPD, ce qui pourrait aussi être récupéré par d'autres moyens, plutôt que seulement à travers HTTP.

Figure II-4 illustre la communication entre serveur et client dans un .MPEG-DASH , D'abord le client récupère le fichier MPD et ensuite il demande successivement les segments médias. Dans chaque période, un niveau de représentation est sélectionné (11),

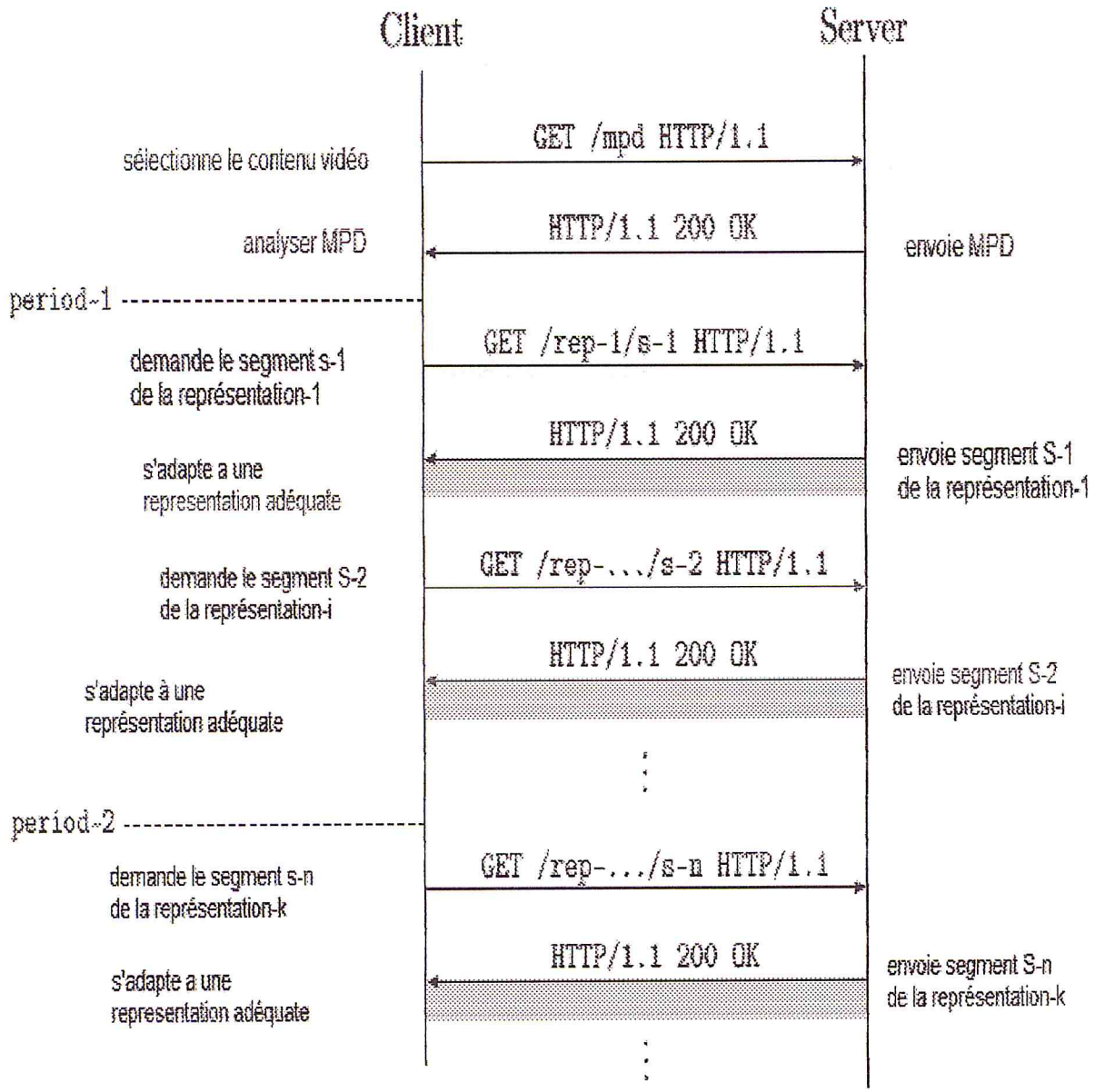


Figure II-4 Exemple d'adaptation du client en MPEG-DASH

II.3.5 Solutions propriétaires

Il existe plusieurs applications commerciales qui utilise le concept d'adaptation

II.3.5.1 Microsoft Smooth Streaming

En 2009, Microsoft Corporation a publié son approche (14) pour le streaming adaptatif sur HTTP Microsoft Live Smooth Streaming (LSS).

cette technique utilise un fichier XML (*.ism) afin de communiquer le tableau des URL au client et permettant d'identifier quels codecs, quels débits et quelles résolutions seront utilisés pour chaque fragment. Ici chaque chunk contient de l'audio et de la vidéo dans un conteneur

MP4 fragmenté . L'audio et la vidéo sont ainsi demandés séparément et le client peut accéder par système commuté vers différentes qualités (15).

Microsoft fournit une démonstration Smooth Streaming qui nécessite du plug-in Silverlight. Dans cette application en ligne, la bande passante disponible peut être facilement ajustée dans une interface utilisateur très simple. Un graphe de l'utilisation du réseau est affiché de manière dynamique ainsi que la sortie vidéo adapté Le Player silverlight ne supporte que les formats H264 et VC-1 en vidéo, ainsi que AAC et WMA en audio (16).

II.3.5.2 Apple Adaptative Streaming

En mai 2009, Apple a publié un protocole HTTP Streaming Media communication (Apple HLS) (17) . Apple HLS est basé sur la technologie de streaming Media Emblaze Network Media Streaming qui a été publié en 1998.Selon cette spécification, un flux global est décomposé en une succession de petits téléchargements de fichiers basés sur HTTP, où les utilisateurs peuvent sélectionner des flux alternatifs encodés à des débits différents (18).

Cette méthode fonctionne à travers les pare-feu et serveurs proxy (contrairement aux protocoles transitants par UDP tels que RTP qui nécessitent des ports ouverts dans le pare-feu ou nécessitent l'utilisation d'une passerelle de couche application) (18).

Ici la table des URL est fournie à partir d'un fichier texte appelé « playlist », Le niveau supérieur de la playlist contient la liste des qualités disponibles avec pour chacune d'elles une sous-playlist , La sous-playlist énumère les URL de chaque chunk. Ces chunks contiennent à la fois audio et vidéo dans un format MPEG-2 TS. Apple permet uniquement le H264, AAC et MP3 comme codec vidéo ou audio (19) . la figure 3.5 represente L'architecture de HTTP Live streaming .

Du côté serveur, le protocole est le suivant :

- le contenu multimédia est codé à différents débits pour produire des flux qui présentent le même contenu et la durée (mais avec des qualités différentes)
- Chaque flux est divisé en fichiers individuels (segments) à environ durée égale
- un fichier playlist est créé il contient une URL pour chaque fichier multimédia indique sa durée (la liste peut être consultée via une URL). d'autres changements à la playlist fichier doit être effectué automatiquement (17).

et du coté client :

- la sélection du fichier multimédia qui doit être jouée doit être faite
- recharger périodiquement le fichier playlist

La Figure II-5 représente l'architecture de Apple adaptative streaming

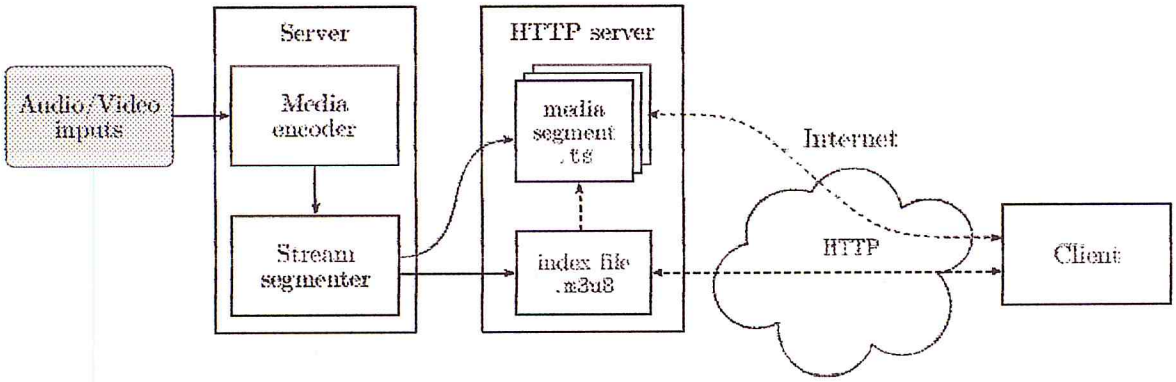


Figure II-5 L'architecture de HTTP Live Streaming (17)

II.3.5.3 Adobe Dynamic Streaming

Adobe Dynamic Streaming (HDS) supporte les protocoles HTTP et Real Time Messaging Protocol (RTMP) (20) ,Il utilise différentes spécifications de format pour les fichiers multimédias (Flash Video ou F4V, basé sur le standard MPEG-4 Part 12) et manifestes (Flash Media manifeste ou F4M). Pour déployer la solution d'Adobe, il est nécessaire de mettre en place un Flash Media Streaming Server (21), qui est un produit propriétaire et commercial.

En outre, les utilisateurs doivent installer Flash Player d'Adobe.

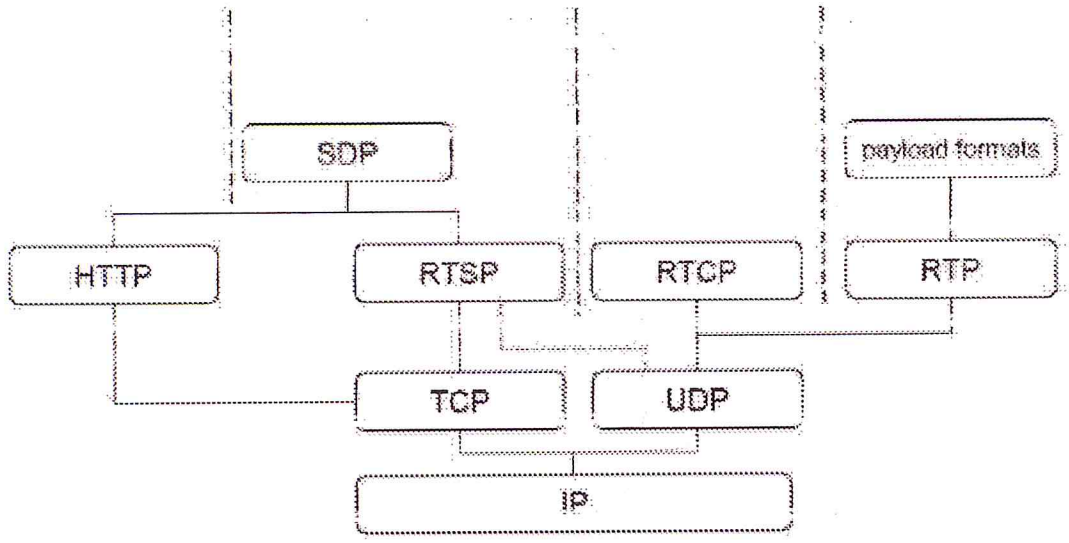
Adobe utilise aussi un fichier XML afin de communiquer au client le tableau des URL (*.f4m).Il a sa variante MP4 (F4F), Les méta datas sont aussi fragmentées.

Le flash player (client) ne supporte que H264, VP6, AAC et MP3 (21) .

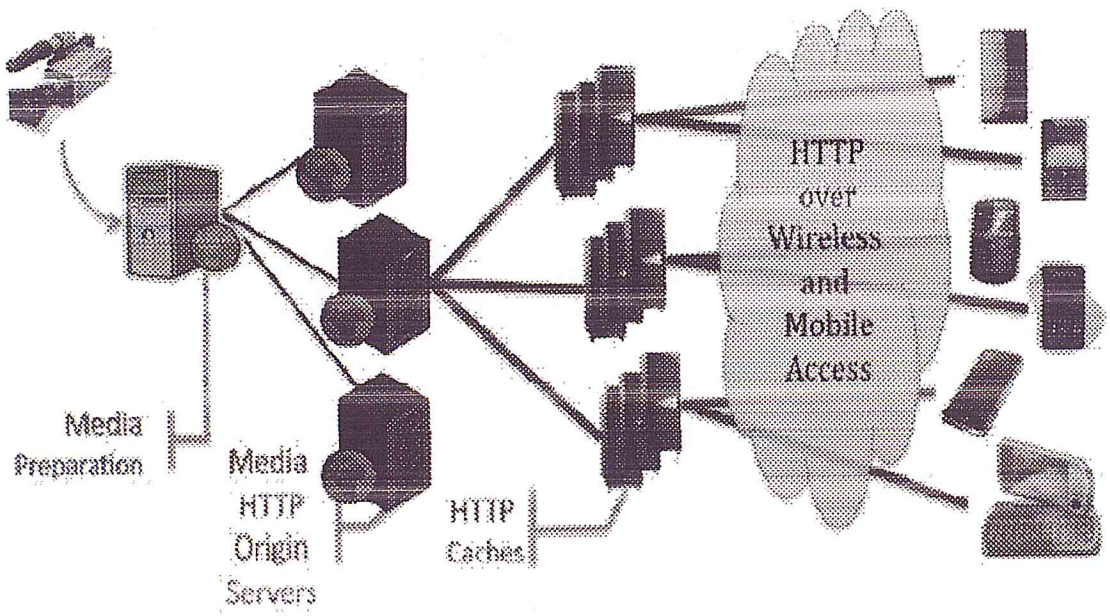
Le Tableau 1 résume les principales caractéristiques de streaming adaptative :Microsoft-LSS, Apple HLS, et MPEG-DASH (22) (23) .

En outre puis que le HTTP utilise TCP donc il obtient automatiquement une livraison de flux d'octets fiable et TCP propose de nombreux mécanismes de contrôle de congestion (24)

les figures II-6 La pile de protocoleII-7 Le principe de cashe montrent respectivement l'architecture des protocoles de streaming , le principe de cache



II-6 La pile de protocole



II-7 Le principe de cashe

Pour pourvoir connaitre le mécanisme de l'adaptation il nous a fallu d'étudier deux algorithme , le premier c'est un algorithme proposé par un group de recherche a l'université de Berlin et le deuxième est un algorithme intégrer dans la plateforme DASH

II.5 Algorithme proposée Par Konstantin Miller et Adam wolisz

II.5.1 Contexte

Konstantin Miller et al (25) travaillent dans un groupe de recherche a l'université technique de Berlin , (Allemagne). Il ont proposés le travaille suivant , Un flux vidéo composé de n segments égaux contenant chacun T secondes de lecture. T est la durée d'un segment. Chaque segment est disponible en différentes représentations.

Dans le travail présenté, cet algorithme se focalise sur la sélection dynamique d'une représentation basée sur le débit. Par ailleurs, cet algorithme peut contenir d'autres paramètres dans le processus de sélection.

Supposant un utilisateur souhaite regarder une vidéo en streaming à partir d'un serveur donné sur un réseaux :

- 1) Le client télécharge les segments dans l'ordre chronologique

- 2) Chacun de ces segments est téléchargé dans une seule représentation

- 3) les téléchargements sont non préemptif (le téléchargement du segment I - 1 doit être remplie avant le début du téléchargement du segment I).

II.5.2 Objectifs d'adaptation

Le but de l'adaptation est d'optimiser la qualité visuelle soumis à la dynamique de débit du flux TCP sur le chemin de réseau à partir du serveur vers le client. ils ont identifié les objectifs d'optimisation suivante (25) .

-1) éviter les interruptions de lecture

- 2) Maximiser la valeur minimale et la qualité vidéo moyenne.

- 3) Minimiser le nombre de basculement de la qualité vidéo.

- 4) Réduire au minimum le temps après que l'utilisateur demande à voir la vidéo et avant le début de la lecture.

Notez que les buts (1) et (2) constituent un compromis depuis le téléchargement de chaque segment de représentation le plus élevé possible , entraîne des changements fréquents de qualité de lecture chaque fois que le dynamique du débit disponible présentent de fortes fluctuations. En outre, les objectifs (3) et (4) constituent un compromis depuis le segment de début de téléchargement est minimisée si le premier segment est téléchargé à la plus basse représentation.

II.5.3 Description de l'algorithme (25)

Cet algorithme est invoqué au temps t , immédiatement après le téléchargement du segment N(t) est terminée. Afin d'adapter efficacement la qualité vidéo à la dynamique du débit disponible . l'algorithme prend deux arguments d'entrée :

- Les informations sur le débit disponible dans le passé
- niveau de la mémoire tampon (t '), t' ∈ [0, t]

L'algorithme a deux arguments de sortie :

- La représentation sélectionnée pour le téléchargement du prochain segment
- le niveau de tampon minimum en secondes quand le téléchargement démarre (Bdelay)

Le but de deuxième argument de sortie Bdelay est permet de retarder le téléchargements dans le but de réduire la mémoire tampon, s'ils arrivent déjà à la plus haute représentation ou si le débit disponible ne permet pas de sélectionner une plus haute représentation. Le téléchargement du segment N (t) + 1 ne peut pas commencer avant que le niveau de la mémoire tampon (T) tombe au-dessous Bdelay

En plus de ces deux arguments d'entrée, les algorithmes peuvent être configuré en utilisant plusieurs paramètres de configuration

$0 \leq Bmin < Blow < Bhigh$, $\Delta B > 0$, $\Delta t > 0$, $0 < \alpha 1, , , , \alpha 5 < 1$

L'algorithme sélectionne toujours la plus faible représentation de premier segment pour être téléchargé. L'avantage de cette approche est qu'elle minimise Tstart le moment où les premiers bits du premier segment arriver au niveau du client ou, de façon équivalente au retard après les demandes des utilisateurs de regarder le flux jusqu'à ce que la lecture

commence. L'inconvénient est, clairement, que les premières secondes de la vidéo sont téléchargés à plus faible qualité.

