Treball de Fi de Grau

**Enginyeria en Sistemes Audiovisuals**

# Desenvolupament d'una plataforma per a distribució de Vídeo Sota Demanda (VoD)


**Autor**: Nil Torrents González

**Director**: Juan José Alins Delgado

**Codirector**: Juan Mon González



**Convocatòria**: Setembre 2021




UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

# Resum

Veient que any rere any hi ha un creixement exponencial en el consum de contingut audiovisual a través d'Internet i que els usuaris són cada vegada més exigents en relació a la qualitat de vídeo, el temps d'espera o el dispositiu de visualització, s'ha decidit realitzar aquest treball per veure quina és la configuració més òptima per al desenvolupament d'una plataforma per distribuir vídeo sota demanda.

L'any 2002 va aparèixer per primera vegada el concepte de taxa de bits adaptativa i és una tècnica que permet ajustar el flux del contingut audiovisual a cada usuari. Actualment totes les empreses dedicades a aquest sector utilitzen aquest mètode juntament amb estàndards com per exemple HLS o DASH ja que millora, d'entre altres, l'experiència d'usuari de forma notable.

En aquest treball s'ha realitzat un estudi de l'estàndard i del còdec més habituals per a vídeo sota demanda, per després crear una plataforma de distribució de contingut multimèdia utilitzant programari de codi obert. Això servirà per avaluar les codificacions i els algoritmes realitzats prèviament per veure com actua cadascun quan hi ha un estat de xarxa variable i per tant s'exigeix canvis de resolució i de taxa de bits.

La plataforma està formada per tots els elements d'un sistema multimèdia bàsic. S'hi troba en primer lloc el servidor, a continuació un mètode per enviar els recursos al destinatari, és a dir el mètode de distribució i per últim un client que s'encarrega de realitzar les peticions.

Per comprovar el correcte funcionament de la plataforma desenvolupada en el treball, s'ha fet una avaluació comparant la part teòrica amb la part pràctica per veure si es compleixen les premisses i aportar les millores necessàries per al treball futur.

# Abstract

Considering that year after year there is an exponential growth in the consumption of audiovisual content via the Internet and that users are increasingly demanding in terms of video quality, waiting time or display device, it has been decided to carry out this work to study the most optimal configuration for the development of a platform to distribute video on demand.

In 2002, the concept of adaptive bit rate first appeared and is a technique that allows you to adjust the flow of audiovisual content to each user. Currently all companies in this sector use this method together with standards such as HLS or DASH as it improves, among other things, the user experience significantly.

In this work, a study of the most common standard and codec for video on demand has been carried out, to then create a multimedia content distribution platform using open source software. This will be used to evaluate the encodings and algorithms previously performed to see how each one acts when there is a variable network state and therefore changes in resolution and bit rate are required.

The platform consists of all the elements of a basic multimedia system. First there is the server, then a method for sending the resources to the recipient, that is, the distribution method, and finally a client that handles the requests.

In order to verify the correct functioning of the developed platform, an evaluation has been made comparing the theoretical part with the practical part to see if the premises are fulfilled and to contribute the necessary improvements for the future work.

# Índex

# Llista d'abreviatures

ABR = Adaptative BitRate

AVC = Advanced Video Coding

CBR = Constant Bit Rate

CDN = Content Delivery Network

DASH = Dynamic Adaptative Streaming over HTTP

DCT = Discrete Cosine Transform

DoS = Denial Of Service

DPB = Decoded Picture Buffer

EdVP = Educative Video Platform

EVP = Empresarial Video Platform

FPS = Frames Per Second

GOP = Group of Pictures

HBO = Home Box Office

HDCP = High-bandwidth Digital Content Protection

HDR = High Dynamic Range

HLS = HTTP Live Streaming

HTML = HyperText Markup Language

HTTP = Hypertext Transform Protocol

IP = Internet Protocol

IPTV = Internet Protocol Television

IDR = Insatantaneous Decoder Refresh

KBPS = KiloBit Per Second

LAN = Local Area Network

LRU = Least Recently Used

M3U = Mpeg Version 3 URL

MBPS = Megabit Per Second

MBR = Maximum Bit Rate

MIME = Multipurpose Internal Mail Extension

MPEG = Moving Picture Experts Group

OTT = Over The Top

OVP = Online Video Platform

PSNR = Peak Signal-to-Noise Ratio

QoS = Quality of Service

SDL = Simple DirectMedia Layer

URL =Uniform Resource Locator

UTF-8 = 8-bit Unicode Transformation Format

UVLC = Universal Variable Length Code

VBR = Variable Bit Rate

VLC = Video LAN Client

VoD = Video on Demand

Wi-Fi = Wireless Fidelity

WLAN = Wireless Local Area Network

# Llista de taules

# Llista de figures

# Llista de llistes

# 1. Introducció

## 1.1 Objectius

L'objectiu d'aquest treball de fi de grau és el desenvolupament i creació d'una plataforma de VoD (Video on Demand), que ens permeti avaluar el conjunt de paràmetres òptims per a la transcodificació i distribució apropiada d'una copia màster d'una seqüència de vídeo.

Aquesta plataforma es centrarà en la distribució de vídeo sota demanda a través d'Internet en serveis OTT (Over The Top), per la qual cosa s'hauran d'estudiar les tecnologies utilitzades en aquest tipus de servei: L'arquitectura de la plataforma de distribució, l'ABR (Adaptive Bit Rate), el protocol HLS i el codificador de vídeo utilitzat.

Aquesta plataforma ha de permetre veure com es comporta un sistema VoD en funció de la codificació utilitzada i la variació del comportament de la xarxa.

Seguint la tècnica de "bitrate adaptatiu" (ABR) que fa servir l'HLS, s'haurà de seleccionar i codificar un conjunt de fluxos de diferents nivells de qualitat i bitrate (taxa de bits) que permetin una adaptació adequada al ampla de banda disponible entre el servidor i el client.

S'ha d'instal·lar i configurar un servidor de VoD per dur a terme les proves. Una vegada estigui el servidor llest, es realitzaran proves des d'un client extern, on variant les condicions de la xarxa, s'estudiaran els diferents escenaris per a la seva distribució a través d'internet en serveis OTT.

D'aquí es podrà obtenir les conclusions basant-se en els resultats obtinguts per als diferents processos seguits i l'algoritme utilitzat en cadascun de manera que es pugui afirmar quin funciona millor segons l'estat de la xarxa de cada client.

## 1.2 Requisits i especificacions

Per el desenvolupament de la plataforma HTTP (Hypertext Transform Protocol) es farà us de les eines de programari lliure que hi ha actualment com per exemple FFmpeg i Nginx, juntament amb els mòduls que hi ha per poder iniciar la plataforma. També es farà ús d'un dels estàndards de codificació oberts més utilitzats, el qual s'explicarà més endavant en aquest treball, l'H.264.

No hi ha per tant, cap requisit econòmic extern per assolir l'objectiu d'aquest projecte, només requisits tècnics, doncs per tal de fer les codificacions i les configuracions del servidor de manera fluida es recomana utilitzar un ordinador amb un processador Intel Core i5 de 10a generació o equivalent, amb un sistema operatiu Linux o en el seu defecte d'una maquina virtual per instal·lar una distribució de Linux.

La única condició econòmica és el preu per hora personal i professional. És a dir, si s'ha fet un total de 400 hores amb una taxa de 10 euros/hora, el cost econòmic total és 4000 euros.