R est l'ensemble des représentations disponible dans le flux de données vidéo donné, Rf est l'ensemble des représentations possible pour un client donné .

rn(t) est la représentation n a l'instant t

rmax est la plus haute représentation et rmin est la plus bas représentation

Le Pseudo-code de l'algorithme est fourni dans l'algorithme 1.

Algorithm 1: ADAPTATION ALGORITHM

Input: $(\sigma_i)_{i=1,\ldots,n(t)}$
Output: $r_{n(t)+1}$, B_{delay}

```
1  static runningFastStart := true;
2  B_delay := 0;
3  r_{n(t)+1} := r_{n(t)};
4  if runningFastStart ...
5     ∧ r_{n(t)} ≠ r_max ...
6     ∧ β_min(t_1) ≤ β_min(t_2) ∀t_1 < t_2 ≤ t ...
7     ∧ r_{n(t)} ≤ α_1 · p̂(t − Δ_t, t) then
8        if β(t) < B_min then
9           if r↑_{n(t)} ≤ α_2 · p̂(t − Δ_t, t) then
10             r_{n(t)+1} := r↑_{n(t)};
11       else if β(t) < B_low then
12          if r↑_{n(t)} ≤ α_3 · p̂(t − Δ_t, t) then
13             r_{n(t)+1} := r↑_{n(t)};
14       else
15          if r↑_{n(t)} ≤ α_4 · p̂(t − Δ_t, t) then
16             r_{n(t)+1} := r↑_{n(t)};
17          if β(t) > B_high then
18             B_delay := B_high − τ;
19 else
20       runningFastStart := false;
21       if β(t) < B_min then
22          r_{n(t)+1} := r_min;
23       else if β(t) < B_low then
24          if r_{n(t)} ≠ r_min ∧ r_{n(t)} ≥ p̂_{n(t)} then
25             r_{n(t)+1} := r↓_{n(t)};
26       else if β(t) < B_high then
27          if r_{n(t)} = r_max ∨ r↑_{n(t)} ≥ α_5 · p̂(t − Δ_t, t) then
28             B_delay := max(β(t) − τ, B_opt);
29       else
30          if r_{n(t)} = r_max ∨ r↑_{n(t)} ≥ α_5 · p̂(t − Δ_t, t) then
31             B_delay := max(β(t) − τ, B_opt);
32          else
33             r_{n(t)+1} := r↑_{n(t)};
```

Figure II-8 Algorithme d'adaptation (25)

Après la phase de démarrage rapide. L'algorithme définit trois seuils pour le niveau de la mémoire tampon, mesurée en secondes de lecture $0 \leq Bmin < Blow < Bhigh$ tel que , on note l'intervalle [Blow, Bhigh] par B target : Btar = [Blow, Bhigh]

Bopt = 0.5(Blow + Bhigh) (25)

L'algorithme essaie de garder le niveau de tampon à proximité de Bopt. Notez que la seule façon d'augmenter B '(t), qui est la vitesse à la quelle le niveau de tampon est en train de changer, est de passer à une représentation inférieure, alors qu'il ya deux façons de diminuer B'(t). La première est de passer à un représentation plus élevée, tandis que le second est de retarder la suite téléchargements de segment (25)

Tant que (t) ∈ btar, il ne change jamais à une représentation différente. La raison derrière cela est de ne pas réagir à court terme (variations du débit disponible) . S'il observe un pic positif ou négatif d'une durée de quelques secondes sur un canal autrement constante, il ne passe pas à une représentation différente puisque il serait immédiatement forcé pour revenir. Ce comportement diminue l'expérience de visionnement par l'introduction de variations de qualité (25) .

Si une diminution du débit disponible suivie d'une diminution du niveau de tampon au dessous de Blow. L'algorithme sélection une faible représentation. Il continue à passer à une représentation inférieure tant qu'il est en dessous de blow et le débit du segment du dernier segment téléchargé est plus petit que le débit binaire de la représentation en cours de sélection (25) .

Si une augmentation du débit suivie d'une augmentation du niveau de la mémoire tampon au-dessus Bhigh. Dans ce cas, l'algorithme a deux options: rester avec la représentation actuelle et retarder le téléchargement ultérieur, ou sélectionner la prochaine représentation la plus élevée

Chaque fois que le niveau de tampon est inférieur à Bmin, l'algorithme bascule immédiatement vers le plus bas débit. La raison est pour éviter les interruptions de lecture (25) .

L' Algorithme proposée Par Konstantin Miller et al (25)est basé sur le remplissage de buffer , ils ont essayés d'éviter les interruptions de lecture mais ils ont pas vraiment basée sur la maximisation de la qualité vidéo dans la deuxième partie de l'algorithme quand le niveau de buffer est inferieur a Blow la qualité de la prochaine représentation va être diminuer malgré que le buffer n'est pas vide donc il serra un perte d'utilisation de canal se que nous allons essayé d'améliorer dans qui se suit

II.6 Algorithme présent dans la plateforme ITEC

Il est basé sur deux aspects

II.6.1 Remplissage de buffer

a t=0 le buffer level (BL) est a 0s donc BL=0s , une fois la requête est lance il commence le remplissage de buffer jusqu'au BL=20s , Une fois il atteindre le 20s il a deux chemins différent a suivre (26) :

A) Premier chemin :

Si la vitesse de lecture (VL) elle est égal a la vitesse de téléchargement (VT) alors il reste stable au point BL=20s

B) Deuxième chemin :

Il contenu le remplissage de buffer jusqu'au point BL=30s : une fois le niveau de buffer atteindre le point 30s il arrête le téléchargement et il fait que la lecture donc il vide le buffer , une fois le niveau de remplissage de buffer est < 20s il relance le téléchargement

```
DEBUT
        t=0
        BL=0
        telechargement = true
    si  telechargement = true
                    BL ++
         si BL=20
                    alors VT = VL
         sinon si 20<BL<30
                        alors VT > VL
                        BL ++
                 si BL =30
                        alors VT = 0
                        telechargement = false
                            BL - -
                 fsi
        fsi
FIN
```

Algorithme 1 remplissage de buffer

II.6.2 Choix de la représentation Vidéo

pour choisir la représentation d'ordre n l'algorithme se base sur un facteur qui est La Bande passante Cumulative BC(n). Ce dernier se base aussi sur deux facteurs les valeurs de passé et de présent qui sont représentés par Bande passante Cumulative a l'instant n-1 BC(n-1) c'est le passé , et la bande passante Mesure a l'instante n et BM(n) c'est le présent . Une fois le BC(n) calculé l'algorithme choisi la représentation R(n) la plus proche et toujours inferieur à BC(n)

Algorithme de choix de représentation

DEBUT

BC = 0

BM = 0

α=1.1

β=0.9

$$BC(n) = ((\alpha \times BC(n-1) + \beta \times BM(n)) \div 2) \times 0.9$$

R(n)- <------BC(n)

Fin

Algorithme 2 D'adaptation

II.7 Conclusion

Dans ce chapitre , nous avons présenté le streaming et ses différents types qu'ils existent , le streaming progressive ,le streaming server et le streaming adaptatif , cet dernier a été bien détaillée , nous avons exposés aussi dans le cadre de streaming adaptatif ces produits commerciaux et aussi la norme DASH , ainsi l'utilisation de protocole HTTP et pour finir nous avons présenter deux algorithmes d'adaptation .

réel par les téléspectateurs connectés. Cette retransmission live peut être suivie depuis une page web, un téléphone mobile ou avec tout autre appareil compatible. la technique VOD (Video on Demand) permet de diffuser un contenu vidéo enregistré, de toute taille, même en grande quantité, au monde entier. Cela permet aux spectateurs de visionner la video lorsqu'ils le souhaitent , sans contrainte de temps (10). C'est cette deuxième technique que nous avons retenue

III.2.3 Réseaux de transport : Internet

La communication à travers un réseau s'effectue sur un support. Ce support fournit le canal via lequel le message se déplace de la source à la destination. Les réseaux modernes utilisent principalement trois types de supports pour interconnecter des périphériques et fournir le chemin par lequel des données peuvent être transmises. Ces supports sont les suivants : Fils métalliques dans des câbles, Fibres de verre ou optiques de plastique (câbles en fibre optique),

La transmission sans fil .

Tous les supports réseau ne possèdent pas les mêmes caractéristiques et ne conviennent pas pour les mêmes objectifs. Les critères de choix d'un support réseau sont : la distance sur laquelle les supports peuvent transporter correctement un signal ; l'environnement dans lequel les supports doivent être installés ; la quantité de données et le débit de la transmission ; le coût des supports et de l'installation.

Internet est dit réseaux de type "best effort" . Les réseaux de type best effort imposent divers inconvénients Ils introduisent des retards et des erreurs. Ils rejettent les paquets , par saturation des routeurs, ils introduisent délais , gigue, bande passante limites variables . Il est difficile de (reproduire avec exactitude ces comportement). Pour le faire d'une manière artificielle il existe plusieurs outils permettant d'émuler des caractéristiques réseaux différentes . Nous pouvons en citer Dummynet, NISTNet et TC/Netem (27)

Le Figure III-2 présente les fonctionnalités de Dummynet, NISTNet et TC/Netem.

NISTNet et Netem ont des fonctionnalités très proches, car ils partagent du code, mais leur architecture est complètement différente : alors que NISTNet est construit comme un module noyau, et s'appuie sur l'horloge temps réel de la machine (RTC), Netem est intégré dans le

sous-système Linux Trafic Control (habituellement utilisé pour paramétrer de la qualité de service à l'intérieur de réseaux) (27) .

De plus, Netem est distribué avec Linux, tandis que NISTNet est distribué séparément. NISTNet n'est actuellement disponible que pour les versions du noyau antérieures au noyau 2.6.1 à cause d'un problème dans la méthode utilisée par NISTNet pour intercepter les paquets. Il est toutefois relativement simple de modifier NISTNet afin de le faire fonctionner sur des noyaux plus récents. Dummynet a été développé indépendamment de NISTNet et Netem. Son principal avantage comparé à ces derniers est qu'il permet d'intercepter à la fois les paquets entrants et sortants. La Figure III-2 résume les principales caractéristiques Dummnynet, NISTNet et Netem (27)

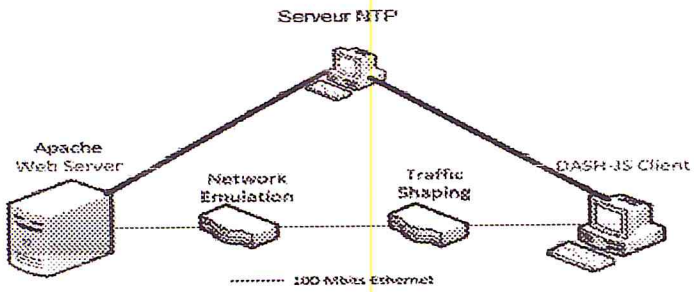
	Dummynet	NISTNet	TC/Netem
Disponibilité	Inclus dans FreeBSD	Disponible pour Linux 2.4 et 2.6 (< 2.6.14)	Inclus dans Linux 2.6
Résolution temporelle	horloge système (HZ)	Horloge temps réel (RTC)	Horloge système (HZ) ou *timer* haute résolution
Sens d'interception des paquets	Entrée et sortie	Entrée seulement	Sortie seulement
Latence	Oui, valeur constante	Oui, avec gigue corrélée suivant une distribution uniforme, normale, de Pareto, ou normal+Pareto	Oui, avec gigue corrélée suivant une distribution uniforme, normale, de Pareto, ou normal+Pareto
Limitation de bande passante	Oui, le délai à rajouter aux paquets est calculé lors de leur entrée dans Dummynet	Oui, le délai à rajouter aux paquets est calculé lors de leur entrée dans NISTNet	Oui, pas directement dans Netem, mais en utilisant le Token Bucket Filter de TC
Perte de paquets	Oui, mais sans corrélation	Oui, éventuellement corrélées	Oui, éventuellement corrélées
Réordonnancement de paquets	Non	Oui, éventuellement corrélé	Oui, éventuellement corrélé
Duplication de paquets	Non	Oui, éventuellement corrélé	Oui, éventuellement corrélé
Corruption de paquets	Non	Oui, éventuellement corrélé	Oui, éventuellement corrélé

Figure III-2 Foncionnalités de Dummynet-NIStNet , et TC/Netem (27)

Notre choix c'est naturellement porté vers NetEm puisqu'il est inclus dans linux et utilise la sortie seulement comme un sens d'interception des paquets en plus il utilise (HZ) comme horloge système . son utilisation sera détaillé dans les parties suivantes .

III.3 Plateforme d'expérimentation

Afin de mettre en place notre plate-forme d'expérimentation , il a été nécessaire de déterminer la structure du réseau permettant à un client de lire des vidéos situées sur un serveur distant. Il a également fallu déterminer comment gérer la partie adaptative. Pour ce faire, nous avons identifiés 4 fonctions essentielles , un serveur et un client , un émulateur réseau et un serveur (NTP) . Ce dernier sert à faire la synchronisation de l'horloge dans la plateforme . La figure III-3 représente notre plate forme d'expérimentation .



III-3 Plateforme d'expérimentation

Pour faire la mise en place du protocole de streaming http nous avons installer la plate forme Dash-JS sur le serveur

III.3.1 DASH : DASH-JS

Dans la partie DASH, nous avons opté pour DASH-JS. DASH-JS est une nouvelle implémentation en java-script du streaming HTTP adaptatif. Elle est suffisamment flexible , et facile à mettre en œuvre. Cette implémentation permet l'utilisation de DASH par différents types de client (mobile, browser Internet). Son utilisation conjointe avec HTML5 et la MediaSource API dans les nouveaux browser permet l'exploitation des nouvelle normes Video (WEBM, vp8) disponible en licence publique.

La Figure III-4 représente l'architecture de java script mise en œuvre par la norme MPEG-DASH. DASH-JS comprend les composants suivants:

le Media Source API le media source API permet l'envoi des données de médias à HTMLMediaElement ce dernier a son tour permet l'activation de JavaScript pour générer des flux de médias pour la lecture cette API rend le navigateur plus souple (11) .

Les gestionnaires d'événements (Event Handlers) joue le rôle d'une interface pour le média Source API et aussi un déclencheurs des événement lancer par le médias source API. Lorsque le Media Source API délivre l'événement **"source open"** le gestionnaire d'événements déclenchera le téléchargement du MPD et ensuite le MPD sera remis à l'analyseur MPD (MPD Parser) et quand il délivre l'événement **"source ended "** l'event handlers finira le telechargement du fichier MPD (11) .

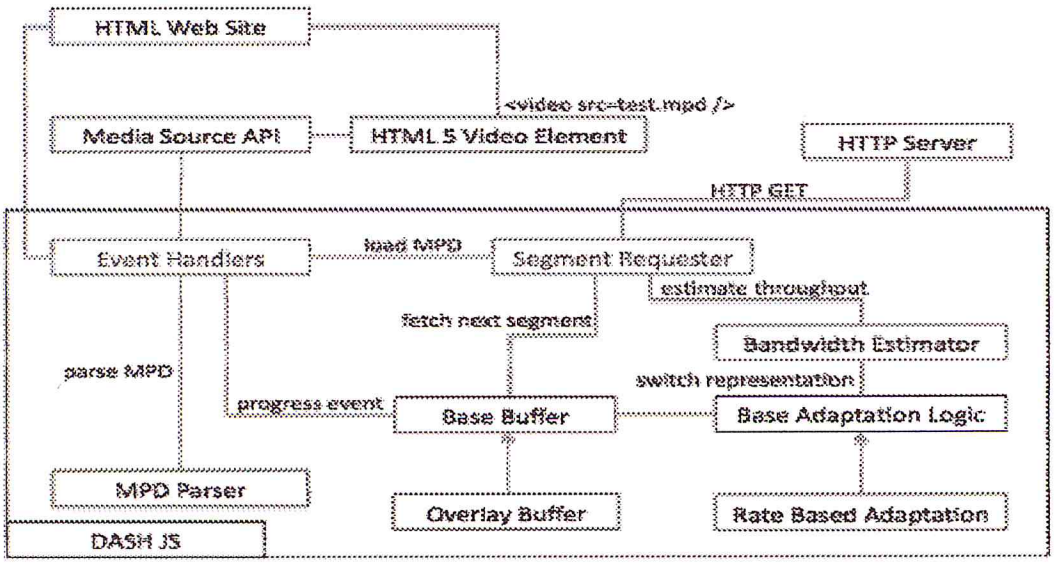


Figure III-4 L'architecture de DASH-JS (11)

L'événement Progress est utilisé pour appeler la mémoire tampon (buffer) qui est responsable de la récupération et la mise en mémoire tampon des segments de la représentation multimédia (11) .

L'analyseur MPD (MPD Parser) est utilisé pour analyser la demande de MPD. Cela va générer un objet qui contient toutes les informations sur le MPD concernant la représentation et leurs segments, le bitrate de chaque représentation, la résolution des images vidéo. ces

informations sont utilisées par la logique d'adaptation pour déterminer des représentations qui seront adéquates (11) .

le demandeur de segment (segment request) fait la demande de MPD et utilise l'objet XMLHttpRequest pour générer les demandes http (11) .

la bande passante Estimer (bandwidth estimator) est utilisé par le demandeur de segment c'est pour mesurer et estimer le débit mesuré .

Le tampon de base de la classe (class Base Buffer) fournit la classe de base pour le tampon implémentations. Il offre la possibilité d'enregistrer les gestionnaires d'événements pour des événements spécifiques (par exemple, criticalFillLevel) (11) .

En raison du fait que le Media Source API ne permet pas l'accès à la mémoire tampon de l'élément vidéo HTML5 il existe **un tampon de recouvrement (Overlay Buffer).** Ce tampon hérite le tampon de base et imite le tampon de la vidéo réelle (11) .

Le tampon (l'etat de remplissage de la memoire tampon) et la largeur de la bande estimée peuvent être utilisé comme des facteurs pour décider qu'elle est la prochaine représentation qui doit être sélectionné . Pour permettre la mise en œuvre des differentes logiques d'adaptation de la classe de base .

Logique d'adaptation (base Adaptation Logic) : Chaque logique d'adaptation doit hériter de cette classe afin d'être utilisé. La logique d'adaptation est appelée après chaque segment téléchargé. La classe de base met en œuvre une seule méthode appelée switchRepresentation (). Cette méthode doit contenir la logique de décision d'adaptation (11).

Media Source API est activé dans ces derniers version (> v. 23) de google chrome mais dans les anciennes versions peut être qu'on est obligé de l'activé via chrome://flags/ comme la montre la Figure III-5

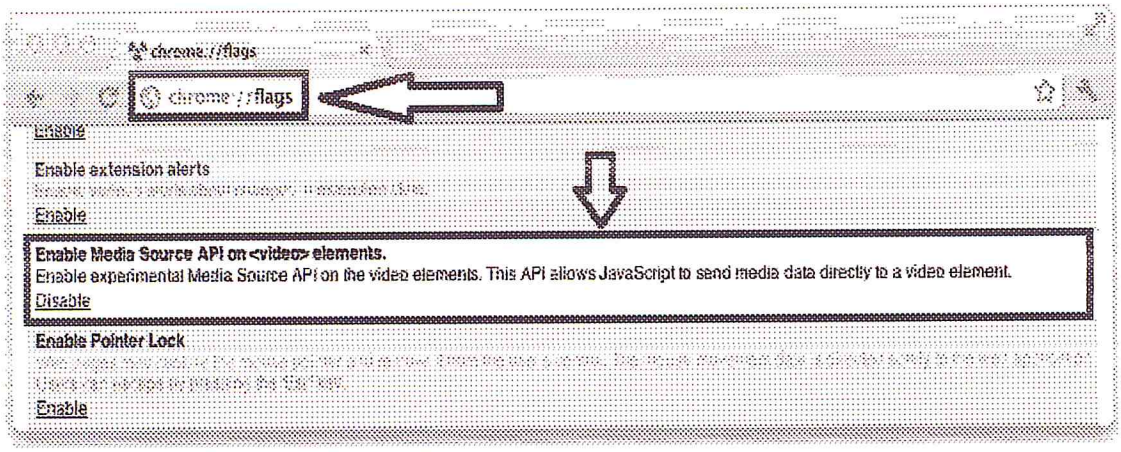


Figure III-5 Activation de Media Source API

Pour que le client puisse lire la vidéo sur le serveur il nous a fallu de convertir la vidéo au format convenable .

III.3.2 Le contenu : Vidéo

Dans cette partie, nous avons naturellement opté pour de la VOD. Ce choix nous facilite relativement la génération du contenu vidéo sans pour autant altérer la nature des résultats escomptées. Aussi nous avons utilisé une banque de vidéo disponibles gratuitement sur ce lien http://www-itec.uni-klu.ac.at/dash/?page_id=6 . Ces vidéos sont disponibles au format brut YUV (y4m), nous les avons donc compressé au format VP8 et mis dans des conteneurs WEBM par l'outil FFMPEG (référence). Comme nous l'avons vu plus haut, le format WEBM (vp8) est récent, performant disponible gratuitement et intégré dans Google Chrome.

III.3.2.1 FFMPEG

FFMPEG est un ensemble de programmes open source dédiés au traitement de flux numériques audio et vidéos (enregistrement, logiciel de lecture et encodage de vidéo). , compatible avec de nombreuses plateformes (Linux, Windows...) . il assure en ligne de commande la possibilité de convertir les fichiers vidéo d'un format à un autre[36] .Il est simple et facile à installer sur ubuntu ,il suffit d'exécuter la commande suivante

```
sudo apt-get install ffmpeg
```

Pour utiliser la métrique qui compare deux images (PSNR) définit dans le chapitre 1 , nous avons utiliser la framework EvalVid

III.3.2.2 Mesure de la qualité Vidéo : EvalVid :

Pour évaluer la qualité de la vidéo reçue, nous avons prévu d'utiliser le Framework Evalvid dans lequel la métrique utilisée est le PSNR (Peak Signal to Noise Ratio).Cette métrique donne des résultats assez fiable avec le format yuv . (yuv un format pour représenter la vidéo dans un format brut, sans aucune perte de qualité ou de compression .)

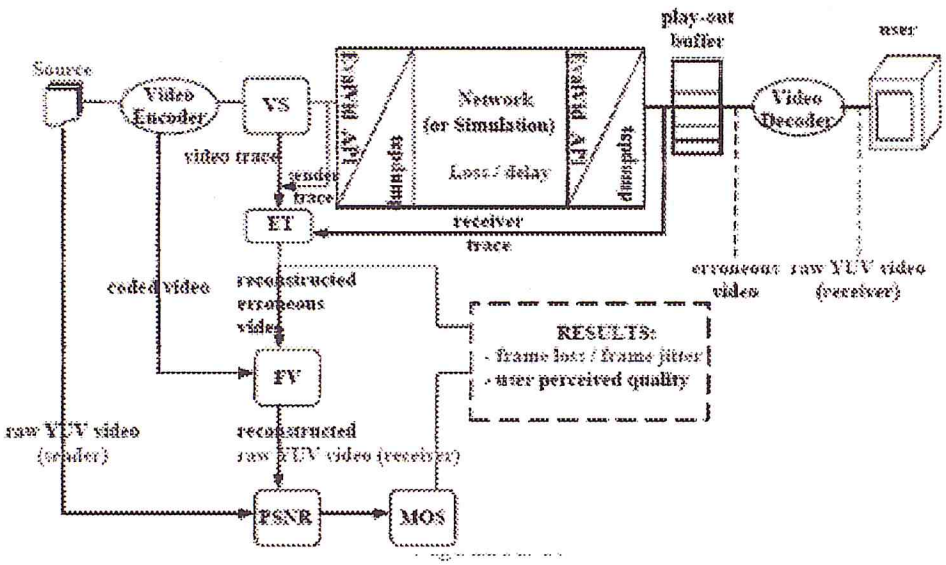
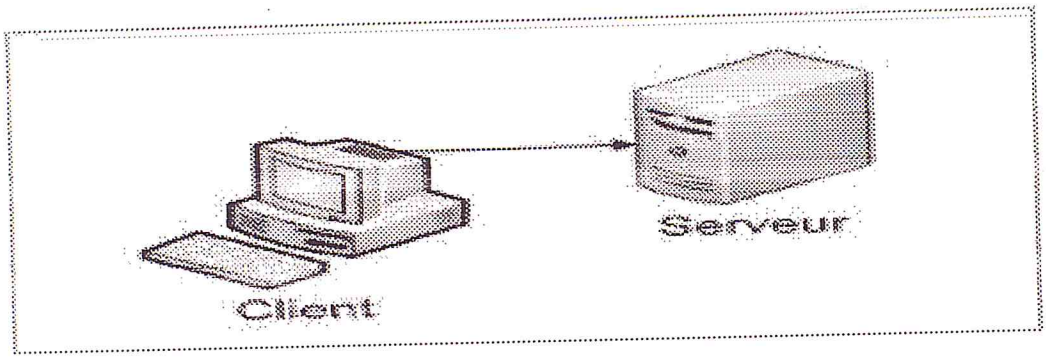


Figure III-6 Structure géneral de EvalVid

III.3.3 Streaming : Modèle Client /Serveur

Comme nous l'avons indiqués dans le chapitre streaming , nous avons utilisés un serveur HTTP et pour le client nous avons installés le navigateur web google chrome pour deux raison, la premier parce qu'il contient les tags HTML 5 ,ce dernier est trés important pour le fonctionnement de DASH et la deuxième raison c'est bien parce qu'il utilise le Media Source API



III-7 Client-Serveur

III.3.3.1 Serveur

Comme serveur HTTP nous avons utilisées le Apache, ce dernier est connu comme **Apache HTTP Server** , est une norme établie dans la distribution en ligne de services de site Web, qui a donné l'impulsion initiale pour l'expansion du World Wide Web. Il s'agit d'une plate-forme serveur web open-source, ce qui garantit la disponibilité en ligne de la plupart des sites actifs aujourd'hui. Le serveur est destiné à servir un grand nombre de très populaires plates-formes web modernes / systèmes d'exploitation tels que Unix, Windows, Linux, Solaris, Novell NetWare, FreeBSD, Mac OS X, Microsoft Windows, OS / 2, . . etc.

Le serveur Apache est le serveur Web le plus populaire sur l'Internet depuis Avril 1996. et en 2007 selon netcraft plus de 60% des sites web utilisent le serveur apache , Il n'est en aucun cas considérée comme un critère de plate-forme pour le développement et l'évaluation d'autres serveurs web réussie.

Pour reproduire les fonctionnalités offertes par la plate forme réel sur notre plate forme d'expérimentation il nous a fallu de faire une émulation , cette dernière donne un comportement similaire au plate forme réel .

III.3.4 Réseau : Emulation

L'émulation permet de reproduire le comportement d'un système donné sur un autre système. Pour cela, le système d'émulation doit être dimensionné par rapport au plateforme réel.

Sur la Figure III-8 on constate que les services rendus par le système réel et par le système émulé sont équivalents pour les applications ou les protocoles.

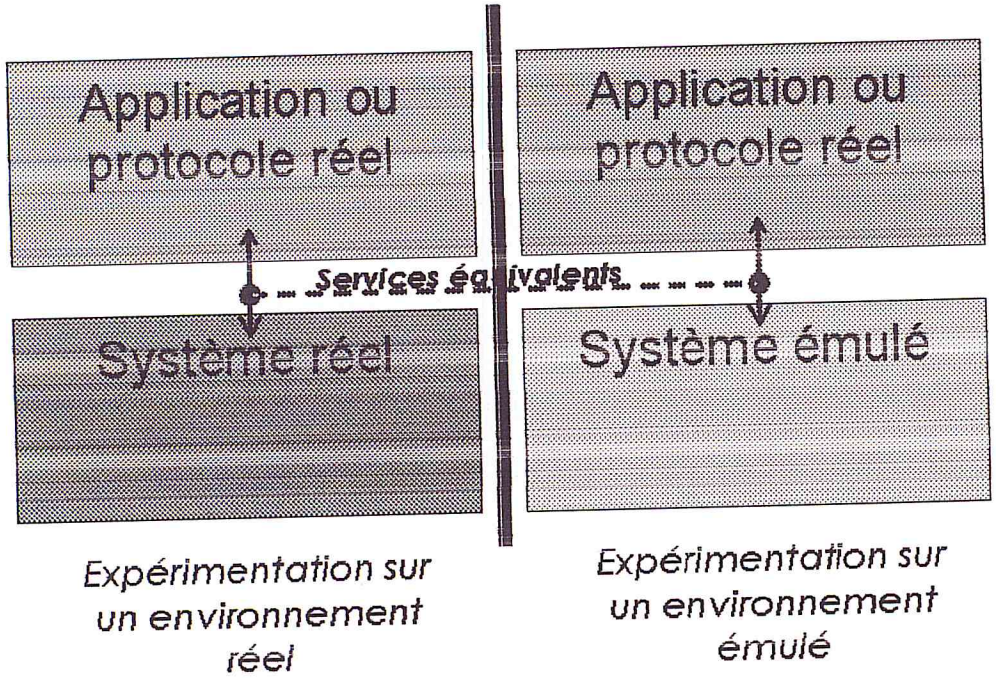


Figure III-8 Expérimentation en environnement réel ou émulé

Pour que l'on puisse faire cette émulation sur notre plateforme nous avons utiliser un logiciel qui s'appelle netem , Netem (fonctionnant avec les noyau Linux ≥ 2.6) permet d'émuler les propriétés de nombreux réseaux. La dernière version permet d'émuler des délais variables, des pertes, des duplications et des dé-séquencements de paquets, mais il faut aussi noter que netem est toujours en développement . Dans notre plateforme nous nous sommes concentrés sur le delai de transit et la bande passante . Pour utiliser netem, nous avons installer le paquet iproute dans le gestionnaire de paquets Ubuntu.

sudo apt-get install iproute

Une fois iproute est installé , nous pouvons utiliser l' outil, le Trafic Control (Tc), qui met en œuvre un ensemble de mécanismes afin de conditionner le trafic réseau. Tc est un programme utilisateur permettant de créer et d'associer des files à une interface de sortie. C'est un tout en un, il est utilisé pour installer divers types de files, associer des classes à ces files, mettre en place les filtres de classification.

La configuration de ce module se fait via la commande en ligne tc. tc fait partie du package iproute2 . Le principal inconvénient de cet outil résidait dans le fait qu'il ne proposait que la limitation de bande passante mais pas de paramétrage de délai ou de pertes. La qdisc netem,

développée par S. Hemminger, résout ce problème qui permettent à l'utilisateur, grâce au mécanisme TBF (Token Bucket Filter) de séparer en queues les différents flux .

III.4 Mise en place de la plateforme d'expérimentation:

Pour mettre en place de notre plateforme d'expérimentation et la rendre opérationnelle ,nous avons installé le serveur et le client , nous avons installer l'outil DASH-JS matser et nous avons générés le contenu nécessaire. Nous avons ensuite généré les fichiers a partir de lien suivant : https://github.com/dazedsheep/DASH-JS

Voici sur la Figure III-9 les étapes que nous avons suivis dans cette partie pour la mise en place de la plate forme d'expérimentation .

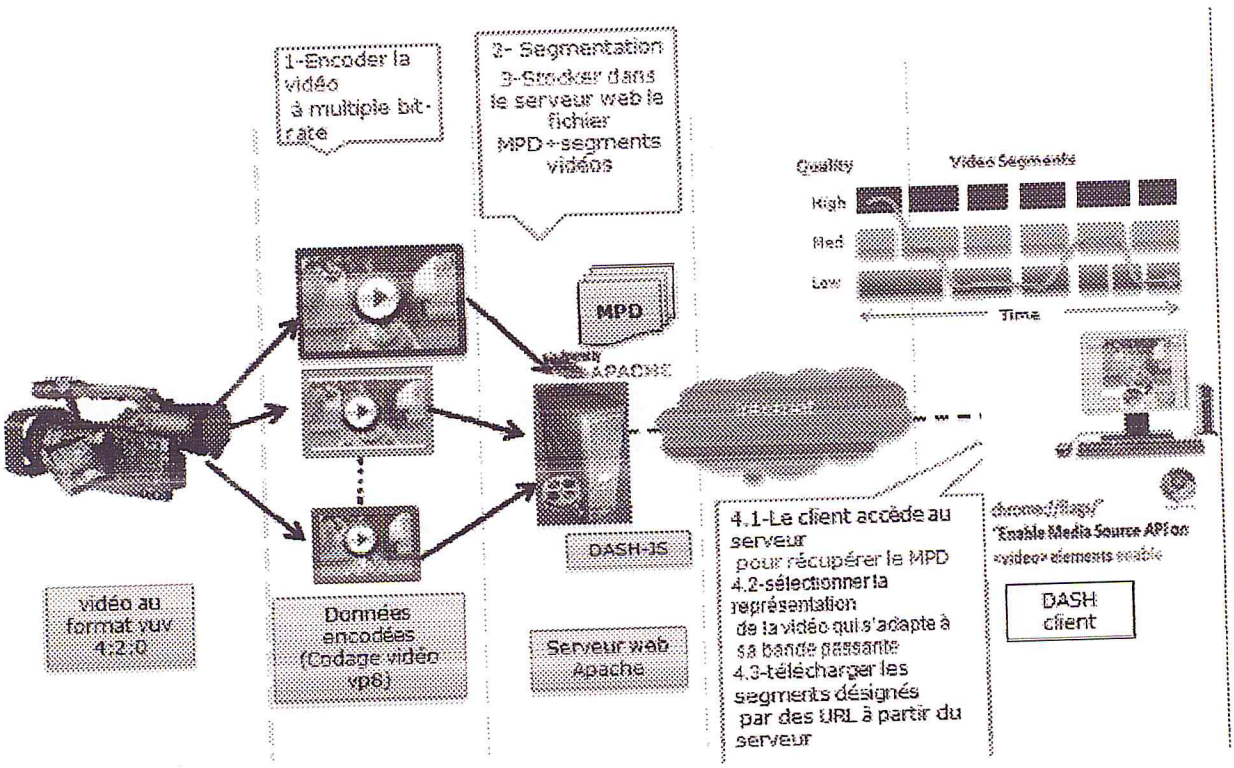


Figure III-9 Mise en place de la plate forme d'expérimentation

III.4.1 Vidéo

Tout d'abord nous avons commencé par encoder la vidéo à multiple bit-rate avec la préparation du contenu DASH en découpant chaque vidéo en plusieurs segments (*chunks*) de

2 secondes et la génération du fichier de description MPD (Media Presentation Description) de plusieurs qualités .Ces fichiers ainsi les vidéos ont été stockés dans le serveur web le fichier MPD + segments vidéos .