# 2. Estat de l'art

Com ja s'ha comentat, un del objectius del projecte es instal·lar i configurar un servidor de VoD per oferir serveis de distribució de vídeo OTT. Però abans de tot, necessitem veure què és OTT.

## 2.1 Què és OTT

Les plataformes i aplicacions over-the-top, o plataformes OTT, són aplicacions que ofereixen contingut de vídeo a través d'Internet en lloc de televisió per cable o satèl·lit (la forma tradicional). Permeten transmetre instantàniament vídeos en dispositius mòbils, web i televisors que utilitzen dispositius com Chromecast, AppleTV, Amazon TV o SmartTVs [1].

Per consumir aquest servei calen dos requisits: comptar amb un dispositiu compatible i una connexió a Internet. Per això, generalment, aquestes infraestructures tenen un cost inferior als tradicionals sistemes de difusió, sobretot pel fet que no han de pagar quotes als operadors per utilitzar les seves xarxes ni han de desenvolupar nous circuits de distribució. El finançament dels serveis s'acostuma a obtenir gràcies a la publicitat o per sistemes de subscripció.

A més, aquest terme, comprèn una varietat de serveis audiovisuals com ja s'ha mencionat però també serveis de comunicació, informàtica o emmagatzematge, com per exemple trucades de veu per IP, missatgeria instantània i aplicacions web.

El serveis més rellevants utilitzats sobre OTT són els següents:

- Audiovisuals: Netflix, HBO (Home Box Office), Amazon Prime, Youtube, Apple TV, Rakuten TV.
- Comunicacions: Skype, Whatsapp, Snapchat, Google Hangouts.
- Apps i emmagatzematge: Google, Dropbox, Amazon, OneDrive

En els últims anys, el serveis de vídeo en streaming que treballen sobre OTT han adoptat un paper molt important per a moltes empreses, i el creixement dels usuaris que utilitzen aquests serveis ha incrementat exponencialment.

Segons el diari El País ([2]), a l'any 2019 s'estimava que la quantitat d'usuaris de VoD (Video on Demmand) al 2020 superessin el número de 400 milions, tenint en compte que al 2018 ja hi havia un total de 283 milions d'usuaris en les plataformes principals (Netflix, HBO i Amazon Prime). Però, un article publicat al març de 2020 ([3]), confirma que ja s'ha superat la xifra de 1.100 milions d'usuaris, superant àmpliament el valor estimat un any abans.

És per això, que totes les empreses dedicades a aquest sector, pateixen problemes de fragmentació causats pel gran nombre de dispositius als que han de servir i quins estàndards admeten. Però, el problema més gran, és com gestionar de manera eficient tots els paquets de fluxos de vídeo que tenen emmagatzemats en un format determinat i que es transmeten utilitzant una sèrie de protocols. Ja que, al existir dispositius que accepten un número limitat de protocols, és difícil per a les empreses gestionar aquesta gran quantitat de formats de paquets i el cost de tot el seu emmagatzematge.

L'objectiu dels serveis de VoD és oferir als seus clients, els vídeos que sol·liciten, en qualsevol moment, sense temps d'espera i amb la màxima qualitat possible. Aquest projecte s'encarregarà d'avaluar el protocols actuals per la distribució de vídeos a través d'Internet en serveis OTT i de posar-los en pràctica dins una plataforma via HTTP. A partir dels resultats obtinguts, es buscarà la millor manera d'optimitzar els vídeos i de fer-los adaptatius a cada usuari per oferir la millor experiència multimèdia.

Els sistemes de vídeo sota demanda tenen molts paràmetres que es poden configurar per tal d'optimitzar el seu funcionament i així maximitzar el seu benefici. Dos dels paràmetres més importants i que s'han de tenir en compte, són els següents:

- Ample de banda

L'ample de banda es mesura com la quantitat de dades que es poden transferir entre dos punts d'una xarxa en un temps específic. Normalment, l'ample de banda es mesura en bits per segon (bps) i s'expressa com una taxa de bits. Com més gran sigui aquest valor, més dades podran ser enviades i rebudes i per tant més informació arribarà al seu destí en menys temps.

Un dels problemes que presenta una xarxa com Internet, es que no hi ha qualitat de servei (QoS). Això vol dir que tots els paquets de dades que corren a través d'internet són tractats amb la mateixa prioritat. Des del punt de vista d'una connexió extrem a extrem, la manca de QoS es veu reflectida com una variació de l'ample de banda (principalment, degut als episodis de congestió del dispositius d'encaminament de la xarxa). Els serveis de VoD sobre Internet (OTT) es veuen afectats per aquesta variació d'ample de banda. Si l'ample de banda baixa per sota d'uns certs llindars entre el servidor i el client, el vídeo no es reproduirà correctament.

Però, per poder ajudar al client, hi ha algunes configuracions que optimitzen el procés i permeten estalviar ample de banda. Com per exemple l'elecció del codificador de vídeo, els paràmetres de codificació o tècniques com l'ABR (Adaptative BitRate), que més endavant s'explicarà com funciona cadascun d'aquests paràmetres.

- Emmagatzematge

L'altre paràmetre de vital importància es l'emmagatzematge, és a dir la mida dels fitxers que es carreguen al servidor i que són necessaris ser descarregats per part del client. És evident que aquest problema incideix de forma directa al ens explotador del servei de VoD. Hom pot pensar, per exemple, en Netflix i la gran quantitat de títols emmagatzemats per ser distribuïts. Tenint en compte que les tècniques d'ABR passen per codificar varies copies de diferent qualitat del mateix títol, és important que donat un cert nivell de qualitat d'una còpia, aquesta ocupi el mínim espai possible.

Per aconseguir optimitzar aquest procés i que tot ocupi el mínim d'espai, hi ha tècniques de codificació que ho permeten fer. Per exemple, limitar el bitrate dels vídeos en els paràmetres de codificació disminueix la mida del fitxer, també aspectes com limitar el CRF (Constant Rate Factor) o donar uns valors de resolució més baixos ajuden a obtenir fitxers més petits a l'hora de codificar.

També, l'elecció del codificador és molt important, doncs pot ser el factor que marqui més la diferencia. En aquest treball s'ha utilitzat l'H.264, que com es podrà veure més endavant, permet estalviar molt ample de banda a la vegada que incrementa la qualitat del vídeo considerablement sense haver d'augmentar l'espai que ocupa en la memòria.

En quan a la tecnologia dels servidors de vídeo a la carta, sol incorporar-se una xarxa de distribució de contingut (CDN) que emmagatzema el contingut de vídeo en diversos servidors molt dispersos, per tal de gestionar de manera eficaç els volums de trànsit elevats que es generen.

Aquesta xarxa de distribució millora el rendiment dels servidors de VoD mitjançant un procés anomenat emmagatzematge en memòria cau HTTP, que consisteix en la creació de rèpliques del contingut a mesura que es distribueix a través de la xarxa i l'emmagatzematge temporal de les còpies. Quan un usuari envia una sol·licitud de vídeo, el servidor CDN més proper a la ubicació del client respon a la sol·licitud i envia el contingut desitjat des de la seva memòria cau. Això redueix la latència per a l'usuari final, així com la càrrega sobre els servidors d'origen i la xarxa principal de l'operador [4].