III.4.1.1 La Conversion De La Vidéo

l'encodage des vidéos peut se faire par deux étape

1ere étape : pour faire les mesures de qualité (PSNR) nous devons d'abord convertir la vidéo en format yuv , car y4m c'est les entête + yuv . A l'aide de logiciel FFMPEG il suffit Just d'exécuter la commande suivante :

```
ffmpeg -i nomvidéo.y4m nomvidéo.yuv
```

Figure III-10 Commande de conversion de format y4m vers yuv

2eme étape : Convertir une vidéo yuv vers WebM

la Figure III-11 représente le mécanisme de la conversion de yuv vers WebM

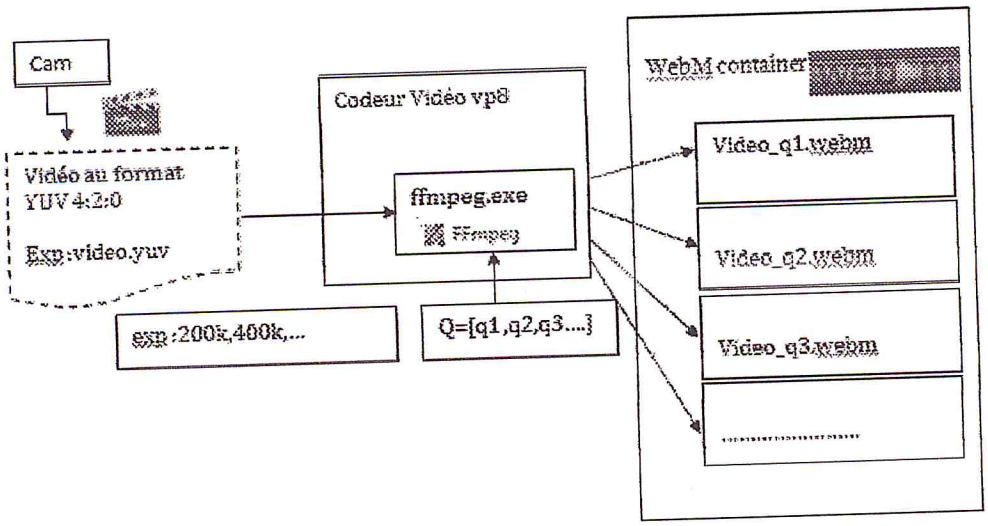


Figure III-11 Conversion Vidéo de format yuv vers webM

aprés le passage du format y4m a yuv maintenent on fait la conversion de la vidéo du format yuv vers WebM en executant la comande suivante :

Figure III-12 La commande qui fait la conversion de format yuv vers WebM

Le Tableau 2 représente la description de la commande qui fait la conversion de la vidéo du format yuv vers le Webm

Paramétre de ffmpeg	Description de paramètre	Exemple
-i	Nom du fichier yuv"	Big_buck_bunny.yuv
-vcodec	libvpx (VP8) as the video encoder	libvpx
-vb	$b:bit rate	200k
-keyint_min –g48	-keyint_min 48 –g 48	
	FFmpeg à utiliser 48 images que l'intervalle de trame clé minimale et maximale. Ces valeurs sont utilisées pour aligner tous les points de synchronisation dans les flux de données vidéo d'être commutés. Dans le courant de WebM DASH javascript joueur ces valeurs doivent être identiques	
-an	Cela indique FFmpeg pour ne pas sortir un flux audio	
nomV.webm	Fichier de sorti	
$3 * $4	Longuer*largeur	640*320

Tableau 2 tableau descriptif de la commande ffmpeg

Afin de convertir plusieurs qualités , nous avons implémenter cette commande dans un script qui nous permet , a partir d'un fichier yuv de convertir plusieurs qualités .

Script de conversion d'un fichier yuv en webm

```
#!/bin/bash

if [ $# = 3 ]

then

for b in $2; do ffmpeg -i $1.yuv -s resolution -vcodec libvpx -vb $b -
keyint_min 48

-g 48 -an $1"_"$b.webm; done

else

echo "Syntaxe:$0 nom_videoenYuv listBitrate resolution"

fi

$1 :nom du vidéo

$2 :Bitrate(kbps)

$3:resolution
```

III.4.1.2 Segmentation (Création de MPD de plusieurs Qualités)

Après la conversion de la vidéo en plusieurs qualités nous allons créer un fichier MPD . Un fichier MPD est un document XML qui décrit les différents médias disponibles sur le serveur

ainsi que leurs localisations. Le client peut sélectionner, via toutes les métadonnées fournies, le média qui répond à ses exigences. Celui-ci s'adapte aux propriétés du réseau du client .Via un script python on exécutons la commande suivant :

```
allo@allo-HP-G72-Notebook-PC:/var/www/dash/DASH-JS-master$ python  create_mpd_segment_info.py
<file><bitRates><Segment_duration><min_buffer_time><base_url>
```

Figure III-13 commande de creationde fichier MPD

l'exécution de cette commande elle nous permet de crées pour chaque qualité d'une vidéo un fichier mpd

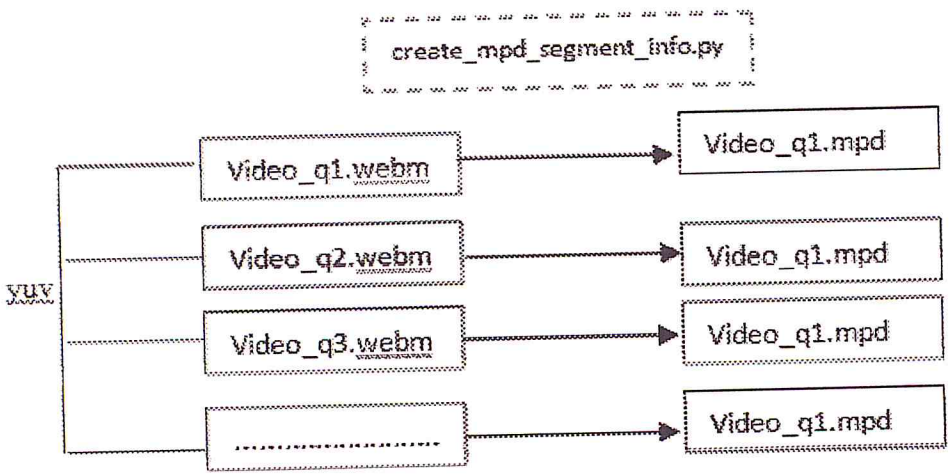


Figure III-14 création de fichier MPD pour chaque qualité

Nous avons fait une modification dans ce script qui nous a permis de crée un fichier mpd pour toutes les qualités de la vidéo . Il suffit juste d'exécuter la nouvelle commande

```
python  create_mpd.py <nomF><bitRates><Segment_duration><min_buffer_time><base_url> > <nomFS>
```

Figure III-15 la commande de création d'un fichier MPD

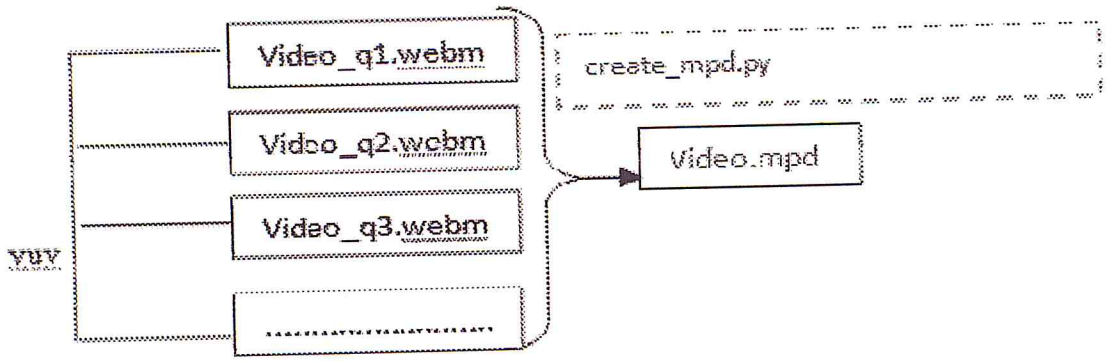


Figure III-16 fichier MPD pour toute les qualités d'une Vidéo

Dans Tableau 3 on décrit toutes les fonctionnalités de la commande de création de fichier MPD :

Paramètre	Description de paramètre	Exemple
<nomF>	Nom du fichier WebM "NomVideo"	big_buck_bunny_360p24
<bitRates>	La listes des bit rates	"200k 400k . . ."
<Segment_duration>	La longueur du segment en secondes(<SegmentList duration= Segment_duration)	2
<min_buffer_time>	Taille de tampon minimale(<minBufferTime>)	15
<base_url>	URL utilisée dans l'élément(<BaseURL>)	http://localhost/dash/DASH_JS/
<nomFS>	Nom du fichier de sorti "NomFichier MPD"	Big_buck_bunny.mdp

Tableau 3 description de la nouvelle commande

```
alilo@alilo-HP-G72-Notebook-PC:~$ sudo python ../create_mpd.py  VideoData/big_b
uck_bunny2/big_buck_bunny_360p24 "200k 400k 600k 800k 1000k 1200k 1400k 1600k 1
800k 2000k 2200k" 2 15  http://10.1.60.64/dash/DASH-JS-master/EmulationNetem/ >
 /home/alilo/documentation/big_buck_bunny.mpd
```

Figure III-17 Exemple du nouvelle commande

Aprés la préparation de notre serveur L'étape suivante sera consacré à l'émulation de réseaux

III.4.2 Émulation de réseaux

III.4.2.1 Simuler un délai de transit constant

Le délai est le temps de transit réseau d'un paquet IP. Il dépend de plusieurs paramètres (traversé des équipements, taille des buffers et distance physique entre les deux points du réseau). Nous allons utiliser la commande *delay* qui va simuler un délai de transit de X ms sur tout les paquets IP **sortant** de l'interface réseau. On va utiliser la commande "ping" pour vérifier que tout fonctionne comme prévu. Test du réseau avant la commande tc:

```
alilo@alilo-HP-G72-Notebook-PC:~$ ping 10.1.60.61
PING 10.1.60.61 (10.1.60.61) 56(84) bytes of data.
64 bytes from 10.1.60.61: icmp_req=1 ttl=64 time=0.211 ms
64 bytes from 10.1.60.61: icmp_req=2 ttl=64 time=0.218 ms
64 bytes from 10.1.60.61: icmp_req=3 ttl=64 time=0.218 ms
64 bytes from 10.1.60.61: icmp_req=4 ttl=64 time=0.187 ms
64 bytes from 10.1.60.61: icmp_req=5 ttl=64 time=0.209 ms
64 bytes from 10.1.60.61: icmp_req=6 ttl=64 time=0.238 ms
64 bytes from 10.1.60.61: icmp_req=7 ttl=64 time=0.216 ms
64 bytes from 10.1.60.61: icmp_req=8 ttl=64 time=0.211 ms
64 bytes from 10.1.60.61: icmp_req=9 ttl=64 time=0.195 ms
64 bytes from 10.1.60.61: icmp_req=10 ttl=64 time=0.231 ms
64 bytes from 10.1.60.61: icmp_req=11 ttl=64 time=0.209 ms
64 bytes from 10.1.60.61: icmp_req=12 ttl=64 time=0.192 ms
64 bytes from 10.1.60.61: icmp_req=13 ttl=64 time=0.218 ms
64 bytes from 10.1.60.61: icmp_req=14 ttl=64 time=0.219 ms
64 bytes from 10.1.60.61: icmp_req=15 ttl=64 time=0.214 ms
64 bytes from 10.1.60.61: icmp_req=16 ttl=64 time=0.221 ms
```

Figure III-18 Test de Réseaux avant la commande TC

On simule un délai de **20**ms sur tout les paquets sortant (soit environ le délais sur une liaison ADSL):

Figure III-19 commande Tc pour simuler un délai de 20ms sur tout les paquets sortant

et on re-test notre réseau

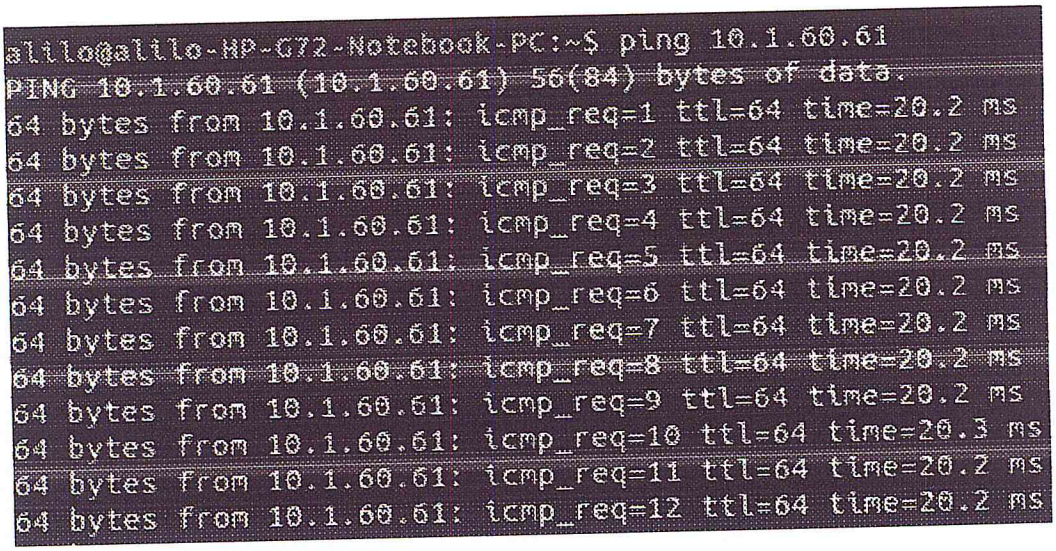


Figure III-20 resultat de test aprés l'execution de la commande Tc

III.4.2.2 Simuler une bande passante limite

Cette fonction ne fait pas partie de Netem mais utilise tout de même la commande tc pour se configurer. Avant de commencer nous allons tester la capacité de notre réseau LAN avec la commande IPerf (à lancer en mode serveur UDP sur notre machine , 10.1.60.61 dans notre cas):

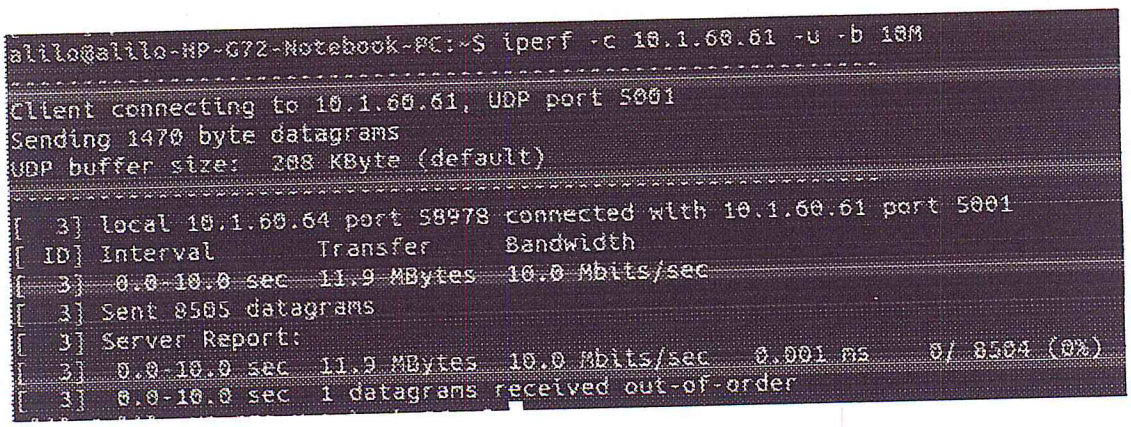


Figure III-21 Commande Iperf pour tester la capacité de notre réseau LAN

On a donc bien un débit de 10 Mbps.

Puis on ajoute un "tuyau" limitant le trafic sortant à 512 Kbps

```
allio@allio-HP-G72-Notebook-PC:~$ sudo tc qdisc replace dev eth0 root tbf rate 5
12kbit  burst 12kb latency 1s
```

Figure III-22 Commande Tc pour simuler un trafic sortant a 521 kbps

et on re-test notre réseau

```
allio@allio-HP-G72-Notebook-PC:~$ iperf -c 10.1.60.61 -u -b 10M
------------------------------------------------------------
Client connecting to 10.1.60.61, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.1.60.64 port 60532 connected with 10.1.60.61 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  11.9 MBytes  10.0 Mbits/sec
[  3] Sent 8505 datagrams
[  3] Server Report:
[  3]  0.0-11.3 sec   690 KBytes   499 Kbits/sec  10.277 ms 8022/ 8503 (94%)
```

Figure III-23 Résultat de test

On arrive bien à limiter le débit réseau sortant à 499 Kbps .

après nous avons commencer par faire des test sur une vidéo de 10mnt , nous avons fixer le bitRate et on varier le burst , les résultat sont dans le tableau ci-dessus :

Bitrate	Burst	N° 01er seg	Time
300K	4 Kb	10	238,05
	5 Kb	1	4,86
	6 Kb	1	4,49
	9 Kb	1	4,3944
	10 Kb	1	3,58
500K	9Kb	1	4,05
	12 Kb	1	3,78
	14 Kb	1	3,74

700K	8 Kb	3	18,08
	10 Kb	1	3,77
	12 Kb	1	3,85
	14 Kb	1	3,9
900K	10Kb	8	39,5
	12 Kb	1	3,69
	14 Kb	1	3,64
	16 Kb	1	3,7
	18 Kb	1	3,72
	20 Kb	1	3,69
1000 K	14 Kb	1	3,78
	16 Kb	1	3,83
	18 Kb	1	3,91
	20 Kb	1	3,88
	22 Kb	1	3,86
	24 Kb	1	3,88
1100 K	16 Kb	1	4,01
	18 Kb	1	3,99
	20 Kb	1	4,03
	24 Kb	1	4,11
	28 Kb	1	4,09
1300 K	18 Kb	1	4,19
	20 Kb	1	4,11
	24 Kb	1	4,22
	28 Kb	1	4,27
	32 Kb	1	4,29
1500 K	20 Kb	1	4,3
	24 Kb	1	4,35
	28 Kb	1	4,38
	32 Kb	1	4,37
	36 Kb	2	7,5

	38 Kb	2	7,6
800 K	30 Kb	1	4,63
	32 Kb	1	4,62
	38 Kb	2	7,27
	40 Kb	2	7,31
	42 Kb	2	7,3
2100 K	50 Kb	2	5,84
	60 Kb	2	5,9
	70 Kb	2	5,52
	80 Kb	2	5,53
	100 Kb	2	5,32
	130 Kb	2	4,99
	160 Kb	2	4,86
2400 K	70 Kb	2	6,69
	80 Kb	2	6,5
	90 Kb	2	6,49
	110 Kb	2	6,3
	140 Kb	2	6,06
	170 Kb	2	6

Tableau 4 Tableau des resultats des tests

La valeur du burst doit être varie entre [0.8 ; 1.2] du rapport BitRate/fréquence d'interruption

Pour revenir à la configuration initiale (sans simulateur), on utilise la commande suivante:

```
allio@allio-HP-G72-Notebook-PC:~$ sudo tc qdisc del dev eth0 root
```

Figure III-24 Commande Tc pour revenir a la configuration initiale

nous nous avons implémenté toutes ces commandes dans un script python qui nous permet de fixer le délais et d'obtenir des différent bitRate a l'aide de la fonction uniforme qui nous aide a crée un canal difficile des différentes débits vont être générer chaque 10 sec

```
delay=sys.argv[1]      # delay de transmission d'un paquet
CMD1='tc qdisc add dev  eth0 root handle 1:0 netem delay'+delay+'ms'
os.system(CMD1)
for i in range (0.60)
   bitRate=int(uniform(200,2200))  #fonction unifrom
   bur=int(biteRate)/int(48)
   CMD2='tc qdisc replace dev eth0 root tbf rate' + str(bitrate)+'Kbit burst' +str(bur)+'Kb latency ls'
   print str(bitrate)+'kbit'
   os.system(CMD2)
   time.sleeep(10)
```

Figure III-25 Script python (coté serveur)

Dans l'étape suivante nous allons faire quelque test sur la bande passante (bitRate) et le burst sur une vidéo de 10mnt (big_buck_bunny_360p24.html) pour faire ces tests il faut d'abord qu'on règle le problème de retard (horloge) entre le client et le serveur donc il nous faut un serveur NTP qui fait cette synchronisation

III.4.3 Serveur NTP

De nos jours, les équipements informatiques disposent d'une horloge à laquelle ils font référence pour horodater les connexions des utilisateurs, les fichiers, les transactions, les courriers électroniques, et bien plus.

Lorsque les machines sont en réseau et partagent des ressources communes comme des systèmes de fichiers, des systèmes de gestion du réseau (sécurité, logs, certificats), ceci deviendra d'autant plus gênant si les horloges des ordinateurs ne sont pas synchronisées.

La synchronisation des horloges des équipements devient capitale, car permet d'éviter certaines pannes qui pourraient paralyser le réseau. C'est la finalité du protocole NTP

(Network Time Protocol). Ce tutoriel explique comment configurer un server NTP Sous Red Hat entreprise Linux 6.

III.4.3.1 Définition

Le Protocole d'Heure Réseau (*Network Time Protocol* ou NTP) est un protocole qui permet de synchroniser, via un réseau informatique, l'horloge locale d'ordinateurs sur une référence d'heure. C'est un protocole basé sur UDP et utilise le port 123.

Un serveur NTP est tout ordinateur sur lequel est installé le service ntp et sert de référence d'heure pour un ensemble d'ordinateurs clients. Celui-ci peut être configuré pour utiliser une horloge matérielle interne ou une source de temps externe

Un client NTP est tout ordinateur sur lequel est installé le service ntp et synchronise son horloge local sur une référence d'heure d'un autre ordinateur appelé server NTP.[40]

L' installation du server NTP :il suffit d'éxécuter la commande

```
allio@allio-HP-G72-Notebook-PC:~$ sudo apt-get install ntpdate
```

Figure III-26 Commande d'installation de NTP

Les paquets nécessaires (ntp-4.2.4p8-2.el6.i686, ntpdate-4.2.4p8-2.el6.i686) au fonctionnement du service NTP sont par défaut disponibles dans toute installation de base de RedHat entreprise.

```
[root@server ~]# rpm -qa ntp*
ntp-4.2.4p8-2.el6.i686
ntpdate-4.2.4p8-2.el6.i686
[root@server ~]# _
```

Implémentation de la commande ntp sur coté serveur et aussi l'exécution de Netem

```
CMD4='ntpdate -u 10.1.60.59'
        os.systm(CMD4)
        delay=sys.argv[1]      #delay de transmission d'un paquet
        CMD1='tc qdisc add dev eth0 root handle 1:0 netem delay '+ delay + 'ms'
        os.system(CMD1)
        #Lire jusqu'a la fin du liste des bit Rate
        rates = []
        i=0
        fname = "RATES.csv"
        file =open (fname, "wb")
      try :
          c = csv.writer (file.delimiter = ' ')
          lineterminator = "\r\n"
          for i in range (0,66) :
                  bitRate =int(uniform(200.2200))
                  bur=int (bitRate)/int (48)
                  CMD2='tc qdisc replace dev dev eth0 root tbf rate' + str(bitRate)+'kbit burst '+
                  str(bur)+'kb latency ls'
                  print str (bitRate)+'kbit'
                  os.system(CMD2)
                  time.sleep(10)
                  rates.append(str(biteRate)+" " + str (tps))
                  tpss.append(tps)
                  c.writerow( str(int(tps)), str (bitRate)))
                  i+=1
                  print i
```

Figure III-27 Script python avec l'implémentation de serveur NTP

cotés client : nous avons implémentés le script suivant

```
url = "http ://10.1.60.31/dash/DASH-
JS/travauxEncours/Emulationnetem/big_buck_bunny_360p24.html"
CMD4='sudo ntpdate -u 10.1.60.59'
os.system(CMD4)
webbrowser.get('/user/bin/google-chrome').open(url)
```

Figure III-28 Script de lancement de client

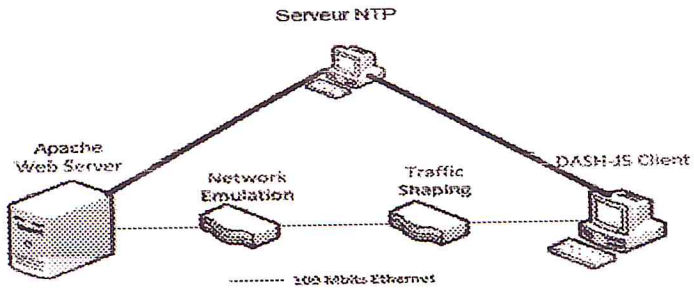


Figure III-29 Représentation de notre réseaux local

D'aprés l'étude que nous avons mené nous avons constaté plusieurs points parmi ces derniers une utilisation maximal de la bande passante disponible , et vu la variation de réseau cela pourrait conduire à la sélection du niveau trop élevé de qualité ce qui implique a des pics , soit une mauvaise utilisation de la bande passante avec une utilisation de la qualité inférieur et tout ca influence a l'effet visuel , en général ces deux algorithme fonctionnent bien avec les changement de débit sur le réseau sauf qu'il n'a pas male de nombre d'interruption plus la perte d'utilisation de bande passante dans qui se suit nous allons étudie le rôle des facteurs α et β plus le coefficient 0.9 et leurs effet sur la qualité de service

III.5 Conclusion

Dans ce chapitre nous avons présenté notre environnement d'expérimentation. Tout d'abord, nous avons parlé d'une façon globale de la plateforme réel et de ses quatre points essentiels (technique de streaming adaptatif DASH, le contenu (vidéo), le streaming , le réseaux de transport) . Par la suite, nous avons présenté les choix que nous avons retenus concernant ces quatre points afin de construire notre plateforme d'expérimentation . La mise en place de cette plateforme a nécessité l'utilisation conjointe d'outil de nature divers (réseau , vidéo , simulation) notons aussi que tous les outils utilisées sont libre d'accès tel que le FFMPEG pour convertir les fichiers vidéo d'un format à un autre et EvalVid pour évaluer la qualité de la vidéo , et pour implémenter le principe de DASH nous avons opté pour DASH-JS qui est un outil open source . Concernant l'émulation réseau nous avons utilisés le NetEm qui est aussi un outil libre et pour synchroniser les horloges de client et le serveur nous avons utilisés l'outil NTP tout ca a été introduit dans des scripts python et tout ces outils libre nous l'avons utilisés sur un system d'exploitation spécifique qui est ubuntu . Le chapitre suivant sera consacré à l'objectif principal de ce travail, à savoir l'implémentation d'un algorithme d'adaptation et l'évaluation de cet algorithme

Chapitre IV. EVALUATION

IV.1 Introduction

Le concept de la vidéo streaming adaptatif est basé sur l'idée d'adapter la bande passante requise par le flux vidéo pour le débit disponible sur le chemin de réseau à partir de la source de flux vers le client. En d'autre terme L'adaptation est de faire varier la qualité de la vidéo selon le débit canal. Une de ces approches ici est de diviser la vidéo en segments et coder chacun des segments dans plusieurs niveaux de qualité, appelée représentations. On se basant sur une estimation du débit disponible, le client peut demander des segments de différentes qualités en vue de faire face aux variations des conditions de réseau. L'algorithmique qui fait cette adaptation (améliorer la qualité de service) est un élément clé et l'un des défis majeurs dans les systèmes de diffusion adaptative.

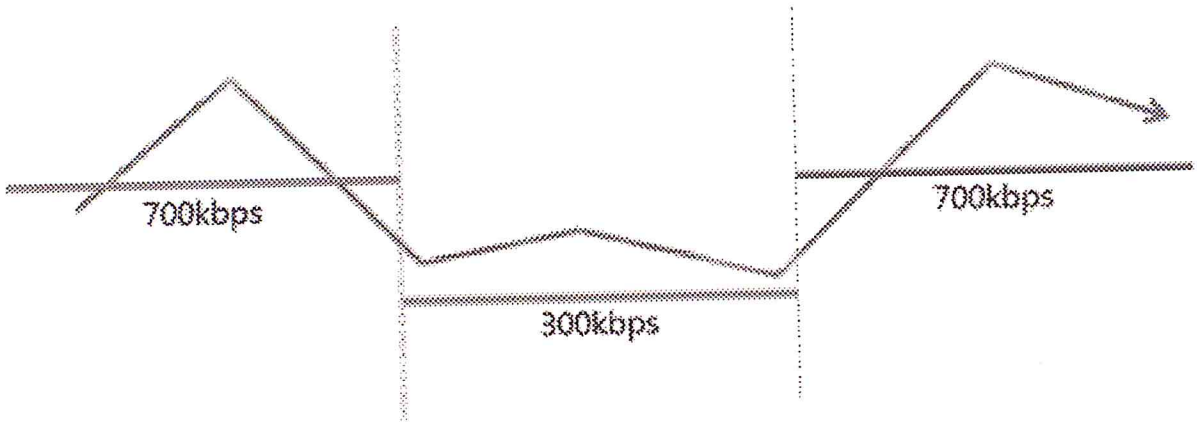


Figure IV-1 Principe du Streaming adaptatif

IV.2 Evaluation du streaming adaptatifs sous différentes contraintes

Nous avons fait quel que tests entre un client et un serveur sur une vidéo de 10mnt , résolution 640*360 , fps :24 , coder en plusieurs qualité [200 :200 :2200]Kbps et a l'aide d'un émulateur de réseau NetEm qui nous a crée un canal d'une bande passante choisi avec un tirage aléatoire [200,2200]kbps et une latence fixe de 10ms , un serveur NTP . Le résultat de ces test ont été exprimé dans les graphes ci-dessus ::

IV.2.1 Scénario 1: Scénario de référence (Original sans modification)

C'est notre point de repère, les résultats des autres scénarios vont être comparés par rapport à lui

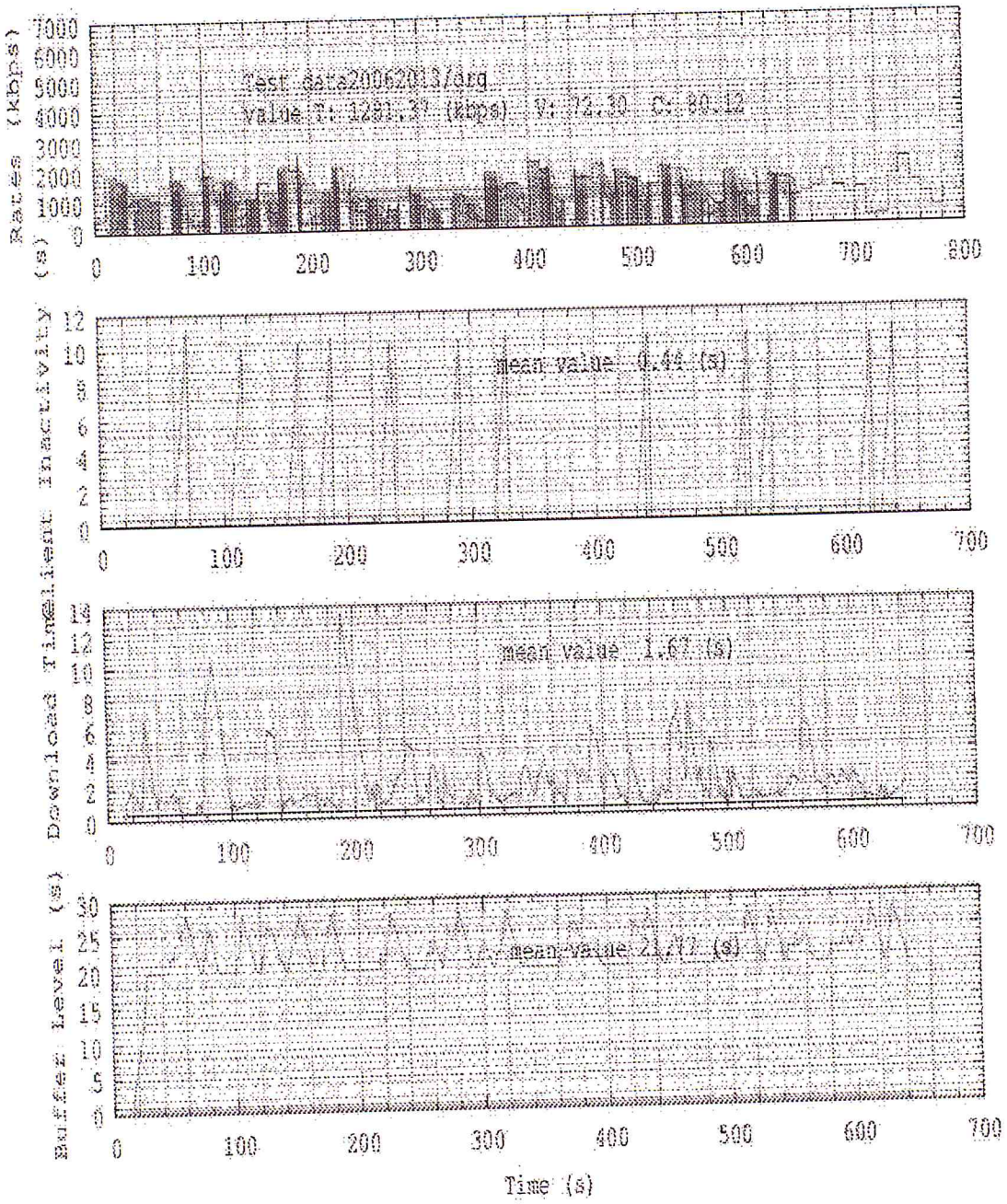


Figure IV-2 Graphes de représentation d'une vidéo

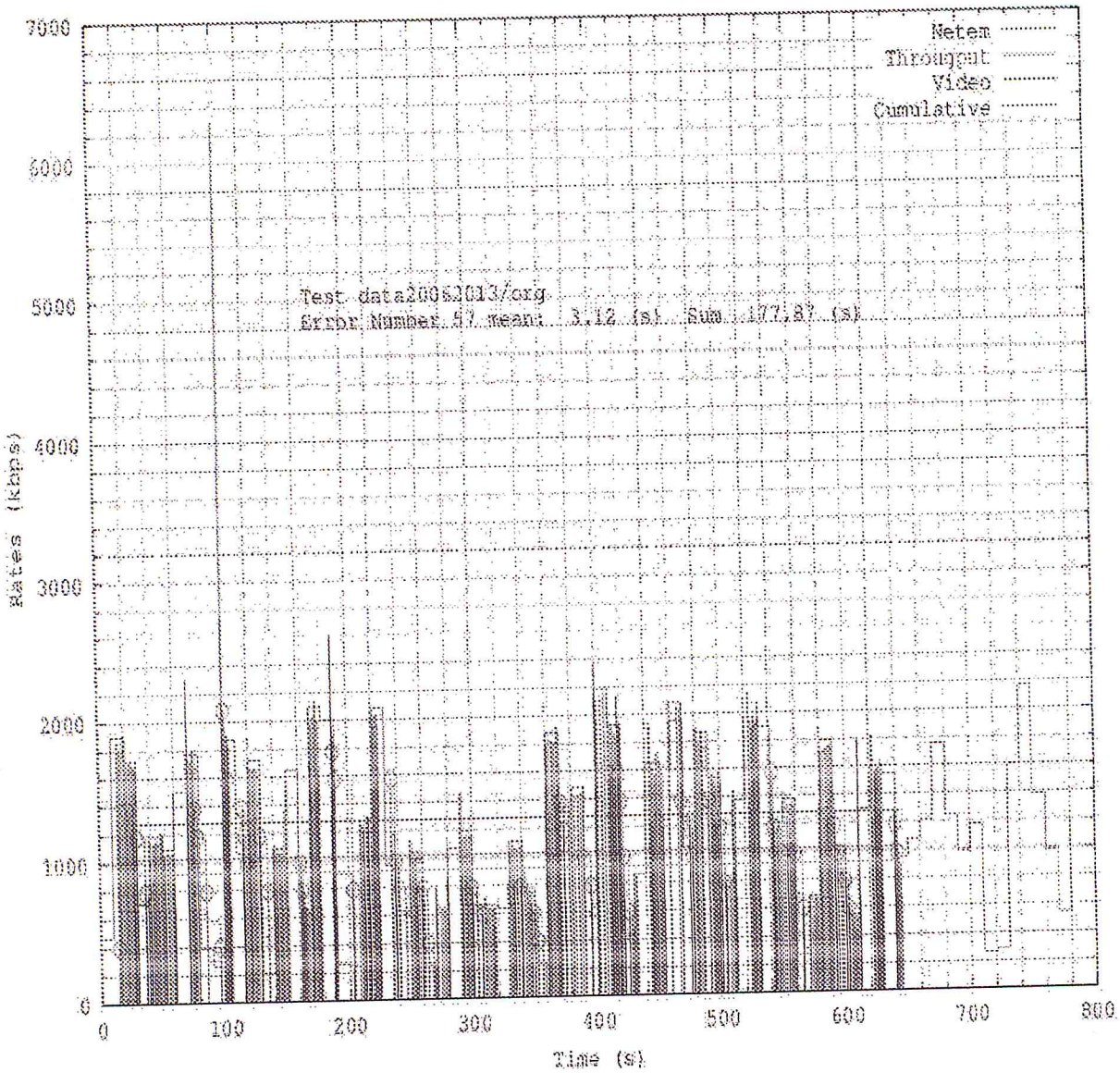


Figure IV-3 Graphe de représentation des débits entre client et le serveur

La Figure IV-2 est un ensemble de graphe , Le 1er représente le débit émulé , débit de la représentation vidéo et le débit cumulative en fonction du temps de simulation pour voir l'adaptation du canal avec le changement de débit de réseau