## 2.2 Empreses que fan servir VoD sobre OTT

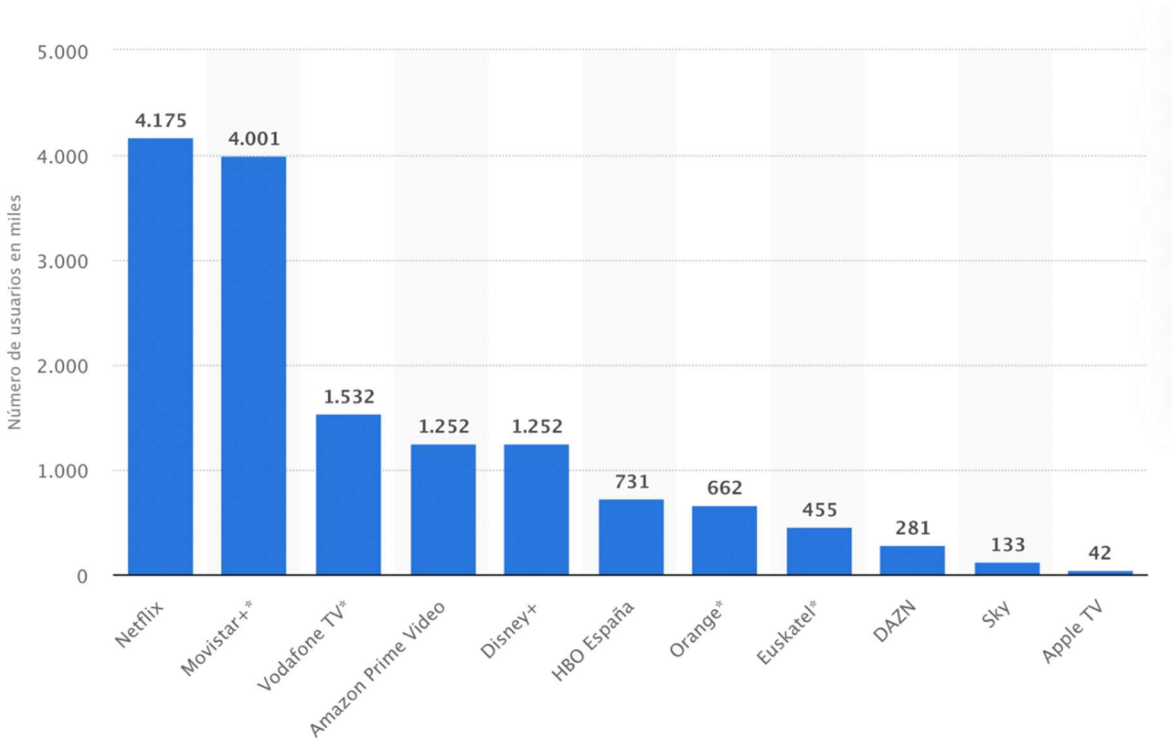



*Figura 2.1: Número d'usuaris en empreses de VoD sobre OTT l'any 2020. Imatge original de Statista* [5].

La Figura 2.1 representa com el número d'usuaris que fan servir plataformes de Vídeo sota demanda sobre OTT és molt elevat. I, que les dues que estan més a l'alça són Netflix i Movistar TV.

Aquests números han tingut un gran creixement en els últims anys i sobretot en el darrer any, on degut a la pandèmia, moltes persones han decidit contractar algun d'aquests serveis per passar un temps més entretingut des de casa [6].

Els vídeos sota demanda s'assemblen força al sistema tradicional de creació i compartició de contingut. Es crea un videoclip, es penja en una plataforma de vídeo i es fa disponible perquè el puguin veure els espectadors. La transmissió de VoD ha transformat la manera de veure i emetre els vídeos.

Una de les avantatges més notables de VoD en OTT per sobre del vídeo en directe és que hi ha més temps disponible per a la producció i per tant apareixen menys errors en el resultat final. Com que els VoD es produeixen prèviament, sempre hi ha lloc per a improvisacions. Tot i que el procés de producció pot trigar més, permet als creadors de contingut polir el seu treball per obtenir els millors resultats. Passa el mateix amb la post-producció, hi ha més temps per editar els vídeos i els resultats són molt millors que en el cas dels vídeos en directe.

Quan per part del client es pot tolerar el retard en la publicació, VoD és la millor opció. Aquest sistema és més fàcil d'implementar si hi ha prou temps i és millor si es vol proporcionar una millor qualitat de vídeo als usuaris.

És per aquest motiu que les empreses han adoptat el sistema de VoD en els últims anys per sobre del vídeo en directe i el creixement és exponencial.

# 3. Arquitectura d'una plataforma de distribució de VoD sobre OTT

## 3.1. Introducció

Durant aquest treball es parlarà de molts conceptes tècnics, per això és important fer una introducció prèvia d'aquests abans de començar i així veure com es relacionen entre si i amb el projecte.

En aquesta introducció es veurà i s'entendrà el que son els còdecs i els contenidors, per a que serveixen i quines diferències hi ha entre aquests. També, quina tècnica, protocol i estàndard és el més útil per assolir l'objectiu d'aquest treball i la raó de la seva elecció.

Com ja s'ha comentat, el serveis OTT es distribueixen sobre Internet. Internet presenta la seva pròpia dinàmica en quant a les aplicacions i protocols que més es fan servir en aquesta xarxa pública. Des del punt de vista de protocol d'aplicació, l'HTTP, i les seves variants segures (HTTPS) són el protocols d'aplicació predominants a Internet. Molts dels serveis que es donen a Internet fan servir l'HTTP, perquè l'usuari ja està habituat a fer servir el client d'HTTP, el navegador web. En particular, les plataformes de VoD sobre OTT també han evolucionat en aquest sentit. La Internet, com també s'ha comentat abans, presenta com a característica principal, que l'ample de banda, entre dos extrems que es comuniquen, es variable, i això es un inconvenient en serveis de distribució de vídeo, ja que aquest tipus d'informació es molt sensible a aquest fenomen. En altres paraules, si pensem que una seqüència de vídeo es pot interpretar com un seguit d'imatges presentades a una certa taxa (fps), i aquestes s'envien a través d'una xarxa amb ample de banda variant, no podrem assegurar que les imatges es rebin a instant que toca, i la reproducció del vídeo serà irregular.

Per pal·liar aquest problema a Internet, les estructures de VoD sobre OTT utilitzen la tècnica ABR per preparar i enviar els vídeos que distribueixen. A mode de síntesi, l'ABR estableix que per enviar una seqüencia de vídeo a través de la xarxa, es codifiquin varies còpies de diferent qualitat de la seqüència original. Al variar la qualitat, també serà necessari un ample de banda diferent per transmetre cada còpia.

Entre el servidor i el client s'estableix un protocol per notificar l'estat de la transmissió, de forma continua, de la seqüència de vídeo. Si l'ample de banda baixa en un cert instant, se selecciona una còpia de menor qualitat per la transmissió a partir d'aquest moment. Si l'ample de banda puja, se selecciona una còpia de major qualitat. Hi ha varis protocols o sistemes que implementen aquest esquema de funcionament. Aquests protocols estableixen com ha de ser el contenidor (muxer) de les sequencies de vídeo, com son els missatges entre client i servidor per dur a terme les notificacions sobre la xarxa, etc. El més rellevants són el HLS d'Apple i el MPEG-DASH. Nosaltres ens hem centrat en l'HLS

Un esquema bàsic d'aquesta relació entre servidor i client, és el que es pot veure en la següent figura.