Le 2éme graphe représente l'inactivité du client c'est les périodes ou le client ne fait pas de téléchargement en fonction du temps de simulation il a une moyenne de 0.44s , il nous permet de connaitre les état du buffer (remplissage /vidage) et l'état de client lors des interruption

3éme graphe représente le temps de téléchargement de segment en fonction du temps de simulation il a une moyenne de 1.67s , il nous permet observer la relation entre le temps des téléchargements des segment et la duré des interruption

4éme graphe représente le remplissage de buffer level sa moyenne est 21.77s il nous permet de voir l'état de buffer au cour de la vidéo

La Figure IV-3 est un agrandissement du 1er graphe elle nous permet de voir les périodes au la qualité vidéo est plus grande que le débit canal (les erreurs) et la duré de ses erreurs

D'après ses graphes nous allons essayer d'améliorer l'utilisation de la bande passante (utilisation maximale de la bande passante) par la maximisation de la qualité vidéo et réduire le nombre et la duré des erreurs

IV.2.2 Scénario 2 : retrait du coefficient 0 .9

Dans cette expérience nous avons enlevé le coefficient (0.9) et nous allons voir sont effet sur la qualité de service , voila les résultats obtenu

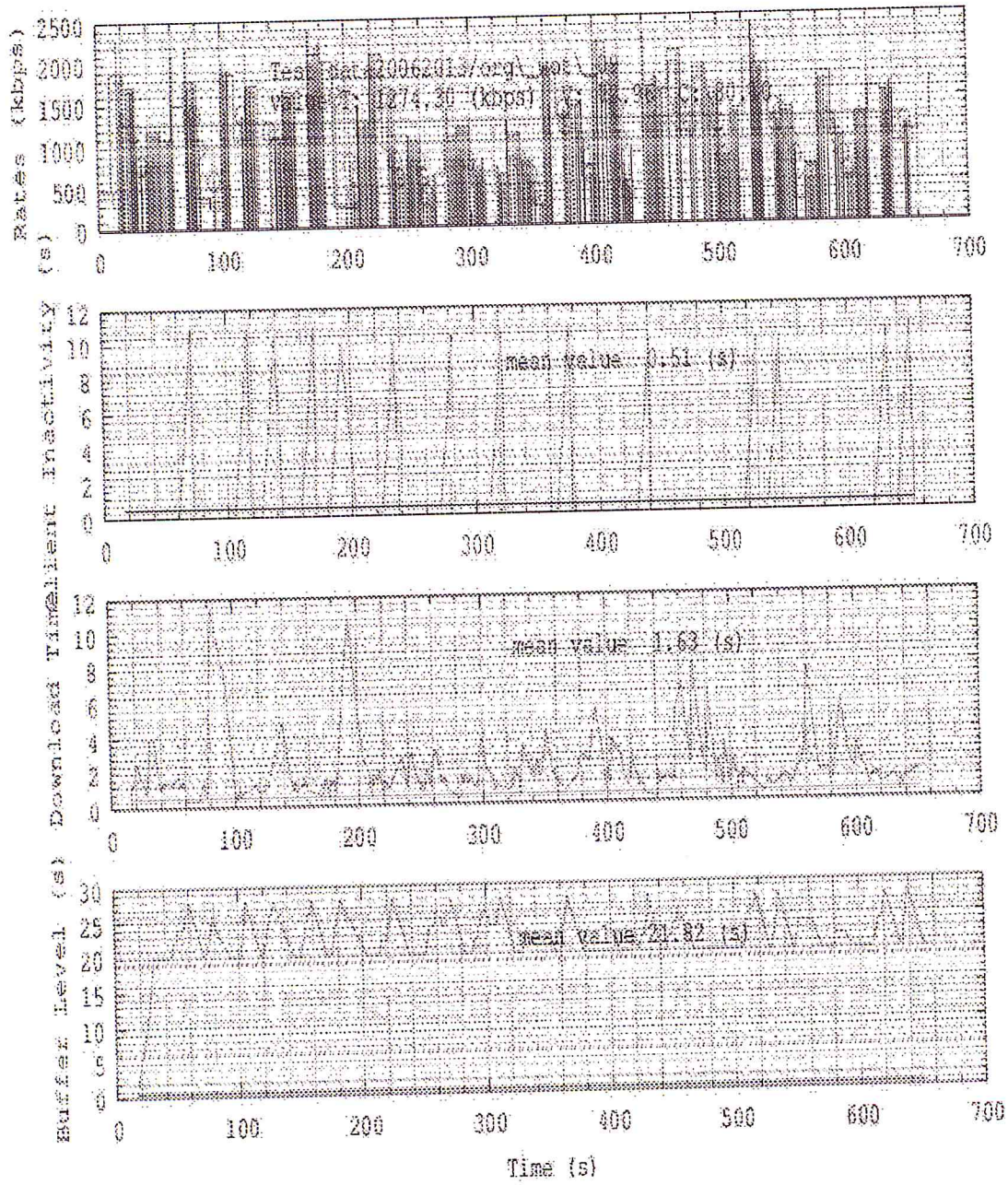


Figure IV-4 Graphes de représentation d'une vidéo

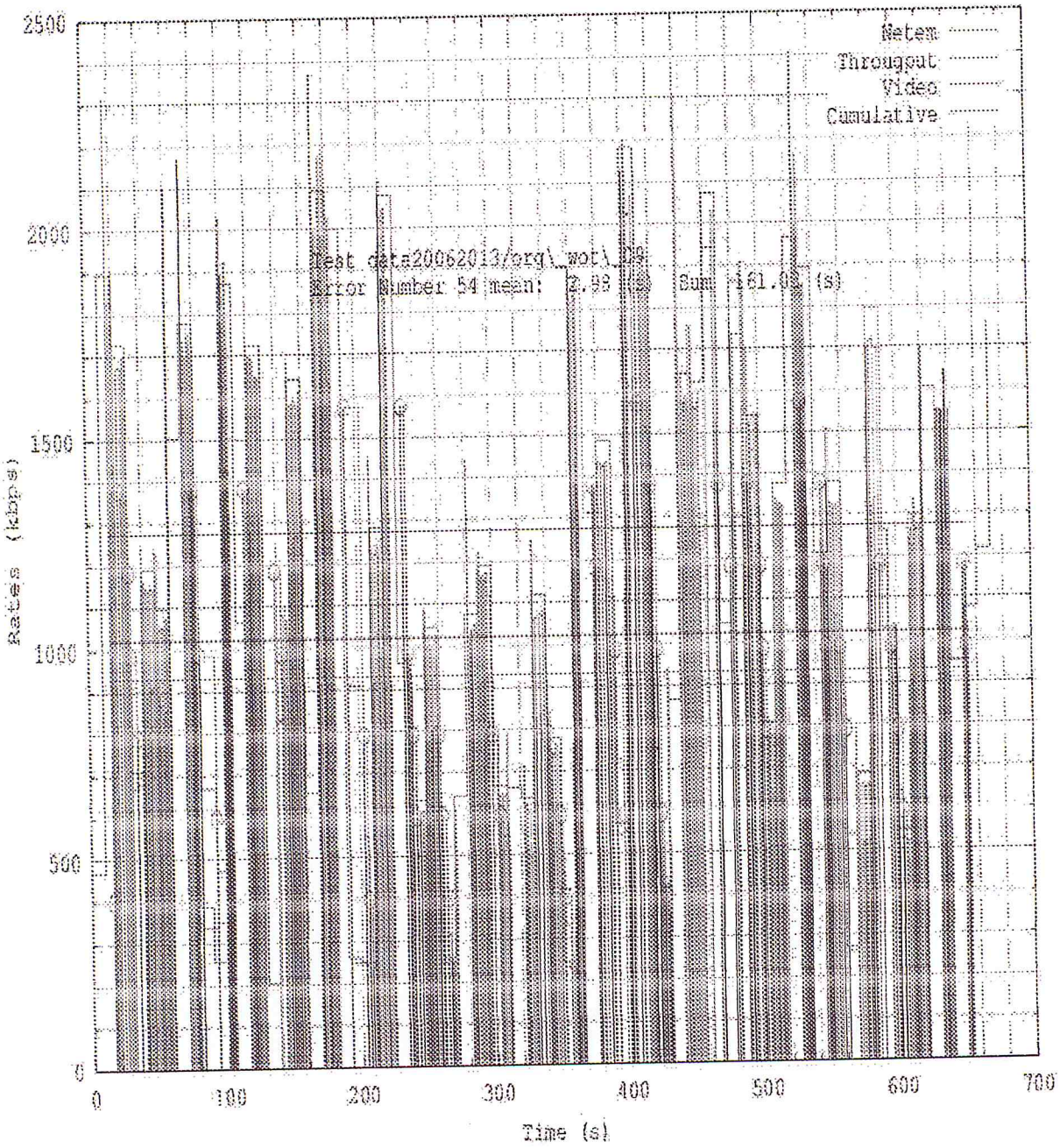


Figure IV-5 Graphe représentation d'une vidéo

IV.2.3 Scénario 3 : coefficient α β intervertis

dans cette expérience nous allons inverser les coefficient α β donc nous allons donner plus de priorité au passé , voici les résultat obtenu

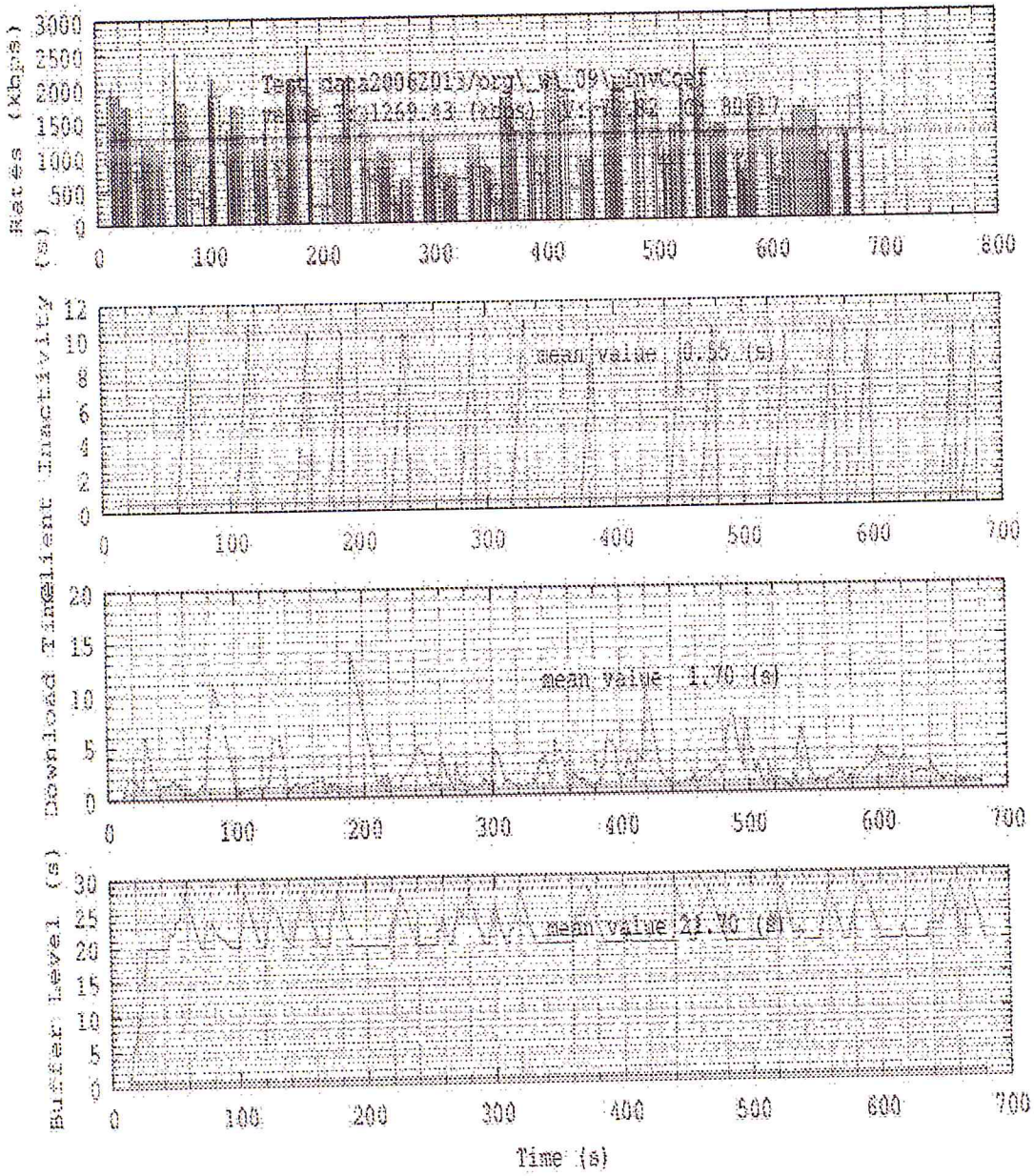


Figure IV-6 Graphe représentation d'une vidéo

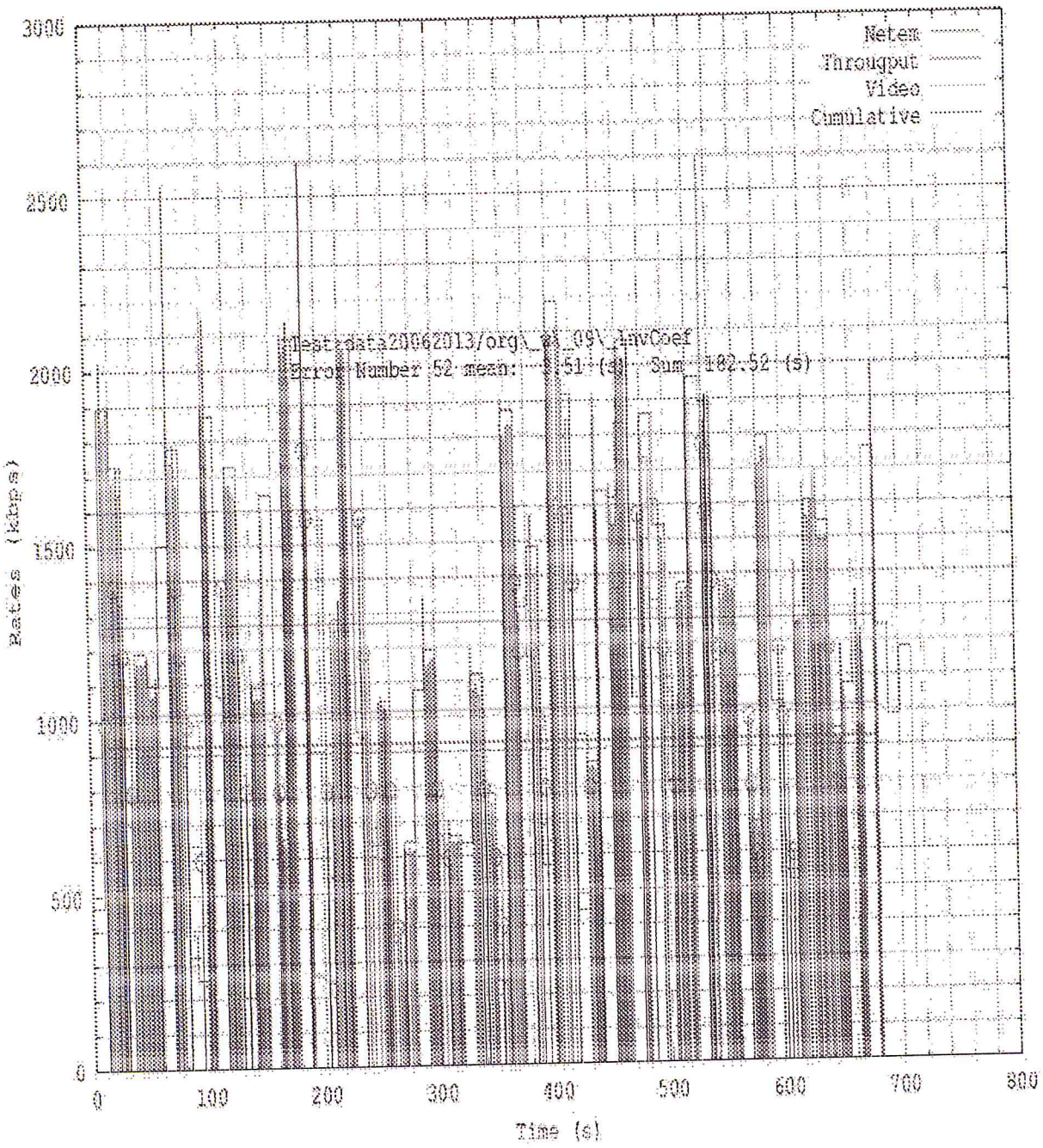


Figure IV-7 Graphe de représentation des débits entre client et serveur

IV.2.4 Scénario 4 : coefficient α β intervertis et retrait du coefficient 0.9

dans ce scénario nous avons laissé la priorité au passé et enlever le coefficient 0.9