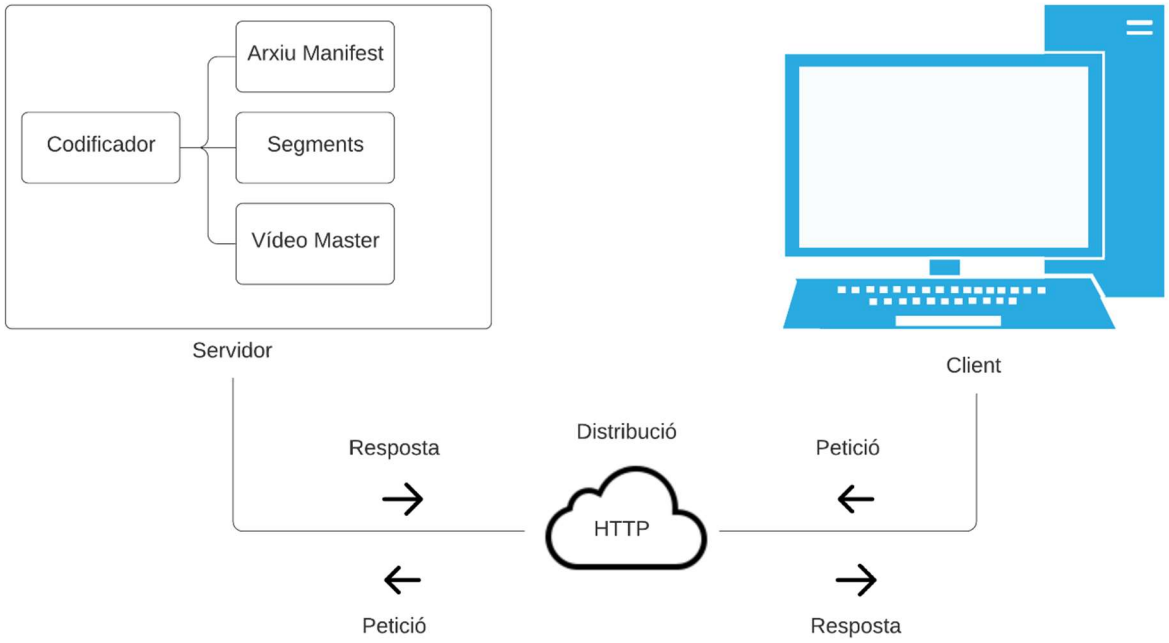



*Figura 3.1: Model Client – Servidor en una cadena de VoD.*

A continuació es presenta en més detall els conceptes i tecnologies que s'esmenten.

## 3.2. ABR

Aquest treball es centra principalment en una tècnica utilitzada per lliurar el vídeo de forma més eficient al client. Aquesta s'anomena ABR (Adaptative BitRate), la qual serveix per ajustar dinàmicament el nivell de compressió i la qualitat de vídeo adaptant-se a l'ample de banda disponible. D'aquesta manera, s'estalvia imprevistos com el *buffering* o *frezzing*, els quals provoquen que el vídeo es quedi congelat i que surti un símbol de càrrega durant uns segons. Això pot ocórrer quan es té un bitrate constant i l'ample de banda disponible disminueix.

Aquesta tècnica normalment s'utilitza amb protocols d'assemblatge com HLS o MPEG-DASH, on diversos fluxos es defineixen per perfils com la baixa, mitjana i alta qualitat. És a dir, es creen 3 còpies del mateix vídeo amb diferents qualitats i cadascuna d'aquestes còpies es divideixen en fragments de vídeo, entre 1 i 15 segons, de manera que els dispositius de visualització poden escollir dinàmicament el fragment de vídeo que millor s'adapti a l'amplada de banda disponible en un moment determinat ([7]). Per tant, una vegada hi ha disponible l'arxiu amb tota la informació dels segments que hi ha disponibles, el controlador utilitzarà la tècnica d'ABR per decidir quin d'aquests segments de vídeo agafar i amb quina qualitat.

Per fer-ho s'utilitza un algoritme que detecta quan ha de canviar a una qualitat inferior en el moment en que el temps de descàrrega d'un segment és superior al temps que dura aquell segment, és a dir que el temps de descàrrega és superior a la velocitat de reproducció i per tant de cara al client, el vídeo s'atura en certs moments per carregar.

Per posar un exemple pràctic, al principi del vídeo, el reproductor pot tenir molta amplada de banda, de manera que comença a transmetre's a la resolució més alta disponible (1080p). La reproducció continua sense problemes durant els primers 5 minuts, però al final d'aquests 5 minuts, la connexió a Internet comença a patir i ara hi ha menys amplada de banda disponible, de manera que el vídeo s'ha de degradar a una qualitat inferior (360p) durant el temps que faci falta. A mesura que torni a haver-hi disponible més amplada de banda, l'usuari tornarà a veure el vídeo a resolucions més altes.

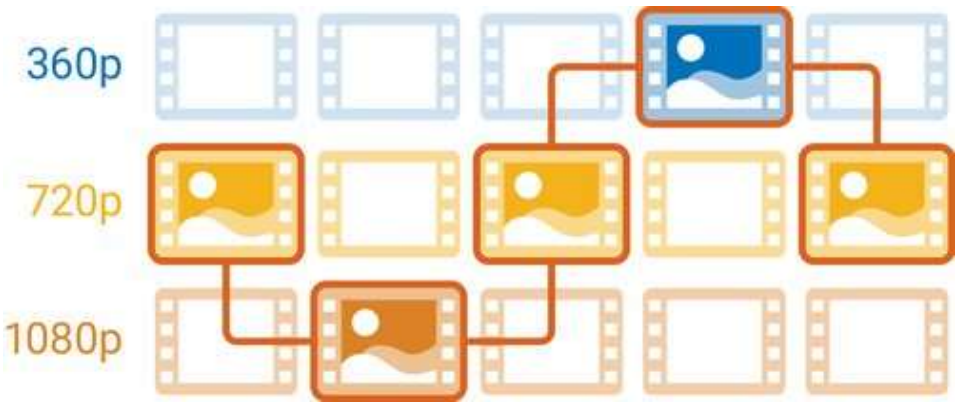



*Figura 3.2: Exemple de cadena de vídeo, aplicant la tècnica ABR. Variació de la resolució. Imatge original de Medium* [8].

Tot aquest canvi de resolució depèn totalment del reproductor, és per això que és tan important fer servir el reproductor adequat per marcar la diferència.

En certes ocasions, la reproducció es pot veure borrosa i després de uns segons es torna a veure correctament. Això pot ocórrer quan la connexió a internet no és la més òptima i just abans que el vídeo es tornés borrós, el reproductor ha determinat que no hi havia prou amplada de banda disponible per mantenir la reproducció en temps real amb tanta qualitat, de manera que té dues opcions:

1. Fer esperar (Buffer), és a dir, aturar el vídeo i mostrar un girador de càrrega que farà esperar mentre es descarreguen més segments de la qualitat més alta.
2. Degrada a una resolució inferior perquè es pugui seguir veient el vídeo.

Un bon reproductor escollirà la opció número 2, doncs és millor donar una resolució més baixa en lloc de fer esperar a l'usuari, tot i que l'objectiu del reproductor sempre serà oferir la màxima interpretació que pugui donar, sense fer esperar a l'usuari final.

Per veure com quedaria el sistema al complet, a la següent figura es mostra en primer lloc un servidor amb un vídeo separat per segments de 3 qualitats diferents, després s'envia a la xarxa, la qual té un valor d'ample de banda variable en el temps per a cada client. Per tant, en últim lloc, el reproductor del client haurà d'agafar el segment del vídeo que correspongui a la qualitat permesa per l'ample de banda.

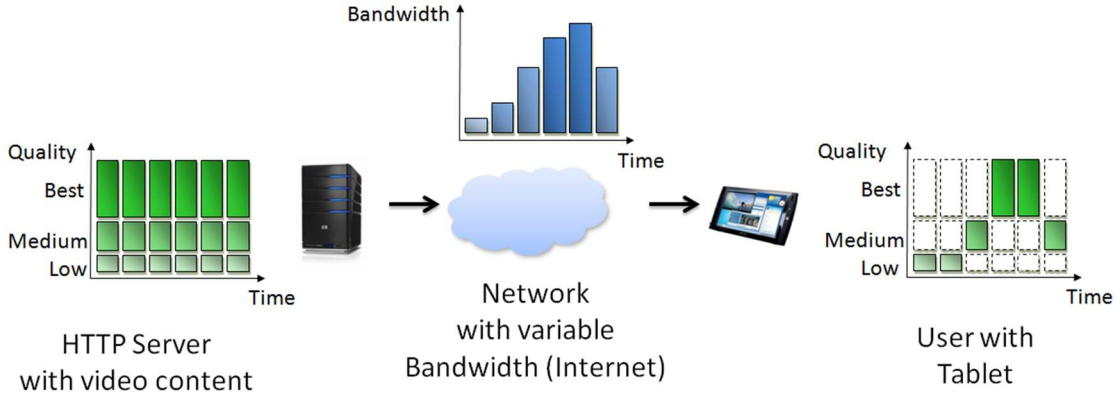



*Figura 3.3: Imatge d'un sistema client-servidor amb tècnica ABR* [9].

## 3.3. HLS

Ara que ja s'ha escollit quina tècnica es farà servir durant la realització del treball, és hora de triar el protocol de streaming, el qual serà l'encarregat de crear els arxius de vídeo que posteriorment s'utilitzin per a la reproducció. És a dir, agafa la informació d'un fitxer i la codifica de manera que pugui ser transmesa per la xarxa i que finalment el reproductor la pugui descodificar.

El terme HLS prové de "HTTP Live Streaming", i es tracta d'un protocol de streaming de vídeo creat per Apple que s'encarrega de convertir els arxius de vídeo en arxius HTTP òptims per ser descarregats en un format més petit per a més endavant ser reproduïts per part del client. Tot dispositiu connectat a internet és compatible amb HTTP, per tant, HLS té una implementació mes fàcil que altres protocols, com seria el cas de MPEG-DASH, que no té suport en els dispositius iOS ([10]). Durant la retransmissió, el vídeo pot augmentar o disminuir de qualitat en funció de les condicions de la xarxa sense arribar a interrompre's, tal i com s'ha vist en el punt anterior, gràcies a l'ABR [11].

Algunes de les característiques principals d'aquest protocol són les següents:

### 3.3.1. M3U8

En primer lloc, es troben les llistes de reproducció m3u8, creades duran la codificació amb el protocol HLS. Aquestes llistes contenen a dins múltiples arxius M3U, també coneguts com MP3 URL (Uniform Resource Locator). Aquest tipus d'arxiu conté tota la informació rellevant sobre el contingut multimèdia al què es vincula, inclòs el nom d'usuari, la seva ubicació, el títol o informació sobre la xarxa.

Inicialment, els arxius M3U només s'usaven per compilacions de MP3. No obstant això, el seu abast es va expandir amb el temps per incorporar arxius de vídeo. Avui dia, els arxius M3U i les llistes de reproducció són la forma més popular de connectar-se als canals d'IPTV (Internet Protocol Television).

Cada arxiu M3U està compost per dos components: el nom i l'enllaç. El primer conté el nom de canal que es veu a la llista de reproducció. La segona part conté l'enllaç a on es pot trobar el canal. La secció d'enllaços està composta per la direcció IP, nom d'usuari i contrasenya. La identificació de canal també s'ha d'incloure [12].

El text de l'arxiu pot utilitzar caràcters Unicode però en aquest cas s'ha de codificar en UTF-8 i utilitzar l'extensió d'arxiu "M3U8" o "m3u8" [13].

En aquest projecte, aquesta llista és molt important, ja que els fitxers d'índex per a la transmissió en directe HTTP es desen com a llistes de reproducció M3U8. Els clients accedeixen a l'URL del fitxer d'índex, que després sol·licita els fitxers indexats de forma seqüencial. A l'apartat 3.3.4 i en el 3.3.5 s'explicarà millor com funciona aquest procés i quines son les comandes necessàries per fer-ho.

### 3.3.2. Variants

En segon lloc, un altre de les propietats principals de l'HLS, són les rendicions de vídeo (vídeo rendition) també conegudes com a variants.

Per a una plataforma de VoD, aquest terme és vital, doncs es tracta de diferents versions del fitxer original dins d'un conjunt que s'utilitza per a la transmissió d'ABR. Les variants d'un conjunt solen diferir per resolució i velocitat de bits, però també poden diferir per còdec, fotogrames per segon i idioma. En aquest cas, les rendicions només variaran en resolució i bitrate.

L'objectiu de les variants és tenir el mateix fitxer repetit varies vegades amb les diferencies abans esmentades. D'aquesta manera, gràcies a la tècnica d'ABR, durant la reproducció del vídeo, el propi reproductor és el que s'encarrega d'escollir la variant que millor s'adapta a l'ample de banda en aquell precís instant.

Tal i com es veia a la Figura 3.3, les variants és el que correspon a la primera part de la imatge, on hi ha una gràfica separada en 3 files que són continues en el temps. Aquestes 3 files són les que corresponen als 3 segments creats durant la codificació, gràcies al protocol HLS.

Després, cada variant, es divideix en molts fitxers de curta duració anomenats segments i que s'explicarà a continuació.

### 3.3.3. Segment

Com s'ha explicat en el punt anterior, un vídeo és divideix en variants i cada variant en molts segments o chunks, cadascun contenint només uns segons de vídeo en un extrem, o emmagatzemat en un únic fitxer sense trossos a l'altre.

Un exemple pràctic dels segments es reflexa en com s'estructura normalment una presentació. No es sol parla contínuament d'un únic tema durant tot el temps. És molt més probable que es presenti una sèrie de temes i conceptes interconnectats, i que cada secció es centri en una idea clau o un objectiu d'aprenentatge, que comprèn un concepte més ampli. És a dir, que es separi la presentació en varis trossos.

Doncs tots aquests "trossos" del lliurament presencial proporcionen valuosos punts d'interrupció que es poden utilitzar per planificar i estructurar una presentació.

El mateix us se li dona als segments d'un vídeo. El vídeo es separa en molts fitxers petits que es distribueixen en el temps i d'aquesta manera no entregar un únic fitxer gran amb tota la informació. És doncs, el reproductor. Qui s'ha d'estructurar i repartir bé cada tros, segons les condicions d'ample de banda de l'usuari [14].

### 3.3.4. Manifest

Ara que ja es té preparades totes les variants juntament amb els seus segments, s'ha de tenir un fitxer el qual es pugui entregar al client per tal que conegui com està dividit el vídeo i quan ha de fer el canvi d'una variant a una altre.

Aquest fitxer és conegut com a llista de reproducció principal o manifest i és el primer document que es lliure quan s'inicia una transmissió de vídeo HLS. Té l'extensió M3U8 i proporciona al reproductor de vídeo informació sobre les diferents taxes de bits disponibles per a la transmissió, és a dir, descriu les variants de vídeo disponibles. També pot contenir informació sobre subtítols i fitxers d'àudio (si l'àudio es lliura per separat del vídeo) [15].