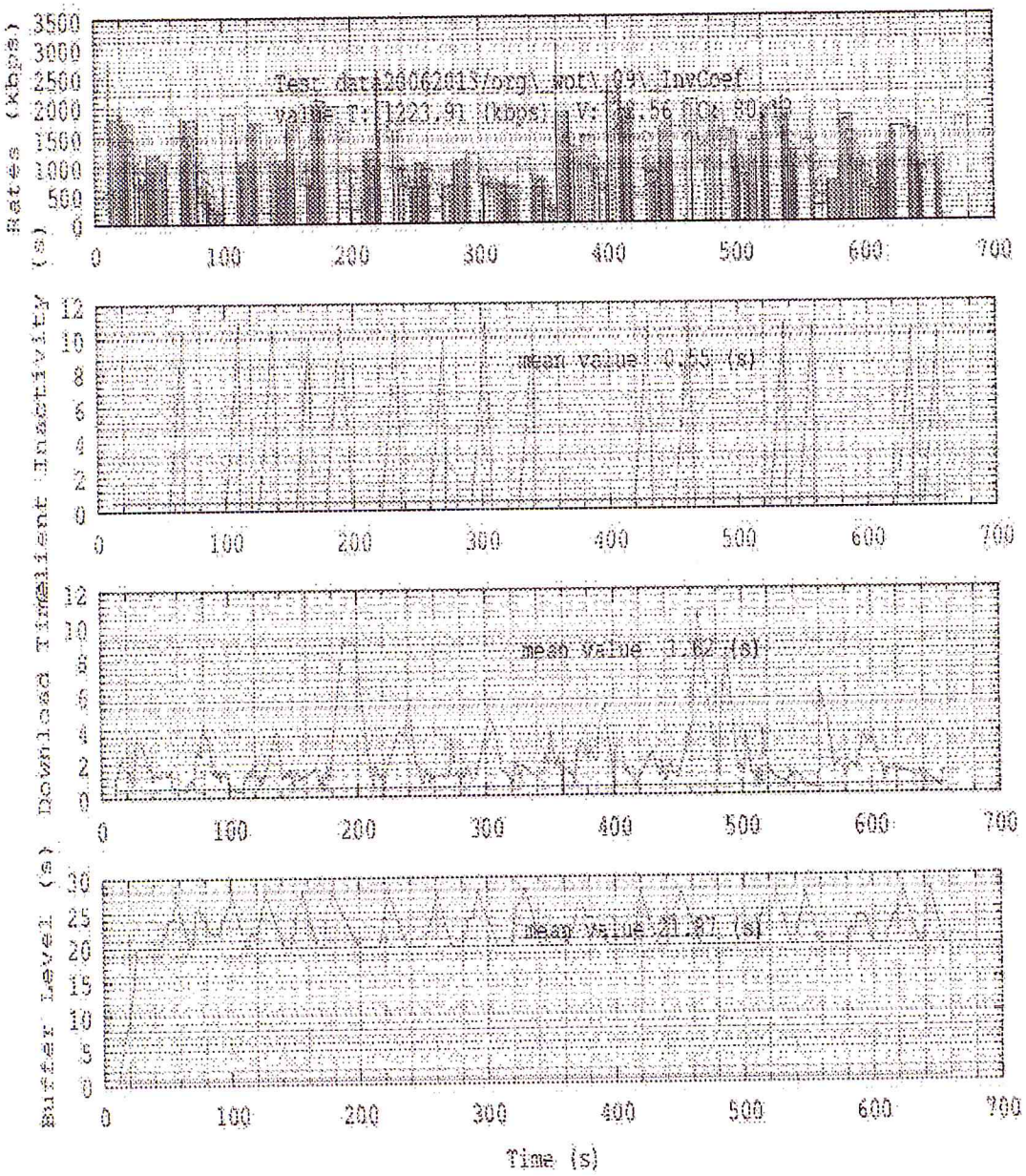


Figure IV-8 Graphe représentation d'une vidéo

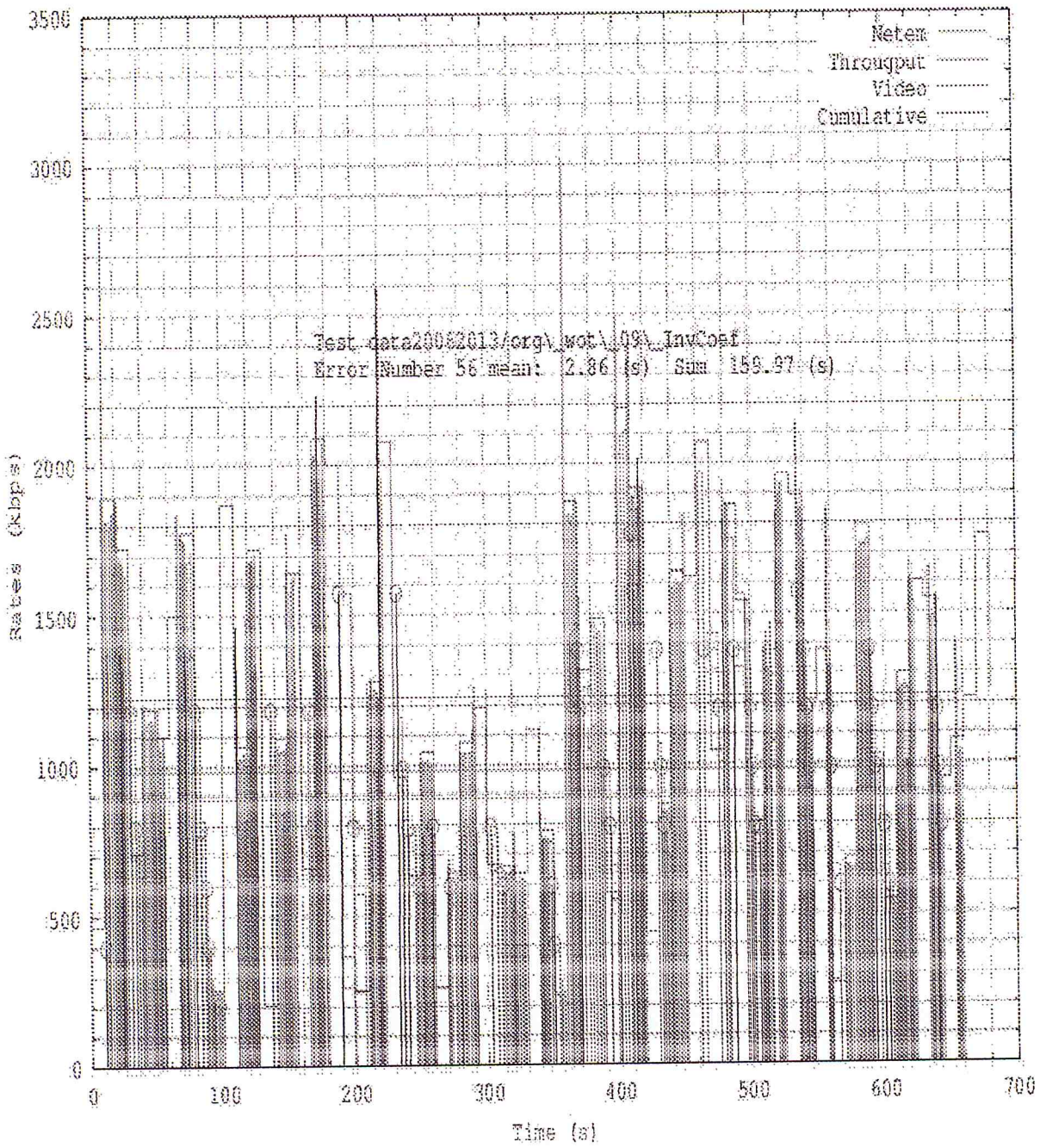


Figure IV-9 Graphe de représentation des débits entre client et serveur

IV.3 Comparaison entre les résultats obtenus

Pour faire la comparaison entre les résultat obtenu nous avons les classer tous dans un tableau a l'aide des critères d'évaluation suivantes

Débit canal : c'est la moyenne de la bande passante du canal en kbps

Représentation vidéo : c'est le débit de la représentation vidéo en % du débit canal

Bande passante cumulative : c'est la bande passante calculé avec la formule BC(n)= ((α *BC(n-1) + β*BM(n))/2)*0.9 et BM la bande passante mesuré en % du débit canal

inactivité du client : c'est les périodes ou le client ne fait pas de téléchargement en secondes

Temps de téléchargement : c'est le temps de téléchargement des segments en secondes

Niveau de buffer : c'est le niveau de remplissage de buffer en secondes

Nombre d'erreur : c'est le nombre de fois ou le débit vidéo est supérieur au débit canal (les interruption)

Moyenne d'erreur : c'est le temps moyen du la durée d'erreur en secondes

Somme d'erreur : c'est la durée totale des erreurs en secondes

Moyen	Scénario 1	Scénario 2	Scénario 3	Scénario 4
Débit canal	1281.37	1274.43	1259.43	1273.91
Représentation vidéo (%)	72.30	72.96	72.82	72.56
Bande passante cumulative (%)	80.12	80.10	80.17	80.13
Client inactive (s)	0.44	0.51	0.55	0.55
Temps de téléchargement (s)	1.67	1.63	1.70	1.62
Niveau de buffer (s)	21.77	21.82	21.70	21.87
Nbre d'erreurs	57	54	52	56
Moyen d'erreur (s)	3.12	2.98	3.51	2.86
Somme d'erreur (s)	177.87	161.03	182.52	159.97

Tableau 5 une comparaison entre les différents tests

IV.4 Analyse de tableau

D'après le Tableau 5 de comparaison et a l'aide des critères d'évaluation on voit que :

Le pourcentage de La représentation vidéo est un légèrement élever dans le scénario2, Le pourcentage de la bande passante cumulative est un légèrement élever dans le scénario 3 donc l'utilisation de la bande passante dans les scénarios est légèrement améliorer ,

Client inactif : dans les scénarios 3 et 4 la moyenne de la durées inactive est plus élever par port au scénario 1 et 2 sauf que le Temps de téléchargement des segments est moins élever dans le scénario 4, c'est ce qui explique la légère augmentation de la proportion de niveau de buffer.

Nombre d'erreur : on voit que le nombre d'erreurs est le moins élever dans le scénario 3 mais la durée moyenne des erreurs est la plus élevé par rapport aux autres scénarios.par contre c'est le contraire dans le scénario 4 .

Chaque changement de paramètre a son avantage et son inconvénient on remarque que le Scénario 4 (coefficient α β intervertis et retrait du coefficient 0.9) c'est le meilleure scénario la somme d'erreur est petite et elle n'a pas durée longue temps malgré que le nombre est élevé par rapport a les autres. Ainsi que l'utilisation de la bande passante est élever par apport a l'original même le pourcentage de la bande passante cumulative , la durée du client inactif, le temps de téléchargement et niveau de remplissage de buffer

IV.5 Conclusion

d'après les résultats obtenu dans les déférent scénario nous avons réussie à :

maximiser l'utilisation de la bande passante (scénario 2) par rapport a l'original

minimiser le nombre d'erreurs (scénario 3) par rapport a l'original

minimiser la duré moyen de l'erreur (scénario 4) par rapport a l'original

CONCLUSION GENERALE

Les travaux de recherche menés dans le cadre de ce mémoire s'articulent autour de deux axes principaux : l'amélioration de la qualité de vidéo, l'utilisation maximal de la bande passante a l'aide de l'algorithme d'adaptation. Dans les deux premières parties de ce mémoire nous avons traité les problématique et les objectifs de l'adaptation, la vidéo , le streaming .Dans la troisième partie, nous avons mit en place une plate forme d'expérimentation. une émulation de réseaux et des tests sur la simulation de délais et de la bande passante . Dans la quatrième partie Nous avons fait des modifications sur l'algorithme d'adaptation de la plateforme

A partir des résultats obtenu dans les tests nous avons observé d'autres phénomènes pouvons être aussi aidés a amélioré la qualité vidéo dans le streaming comme :

- Diminuer ou agrandir la taille des segments
- Diminuer ou agrandir la taille de buffer
- Générer plus de qualités vidéo (diminuer le pas)

et pour que ces résultats soient plus fiable il saurais mieux de les tester sur un pattern de téléchargement réel .

BIBLIOGRAPHIE

1. **(VNI), cisco visual network index.** *Cisco VNI :Forcast and Methodology , 2012- 2017.* USA : s.n., 29 mai 2013.

2. **Prof. M. Ghanbari, Prof. D. Crawford ,Dr. M. Fleury.** *Future Performance of Video Codecs.* University of Essex ,United Kingdom : s.n., 13 juillet 2006.

3. —. *Future Performance of Video Codecs.* University of Essex ,United Kingdom : s.n., 13 juillet 2006.

4. **itu, TELECOMMUNICATION standarization sector.** *Advanced video coding for generic audiovisual H264.* 03/2005.

5. On2 Technologies. *http://www.on2.com/.* [En ligne]

6. WebM An Open Web Media Project. [En ligne] 2013.

7. **WebM.** WebM An Open Web Media Project. [En ligne] 2013. http://www.webmproject.org.

8. **Kozamernik, Franc.** *Media Streaming over the internet.* octobre 2002.

9. **Ali C. Begen, Tankut Akgul, and Mark Baugher.** *Watching video over the web .* San Francisco - USA : s.n.

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

11. **Benjamin Rainer, Stefan Lederer, Christopher Müller, and Christian Timmerer.** *A Seamless Web Integration of Adaptive HTTP Streaming.* Boucharest - romania : s.n., aout 2012.

12. *Dynamic Adaptive Streaming over HTTP: Standards and Design Principles.* **Stockhammer, T.** San Francisco : ACM, 2011. ACM Multimedia Systems Conference (MMSys). 2011, February 23-25. San .

13. *3GPP TS.244. Transparent end-to-end packet switched streaming service (PSS), adaptive HTTP Streaming.* avril 2011.

14. **ISO/IEC, international standard.** *Information technology - Dynamic adaptive streaming over HTTP (DASH) Part 1 media presentation description .* switzerland : s.n., 01/04/2012.

15. *Microsoft Corporation. ISS Smooth Streaming Transport Protoco.* septembre 2011.

16. microsoft smooth streaming . *http://www.microsoft.com/downloads/en/details.aspx?displaylang=.* [En ligne]

17. **Zambelli, A.** *ISS smooth streaming technical overview.Microsoft Corporation.* spetembre 2009.

18. pple Corporation. Best Practices for Creating and Deploying HTTP Live Streaming. *http://developer.apple.com/library/ios/#technotes/tn2010/tn2224.html#.* [En ligne] avril 2011.

19. **Fecheyr-Lippens, A.** *review of HTTP Live Streaming.* janvier 2010.

20. **May, R. Pantos and W.** *HTTP Live Streaming, version 6.* mars 2011.

21. Adobe Systems Inc. Real-Time Messaging Protocol (RTMP) specification. *http://www.adobe.com/devnet/rtmp.html.* [En ligne] 2009.

22. **Hassoun, D.** *Dynamic streaming in FlashMedia Server 3.5: Overview of the new .*

23. **TS.234, 3GPP.** *Transparent end-to-end packet switched streaming service (PSS) Adaptive HTTP Streaming.* avril 2011.

24. **May, R. Pantos and W.** *HTTP Live Streaming, version 6. IETF Internet-Draft .* octobre 2011.

25. **Cisco.** TCP/IP Overview. *http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a008014f8a9.shtml.* [En ligne]

26. **Konstantin Miller, Emanuele Quacchio ,Gianluca Gennari, Adam Wolisz.** *Adaptation Algorithm for adaptive streaming over http .* Technische Universit¨at Berlin, Germany : s.n.

27. http://www-itec.uni-klu.ac.at/dash/. *http://www-itec.uni-klu.ac.at/dash/.* [En ligne]

28. **richard, lucas Nussbaum and Olivier.** *A comparative study of network link emulators.* Grenoble - France : s.n., 14 decembre 2008.