Un exemple visual d'arxiu manifest amb 3 variants i cada variant amb múltiples segments és el següent:

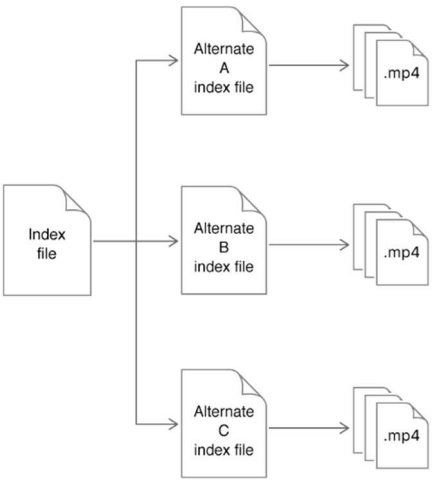



*Figura 3.4: Exemple d'arxiu manifest (Index file) amb 3 variants i múltiples segments. Imatge original de Apple* [16].

En aquesta figura es mostra el fitxer manifest, també anomenat índex, que conté la informació de 3 variants de vídeo, cadascuna amb una resolució i taxa de bits diferent. I, cadascuna d'aquestes variants, conté múltiples segments en format mp4 que són els que es reprodueixen.

Però la informació que hi ha dins l'arxiu principal manifest ha d'estar d'una manera determinada i en un ordre determinat, i una vegada que s'ha llegit per primera vegada l'arxiu, ja no es torna a llegir més, ja que el client assumeix que el conjunt de variants no canvia.

Aquesta informació interna s'anomena etiquetes de llista i a continuació s'explica amb més detall en que consisteixen.

### 3.3.5. Playlist Tags

La informació que hi ha dins el manifest és conegut com a playlist tags, i son un conjunt d'etiquetes amb el seu respectiu valor, les quals algunes són obligatòries i d'altres són opcionals o recomanables segons el còdec, contenidor o arxiu que s'utilitzi. Igual que en els arxius de les variants que també hi ha etiquetes informant en quin punt del vídeo es troba el reproductor. El flux finalitza tan bon punt el client veu l'etiqueta EXT-X-ENDLIST en una de les llistes de reproducció de variants individuals.

Les etiquetes que hi ha en el fitxer principal son les següents ([16]):

EXTM3U: indica que la llista de reproducció és un fitxer M3U ampliat. Aquest tipus de fitxer es distingeix d'un fitxer M3U bàsic canviant l'etiqueta de la primera línia a EXTM3U. Totes les llistes de reproducció HLS han de començar amb aquesta etiqueta.

EXT-X-STREAM-INF: indica que el següent URL del fitxer de llista de reproducció identifica un altre fitxer de llista de reproducció. L'etiqueta EXT-X-STREAM-INF té els paràmetres següents:

- AMPLADA DE BANDA MITJANA: (Opcional, però recomanable) Un enter que representa la taxa de bits mitjana per al flux de variants.
- AMPLADA DE BANDA: (Obligatori) Un enter que és el límit superior de la taxa de bits global de cada fitxer multimèdia, en bits per segon. El valor límit superior es calcula per incloure qualsevol sobrecàrrega del contenidor que aparegui o aparegui a la llista de reproducció.
- FRAME-RATE: (Opcional, però recomanable) Valor en coma flotant que descriu la velocitat de fotogrames màxima en un flux de variants.
- HDCP-LEVEL: (Opcional) Indica el tipus de xifratge utilitzat. Els valors vàlids són TYPE-0 i NONE. S'utilitza TYPE-0 per evitar que la reproducció es reprodueixi tret que la sortida estigui protegida per HDCP (High-bandwidth Digital Content Protection).
- RESOLUCIÓ: (Opcional, però recomanable) Mida de visualització opcional, en píxels, per mostrar tot el vídeo de la llista de reproducció. Aquest paràmetre s'hauria d'incloure per a qualsevol transmissió que inclogui vídeo.
- CODECS: (Opcional, però recomanable) Una cadena entre cometes que conté una llista de formats separats per comes, on cada format especifica un tipus de mostra de suport que es troba en un segment de suport del fitxer de llista de reproducció. Els identificadors de format vàlids són els que es troben a l'espai de noms de format de fitxer ISO definit per RFC 6381 [RFC6381].

Aquí hi ha un exemple d'un fitxer de configuració.

```
#EXTM3U
#EXT-X-STREAM-INF: BANDWIDTH=150000, RESOLUTION=416x234,
CODECS="avc1.42e00a,mp4a.40.2"
http://example.com/low/index.m3u8
#EXT-X-STREAM-INF: BANDWIDTH=240000, RESOLUTION=416x234,
CODECS="avc1.42e00a,mp4a.40.2"
http://example.com/lo_mid/index.m3u8
#EXT-X-STREAM-INF: BANDWIDTH=440000, RESOLUTION=416x234,
CODECS="avc1.42e00a,mp4a.40.2"
http://example.com/hi_mid/index.m3u8
#EXT-X-STREAM-INF: BANDWIDTH=640000, RESOLUTION=640x360,
CODECS="avc1.42e00a,mp4a.40.2"
http://example.com/high/index.m3u8
#EXT-X-STREAM-INF: BANDWIDTH=64000, CODECS="mp4a.40.5"
http://example.com/audio/index.m3u8
```

*Llista 3.1: Exemple de fitxer de configuració amb les playslist tags d'un arxiu Manifest. Codi original d'Apple* [16].

En resum, aquests són els passos que segueix el reproductor a l'hora de reproduir un vídeo respecte a l'arxiu manifest:

- Carrega el manifest principal que conté informació sobre cada interpretació.
- Esbrina quines representacions hi ha disponibles i tria la millor segons l'amplada de banda disponible.
- Carrega el manifest de renderització o variant per esbrinar on es troben els segments.
- Carrega els segments i inicia la reproducció.

Una vegada es troba en funcionament, entra en joc la transmissió de velocitat de bits adaptativa, tal i com s'ha explicat prèviament, que depèn del reproductor i és la que s'encarrega de seleccionar cada variant i per tant de com es veurà el vídeo.

## 3.4. Codificació

Un arxiu de vídeo té molta informació guardada, no només hi ha les imatges que veiem per pantalla. En aquest projecte parlarem sobre totes les propietats que té un vídeo i com optimitzar-los per a vídeo sota demanda.

En aquest apartat es veuran propietats com el còdec de vídeo, els GOP (Group of Images), els segments d'un vídeo, l'arxiu manifest, la velocitat de bits i les resolucions juntament amb les graelles de comparació.

### 3.4.1. H.264

El primer paràmetre a l'hora de codificar és escollir com es procedirà a fer la compressió de les dades i com s'enviaran des del servidor al client. Ja que en la forma original del arxius multimèdia, les seves dades són massa grans per processar-les i ser transmeses pels sistemes de comunicació dels que es disposen avui dia.

Per tant, aquesta feina de codificació multimèdia, recau sobre el còdec (codificador/descodificador). Això no és més que un esquema que regula una sèrie de transformacions en un senyal o informació. Els còdecs tant poden transformar un senyal a una forma codificada (usada per la transmissió o encriptació) com fer-ho al revés, donar el senyal adequat per a la seva visualització o edició (no necessàriament la forma original) a partir de la forma codificada [17].

Aquests es poden dividir en còdec amb pèrdues i còdec sense pèrdues, depenent si la informació que es recupera coincideix exactament amb l'original.

Evidentment com més fidel a la realitat sigui el còdec, millor resultat tindrà, més espai ocuparà i per tant una major taxa de bits serà necessària. És a dir que la mida de la reproducció serà major però la qualitat també. I el mateix a la inversa, la mida de la reproducció pot ser menor però la qualitat també disminuirà [18].

Per optimitzar els fluxos de vídeo, els còdecs emmagatzemen el moviment de la imatge entre fotogrames en comptes del fotograma complet. És a dir, mira quina diferencia hi ha entre dos fotogrames, i aquella és la informació que guarda. D'aquesta manera estalvia molt espai d'emmagatzematge i pot anar més ràpid durant l'anàlisi. Aquest mètode s'anomena compensació de moviment, i existeixen diferents tipus de fotogrames que ho permeten fer:

- El fotograma intra (fotograma I) que és una representació completa
- El fotograma predictiu (fotograma P) que conté la diferència compensada pel moviment entre els fotogrames I anteriors.
- El fotograma Bi-predictiu (fotograma B) que emmagatzema la compensació del moviment entre el passat i el futur I i P.

Com que la intenció d'aquest treball no és fer un anàlisis molt extens sobre els còdecs, es parlarà només sobre el còdec de vídeo que s'ha utilitzat durant aquest treball, l'H.264.

L'H.264 és un còdec digital d'alta compressió estàndard i amb pèrdues, també conegut com AVC "Advanced Video Coding" o MPEG-4 Part 10. Aquest permet augmentar fins a quatre vegades la mida dels fotogrames a la vegada que redueix fins a un terç l'ample

de banda necessari per reproduir-lo ([19]). És a dir, que millora molt la qualitat del vídeo a la vegada que s'estalvia ample de banda, respecte a les versions anteriors.

L'alternativa principal del còdec que s'ha utilitzat en aquest treball, l'H.264, seria l'H.265 que ofereix una qualitat de vídeo fins a 8K i per tant millor experiència audiovisual. Però, l'elecció principal del còdec H.264 és la compatibilitat amb la majoria de serveis i dispositius. Doncs, malgrat que la versió més nova ja porta anys al mercat, no s'ha popularitzat tant i per tant encara hi ha navegadors i serveis que no el suporten [20]. Això no treu que en els propers anys, H.265 superi a H.264 i la majoria de navegadors ja acceptin aquest còdec en les seves últimes versions.

En quan a H.264, dues de les característiques més notables respecte als estàndards anteriors, que més afecten a la qualitat del vídeo i permeten obtenir uns millors resultats son les següents:

### *Imatges*

Aparició de dos tipus nous de fotogrames. Segueixen existint els fotogrames de les versions anteriors: Fotogrames I, P, B. Però ara s'afegeixen dos tipus nous: Fotograma SI (*Switching I*) i fotograma SP (*Switching P*) que codifiquen la transició entre dos fluxos de vídeo. Utilitza predicció temporal o espacial per passar d'un vídeo a l'altre però permet la reconstrucció de valors específics exactes de la mostra tot i utilitzar imatges de referència diferents o un nombre diferent d'imatges de referència en el procés de predicció.

### *Compensació de moviment*

Hi ha una gran varietat de formes i particions de blocs. Aquests serveixen per precisar més el resultat final. És a dir, els macro-blocs es divideixen en sub-blocs per a l'estimació de moviment i poden tenir set mides diferents = 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 i 4x4. Pot haver-hi una precisió de fins a un quart de píxel. Aquesta divisió de blocs, com més petita millor ja que hi ha més precisió en els píxels, tot i que la codificació pot prendre més temps.

Acabades aquests dues característiques i l'explicació de l'H.264, es pot comprendre millor la utilitat i importància de tenir un bon còdec en qualsevol procés multimèdia. Però, seguint les propietats de codificació d'un sistema de VoD, encara n'hi ha moltes que afecten al correcte funcionament i a la qualitat del resultat final, començant pels grups d'imatges.

### 3.4.2. GOP

Aquests grups d'imatges, també coneguts com a GOP, fan referència a un grup de fotogrames ordenats linealment en el temps. Aquests grups comencen amb un fotograma I (fotograma intra), què és el més important, i després conté una sèrie de fotogrames P (fotograma predictiu) i B (fotograma Bi-predictiu). Aquest esquema es va repetint successivament fins al final de l'arxiu multimèdia. Els GOPs són necessaris ja que el segment al qual es vol començar a reproduir el vídeo sempre serà l'inici d'un GOP, amb el seu fotograma I corresponent, per tant si en un vídeo es volgués avançar a un punt aleatori, el reproductor buscaria quin és el fotograma I més a prop que hi ha i començaria a descodificar des d'aquell punt.

Hi ha dos tipus de grups d'imatges, els oberts i els tancats. Els GOPs oberts són grups d'imatges on els marcs poden fer referència a marcs d'altres GOPs per a blocs redundants. Es pot veure en la Figura 3.5, on els dos darrers fotogrames B fan referència al fotograma I del següent GOP per a redundàncies.

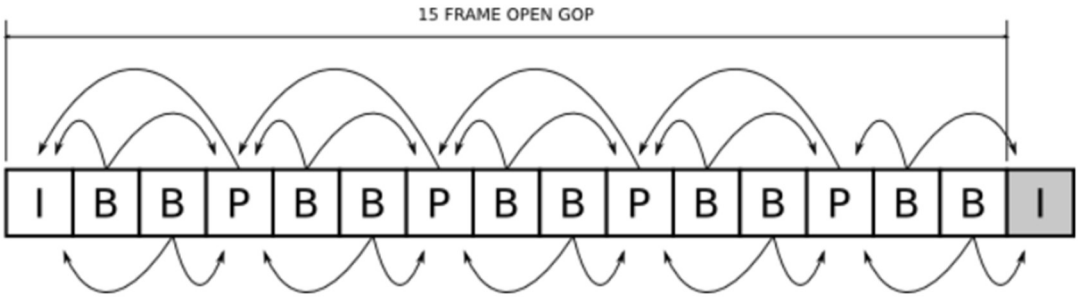



*Figura 3.5: Configuració d'una cadena de vídeo amb els seus marcs i un GOP obert. Imatge original de Tiliam* [21].

Els GOP tancats, són grups d'imatges on els marcs només poden fer referència a marcs del mateix GOP per a blocs redundants.

A la següent figura es pot veure que el segon GOP comença amb un marc IDR (Insatantaneous Decoder Refresh), això significa que s'esborra el contingut del buffer d'imatges de referència. En rebre una imatge codificada per IDR, el descodificador marca totes les imatges de la memòria intermèdia de referència com a "no utilitzades per a referència". Totes les parts transmeses posteriors s'han de descodificar sense fer referència a cap fotograma descodificat abans de la imatge IDR. Per tant, la primera imatge d'una seqüència de vídeo codificat sempre és IDR, de manera que els marcs d'aquest GOP no poden referir-se als marcs anteriors de la imatge I. Si el següent fotograma I també és un marc IDR, aquest seria un GOP tancat [22].