ANNEXE : OUTILS DE DEVELOPPEMENTS

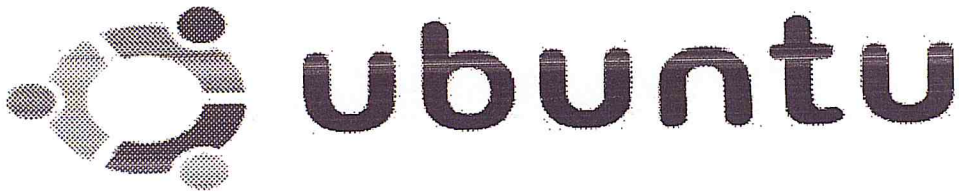
Pour mettre en place cette plateforme d'expérimentation nous avons utilisé plusieurs outils parmi lesquels on peut citer le système d'exploitation Ubuntu , Python et Javascript

IV.5.1 Environnement de programmation

Pour commencer notre travail il nous a fallu installer le system d'exploitation (Ubuntu) , et les langages de programmations suivants : python , java-script , html5

IV.5.2 Système d'exploitation Ubuntu

Ubuntu est un système d'exploitation GNU Linux, basé sur Debian, que chacun est libre d'utiliser, de modifier et de distribuer, aujourd'hui et pour toujours. Il est développé par une importante communauté d'utilisateurs et de développeurs qui s'efforcent de fournir et de maintenir les meilleurs logiciels libres et les formats ouverts

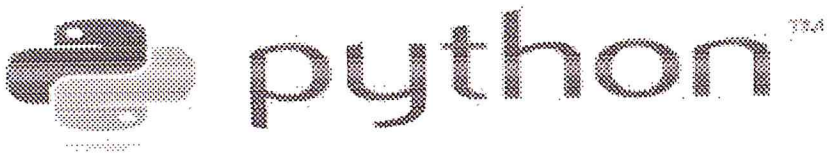


Ubuntu est caractérisé par une installation simple, une stabilité et une communauté très active. Il contient par défaut les dernières versions des logiciels Open-Source. Autre motif qui a influencé notre choix, c'est qu'il est utilisé par notre organisme d'accueil [32] . Nous allons utiliser un émulateur de réseaux netem qu'est déjà activée dans le noyau et la version actuelle de iproute2 est inclus

IV.5.2.1 Python :

Python est un interprétées, programmation orientée objet langage similaire à PERL , qui a gagné en popularité en raison de sa claire syntaxe et sa lisibilité. Python est dit être relativement facile à apprendre et portable, ce qui signifie ses déclarations peuvent être interprétées dans un certain nombre de système d'exploitation s, y compris UNIX les systèmes à base de Mac OS , MS-DOS , OS / 2 , et différentes versions de Microsoft Windows 98 . Python a été créé par Guido van Rossum, un ancien résident des Pays-Bas, dont le favori groupe de comédie à l'époque était Flying Circus des Monty Python. Le code source est

librement disponible et ouvert pour la modification et la réutilisation. Python dispose d'un nombre important d'utilisateurs.



Une caractéristique notable de Python est sa mise en retrait des déclarations de sources pour rendre le code plus facile à lire. Python offre dynamique type des données , ready-made classe et des interfaces à de nombreux appels et des bibliothèques système. Il peut être étendu en utilisant le C ou C + + langage.

Python peut être utilisé comme le script Active Server Page (ASP) la technologie de Microsoft.(exp : Le système de tableau de bord pour le Melbourne (Australie) Cricket Ground est écrit en Python. Z Publishing environnement d'objets, un site Web populaire serveur d'applications , est également écrit dans le langage Python) [33]. Se dernier a nous a aidé beaucoup surtout pour l'exécutions des commandes NetEm

IV.5.2.2 HTML5

Le HTML5 est la dernière version en date du langage de développement web HTML. Les spécifications définitives du HTML5 n'étaient pas encore fixées fin 2011.

Le HTML5 comprend de nouvelles balises et de nouveaux attributs pour les pages web et ouvre surtout de nouvelles possibilités de développement pour les sites mobiles. Le HTML5 entraîne des changements et apporte des nouveautés par rapport aux versions précédentes du langage. Parmi les différences notables, le HTML5 comprend de nouveaux éléments (audio, vidéo), ajoute des nouveaux attributs pour les balises et introduit des nouvelles API.

Ces mise a jour parfois nous causes des problème (les nouvelles API ne sont pas compatible avec Google Chrome)[34] , nous nous avons utilise le HTML5 pour appelé les fichier MPD via la commande : video = document.querySelector('video')

IV.5.2.3 JavaScript :

JavaScript est un langage de script orienté objet principalement utilisé dans les pages HTML. A l'opposé des langages serveurs (qui s'exécutent sur le site), JavaScript est exécuté sur l'ordinateur de l'internaute par le navigateur lui-même. Ainsi, ce langage permet une interaction avec l'utilisateur en fonction de ses actions (lors du passage de la souris au dessus d'un élément, du redimensionnement de la page...). La version standardisée de JavaScript est le ECMAScript.[35]

la partie client dans notre plateforme a été développer et modifiée par js

IV.5.3 Émulation de Réseaux

Il est communément connu que les réseaux sont imparfaits - privé ou public. Ils introduisent des retards et des erreurs. Ils rejettent les paquets. L'objectif principal de l'émulation de réseau est de créer un environnement dans lequel les utilisateurs peuvent connecter leurs appareils, applications, produits et / ou services et d'évaluer leurs performances, la stabilité ou la fonctionnalité contre scénarios de réseau du monde réel. Une fois testé dans un environnement contrôlé contre les conditions réelles du réseau, les utilisateurs peuvent avoir la certitude que le produit testé ne fonctionne pas comme prévu.

Et pour faire l'émulation de réseaux il éxiste plusieurs outils parmi les Dummynet , NISNet , et TC/Netem qui vont être détailler ci-dessus

IV.5.3.1 Outils d'Émulation de réseaux

il est en général très difficile de modifier la configuration du réseau des plates-formes expérimentales puisque cela nécessite en général de manipuler des câbles réseaux, des Switch ou des routeurs, domaines réservés aux administrateurs réseaux, pour faire cette modification d'une manière artificielle il existe plusieurs outils permettant d'émuler des caractéristiques réseaux différentes , par exemple Dummynet, NISTNet et TC/Netem

IV.5.3.2 NetEm

Il est communément connu que les réseaux sont imparfaits - privé ou public. Ils introduisent des retards et des erreurs. Ils descendent. Ils rejettent les paquets. L'objectif principal de l'émulation de réseau est de créer un environnement dans lequel les utilisateurs peuvent connecter leurs appareils, applications, produits et / ou services et d'évaluer leurs performances, la stabilité ou la fonctionnalité contre scénarios de réseau du monde réel. Une fois testé dans un environnement contrôlé contre les conditions réelles du réseau, les utilisateurs peuvent avoir la certitude que le produit testé ne fonctionne pas comme prévu.

Pour faire cette émulation nous avons utiliser le model netem des noyaux Linux , est un moyen de faire de la simulation Wan, Lan, ou tester la robustesse d'un système réseau vis à vis des perturbations présent sur un réseau. Celles-ci peuvent être les suivantes :

-Délai de transit

-Bande passante

-Perte de paquet

-Duplication de paquet

-Réarrangement de paquet

Les noyaux Linux possèdent un outil, le Trafic Control (Tc[60]), qui met en œuvre un ensemble de mécanismes afin de conditionner le trafic réseau. Tc est un programme utilisateur permettant de créer et d'associer des files à une interface de sortie. C'est un tout en un, il est utilisé pour installer divers types de files, associer des classes à ces files, mettre en place les filtres de classification.

Avec Tc, il est possible de définir des Queuing discipline (qdisc) qui permettent à l'utilisateur, grâce au mécanisme TBF (Token Bucket Filter) de séparer en queues les différents flux [38]

NOM :

tc - afficher / manipuler les paramètres de contrôle de la circulation

SYNOPSIS :

tc qdisc [add | change | replace | link] dev DEV [parent qdisc-id | root] [handele qdisc-id] qdisc [qdisc des paramètres spécifiques]

tc class [add | change | replace] dev DEV parent qdisc-id [classid class-id] qdisc [qdisc des paramètres spécifiques]

tc filter [add | change | replace] dev DEV [parent qdisc-id | root]

protocol priority filtertype [filtertype spécifique paramètres] flowid fluw-id

tc [FORMAT] qdisc [show | ls] dev DEV

tc [FORMAT] class show dev DEV

tc filtre show dev DEV

FORMAT: = {-s [tatistique] |-d etails [] |-r [aw] |-p [retty] | i [CE]}

IV.5.3.3 Traffic Control (Tc)

Tc est utilisé pour configurer le contrôle du trafic dans le noyau Linux. Trafic Contrôle se compose de ce qui suit:

• MISE EN FORME (Shaping)

Lorsque le trafic est en forme, son taux de transmission est sous contrôle. shaping peut-être plus que d'abaisser la bande passante disponible Il est également utilisé pour lisser les rafales du trafic pour un meilleure comportement du réseau. shaping se produit sur l'évacuation.

• CALENDRIER (SCHEDULING)

En planifiant la transmission de paquets, il est possible d'améliorer l'interactivité pour le trafic dont il a besoin tout en garantir la bande passante pour les transferts de masse. Réorganisation est également appelé priorités, et ne se produit que sur l'évacuation.

• POLICE (POLICING)

Où façonner traite de la transmission du trafic, de la police rapporte de trafic à l'arrivée. Maintien de l'ordre se fait donc sur la pénétration.

• CHUTE (DROPPING)

Le trafic dépassant une bande passante mis peut aussi être abandonné immédiatement à la fois sur l'entrée et sur la sortie.

Traitement du trafic est contrôlé par trois types d'objets: les qdisc,classes et les filtres.

• CLASSES

Certains qdisc peuvent contenir des classes, qui contiennent d'autres qdisc

le trafic peut alors être mise en queue dans l'un des qdiscs internes, qui sont à l'intérieur des classes. Lorsque le noyau essaie de défillé un paquet à partir d'une telle classe qdisc utile, il peut provenir de l'une des classes. A qdisc peut par exemple

la priorité à certains types de trafic en essayant de défillé certains classes avant les autres.

• FILTRES

Un filtre est utilisé par un qdisc classful pour déterminer dans quelle classe un paquet sera mis en file. Chaque fois qu'un paquet arrive à une classe avec des sous-classes, il doit être classé. Différentes méthodes peuvent être employées pour faire, l'un d'eux sont les filtres. Tous les filtres attachés à la classe sont appelés, jusqu'à ce que l'un d'eux revient avec un verdict. Si aucun verdict n'a été fait, d'autres critères peuvent être disponibles. Cela diffère par qdisc. Il est important de noter que les filtres résident dans de nouveaux qdisc ils sont pas maîtres de ce qui se passe.

• QDISCS

qdisc est l'abréviation de « queueing discipline :discipline files d'attente» il est élémentaire pour comprendre le traffic contrôle . Chaque fois que le noyau a besoin d'envoyer un paquet àune interface, elle est enfillé dans La file d'attente configurée pour cette interface.

Immédiatement après, le noyau essaie d'obtenir autant de paquets que possible de qdisc, pour leur donner de l'adaptateur de réseau conducteur.

Un simple Qdisc est celui «pfifo», qui ne fait pas de traitement du tout, et est un pur First In, First Out queue. Il n'a cependant stocker trafic lorsque l'interface réseau ne peut pas gérer momentanément.[39]

QDISC sans classe (classless)

Les qdisc sans classes sont les suivantes:

[p|b]fifo , pfifo_fast , red(Random Early Detection) , sfq(Stochastic Fairness Queueing) , tbf (The Token Bucket Filter)

nous nous avons utiliser le TBF (Token Bucket Filter)

IV.5.3.4 Le Token Bucket Filter

(TBF) est un gestionnaire de mise en file d'attente simple. Il ne fait que laisser passer les paquets entrants avec un débit n'excédant pas une limite fixée administrativement. L'envoi de courtes rafales de données avec un débit dépassant cette limite est cependant possible.

TBF est très précis, et peu gourmand du point de vue réseau et processeur. Considérez-le en premier si vous voulez simplement ralentir une interface !

L'implémentation TBF consiste en un tampon (seau), constamment rempli par des éléments virtuels d'information appelés jetons, avec un débit spécifique (débit de jeton). Le paramètre le plus important du tampon est sa taille, qui correspond au nombre de jetons qu'il peut stocker.

Chaque jeton entrant laisse sortir un paquet de données de la file d'attente de données et ce jeton est alors supprimé du seau. L'association de cet algorithme avec les deux flux de jetons et de données, nous conduit à trois scénarios possibles :

Les données arrivent dans TBF avec un débit EGAL au débit des jetons entrants. Dans ce cas, chaque paquet entrant a son jeton correspondant et passe la file d'attente sans délai.

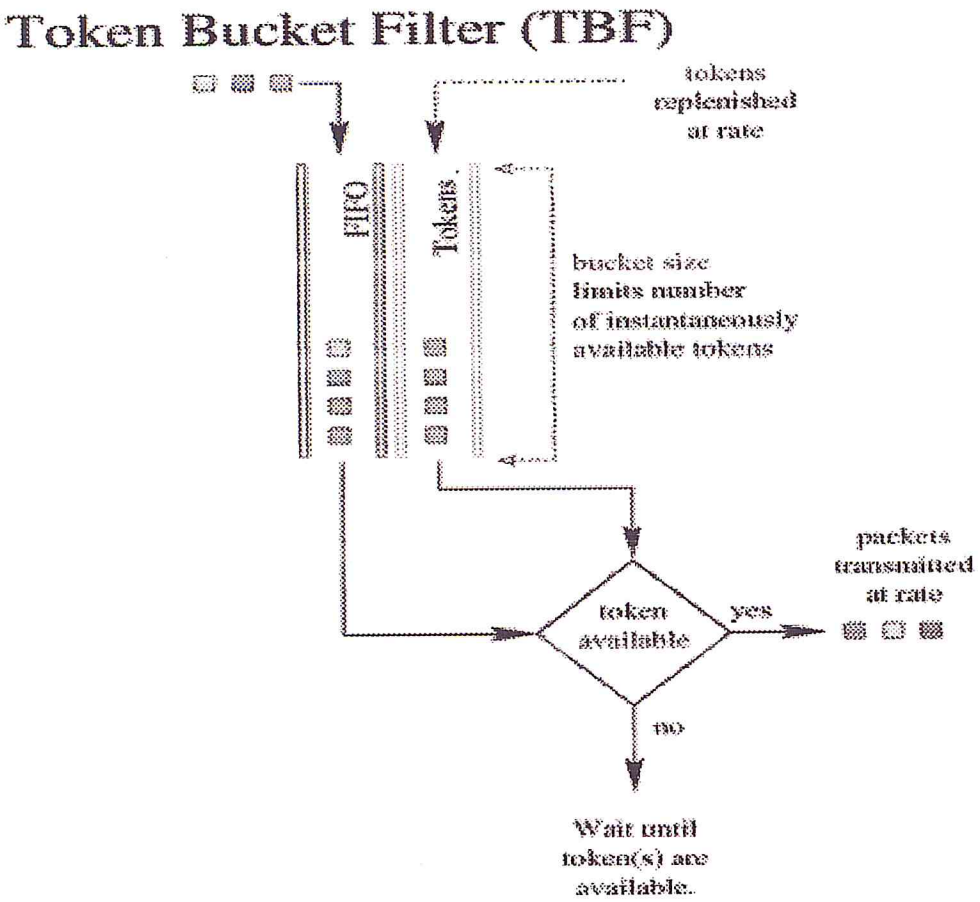
Les données arrivent dans TBF avec un débit PLUS PETIT que le débit des jetons. Seule une partie des jetons est supprimée au moment où les paquets de données sortent de la file d'attente, de sorte que les jetons s'accumulent jusqu'à atteindre la taille du tampon. Les jetons libres peuvent être utilisés pour envoyer des données avec un débit supérieur au débit des jetons standard, si de courtes rafales de données arrivent.

Les données arrivent dans TBF avec un débit PLUS GRAND que le débit des jetons. Ceci signifie que le seau sera bientôt dépourvu de jetons, ce qui provoque l'arrêt de TBF pendant un moment. Ceci s'appelle « une situation de dépassement de limite » (overlimit situation). Si les paquets continuent à arriver, ils commenceront à être éliminés.

Le dernier scénario est très important, car il autorise la mise en forme administrative de la bande passante disponible pour les données traversant le filtre.

L'accumulation de jetons autorise l'émission de courtes rafales de données sans perte en situation de dépassement de limite, mais toute surcharge prolongée causera systématiquement le retard des paquets, puis leur rejet.

Notez que, dans l'implémentation réelle, les jetons correspondent à des octets, et non des paquets.



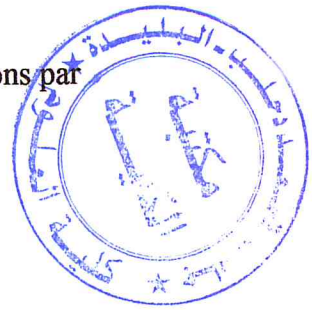
IV.5.3.5 Paramètres et usage :

limit ou latence

Limit est le nombre d'octets qui peuvent être mis en file d'attente en attendant la disponibilité de jetons. Vous pouvez également indiquer ceci d'une autre manière en configurant le paramètre latency, qui spécifie le temps maximal pendant lequel un paquet peut rester dans TBF. Ce dernier paramètre prend en compte la taille du seau, le débit, et s'il est configuré, le débit de crête (peakrate).

burst/buffer/maxburst

Taille du seau, en octets. C'est la quantité maximale, en octets, de jetons dont on disposera simultanément. En général, plus les débits de mise en forme sont importants, plus le tampon doit être grand. Pour 10 Mbit/s sur plateforme Intel, vous avez besoin d'un tampon d'au moins 10 kilo-octets si vous voulez atteindre la limitation configurée !



Si votre tampon est trop petit, les paquets pourront être rejetés car il arrive plus de jetons par top d'horloge que ne peut en contenir le tampon.

On peut aussi calculer le burst

burst = bitRate / fréquence d'interruption

on calcule la fréquence d'interruption via la commande :

cat /proc/interrupts |grep LOC; sleep 1;cat /proc/interrupts |grep LOC

```
alilo@alilo-HP-G72-Notebook-PC:~$ cat /proc/interrupts |grep LOC; sleep 1;cat /p
roc/interrupts |grep LOC
LOC:    1822871   1789302   Local timer interrupts                1
LOC:    1822950   1789400   Local timer interrupts                2
```

Figure IV-10 La commande pour la fréquence d'interruption

après on fait la soustraction 2-1 et on obtiendra les valeurs d'interruption , ces valeurs sont presque égaux on prend la plus grande

Il existe d'autre parametres comme :

mpu : l'unité minimale de paquet (Minimu Packet Unit) détermine le nombre minimal de jetons à utiliser pour un paquet

rate : paramètre de la vitesse

peakrate : débit crête, utilisé pour spécifier la vitesse à laquelle la seau est autorisé à se vider

mtu : Maximal Transfert Unit [39]