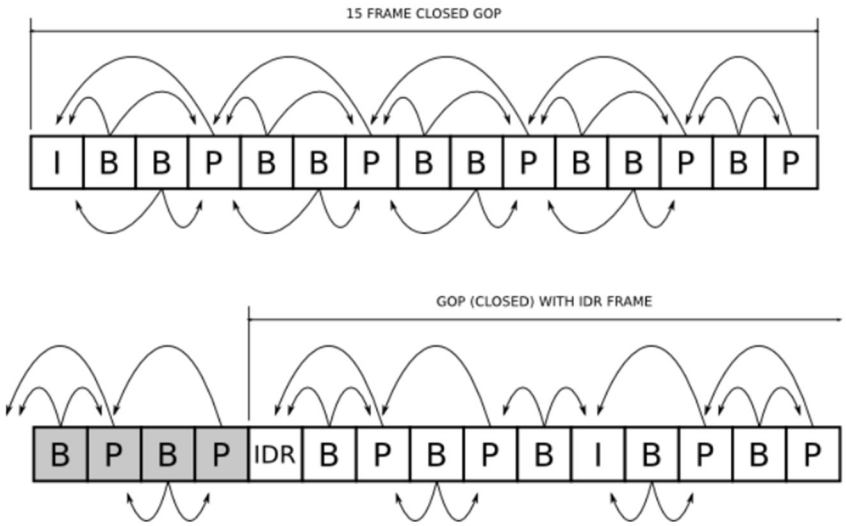



*Figura 3.6: Configuració d'una cadena de vídeo amb els seus marcs i un GOP tancat. Exemple d'imatge IDR. Imatge original de Tiliam* [21].

La codificació de fitxers amb GOPs oberts suposa que produeix una millor qualitat que els GOPs tancats perquè els marcs B i P poden fer referència als marcs situats fora del mateix GOP. No obstant això, els fitxers produïts per a la transmissió amb la tècnica ABR han d'utilitzar GOPs tancats, ja que en aquests casos el reproductor canvia els fluxos en funció de les condicions d'amplada de banda i d'altres factors. Amb un GOP obert, quan el reproductor passa d'un flux de 360p a un flux de 1080p, els fotogrames anteriors al fotograma I al començament del segment són diferents dels fotogrames referits durant la codificació, cosa que pot causar problemes, ja que està utilitzant de referència un marc que pot ser no és de la resolució que es desitja i té un bitrate diferent.

Inclús, en les especificacions d'HLS diu: "Els marcs clau (IDR) haurien d'estar presents cada dos segons". Per tant, tots els fotogrames clau han de ser marcs IDR, cosa que significa que tots els GOPs han de ser tancats [23].

Per defecte, en la configuració del còdec H.264, els grups d'imatges son tancats, en canvi, en el còdec H.265 són oberts. No obstant, això té un efecte molt baix en la qualitat final del vídeo i es pot canviar a l'hora de codificar amb els paràmetres de configuració.

### 3.4.3. Taxa de codificació

Ara que ja es coneix com funcionen els GOPs i es coneix la diferència entre un GOP obert i un GOP tancat, es procedeix a explicar el que es la taxa de codificació i per que té un impacte tan gran durant la codificació i sobretot en el resultat final.

La taxa de codificació o velocitat de bits (bitrate), correspon a la quantitat de dades codificades per unitat de temps, per a la transmissió, normalment es fa referència en megabits per segon (Mbps) per al vídeo i en kilobits per segon (kbps) per a l'àudio. Des d'una perspectiva de transmissió, una taxa de bits de vídeo més alta significa un vídeo de més qualitat que a l'hora requereix més amplada de banda [24].

Hi ha diferents tipus de bitrate, però els més coneguts son el constant i el variable. I depenent de la utilitat que es vulguis donar serà més útil un que l'altre. Les característiques i diferencies principals d'aquests dos tipus són les següents:

- CBR (Constant Bit Rate): Té com a objectiu un nivell d'amplada de banda constant o invariable amb una qualitat de vídeo que pot variar. Mitjançant aquesta codificació, la taxa de bits i la grandària dels fitxers es poden conèixer abans de codificar, si es disposa de les característiques principals d'aquest. De totes maneres, aquesta no és una solució òptima ni per a l'emmagatzematge ni per a aplicacions de streaming, ja que les parts de vídeo que son més complexes es codifiquen amb el mateix nombre de bits que les parts més senzilles d'aquest, això provoca que hi hagi diferències de qualitat, sobretot en aquelles parts que es requereix major velocitat. La solució normalment es troba en escollir una taxa de bits constant elevada per a assegurar-se que les seccions de vídeo més complexes de codificar tindran la qualitat necessària per a una bona recepció i posterior reproducció o tractament. No obstant, aquest mètode augmenta proporcionalment la mida dels fitxers finals, i s'infrautilitzen gran quantitat de bits per a seccions que podrien ser codificades amb valors molt inferiors [23].

- VBR (Variable Bit Rate): Tal i com el seu nom indica, permet variar la velocitat de bits, però manté un nivell de qualitat de vídeo constant. Contràriament a la tècnica de codificació CBR, que fa tot el possible per mantenir una taxa de bits fixa en tot el contingut codificat, aquesta varia la taxa depenent de les característiques de l'arxiu a codificar, assignant de manera flexible més bits a les parts amb informació més complexa per obtenir la millor qualitat possible, i menys bits a les seccions amb informació més senzilla de codificar.

Les desavantatges més importants de VBR són que requereix més temps per codificar que CBR, a mesura que el procés es fa més complex, i que algun hardware pot ser incompatible amb fitxers VBR, tot i que cada cop menys es donen aquest segon tipus de problemes.

A més d'aquests dos tipus de bitrate, hi ha un tercer què és el que s'ha utilitzat en aquest treball. Es tracta de MBR (Maximum Bit Rate), que significa velocitat de bits màxima i que permet variar la velocitat de bits, però només fins a un valor màxim. Això també és conegut com a VBR capped (VBR amb límit) la qual combina les millors parts de codificació VBR i CBR.

Segons IPVM ([23]), en comparació amb una configuració típica de CBR, MBR sovint redueix el consum d'amplada de banda en un 30-70%. Per aconseguir-ho, permet reduir

l'amplada de banda que s'utilitza quan l'escena és senzilla (mentre que el CBR sempre es manté bloquejat a la velocitat de bits fixa).

En comparació amb una configuració típica de VBR, MBR pot reduir el consum d'amplada de banda en un 20-50%. Ho aconsegueix deixant que el consum d'amplada de banda de VBR exploti (normalment en escenes fosques) imposant un nivell d'amplada de banda màxim. És probable que no es produeixi cap pèrdua de qualitat pràctica perquè l'escena fosca redueix els detalls de la imatge capturada de totes maneres.

Per exemple, si una càmera consumeix 2 Mb / s durant el dia, però 10 Mb / s a la nit per agafar millor els detalls, es pot limitar-la fàcilment a 5 Mb / s (o fins i tot menys) sense cap pèrdua de qualitat. Això permet reduir els costos globals tant d'emmagatzematge com d'ample de banda en un 50% [25].

Per això, en aquest treball s'utilitzarà una configuració de MBR, d'ara en endavant coneguda com a VBR Capped. Ja que és la més recomanada per a sistemes de VoD i també quan es vol treballar amb la tècnica ABR.

Per últim, ara es parlarà d'una propietat que va molt relacionada amb la taxa de codificació, ja que la seva configuració haurà d'anar en paral·lel amb aquesta. Aquesta propietat configura la quantitat de píxels que hi haurà en el vídeo i quina taxa de bits tindrà cada configuració.

### 3.4.4. Esgraonat

Per acabar l'apartat de propietats de codificació, es parlarà sobre la l'esgraonat. Aquest és un dels punts més notables de cara al públic, doncs determina amb quina qualitat es reproduirà el fitxer. En aquest punt entren dos variables en joc. La primera és la ja coneguda taxa de bits, i la segona és la resolució de pantalla. La resolució és al que anomenem la suma de píxels que es mostren, i per tant quants més píxels hi ha, més informació es podrà mostrar per pantalla.

En aquets treball es codificarà el vídeo en 3 resolucions i bitrates diferents (3 esgraons), això és degut a que més endavant s'implementarà un mecanisme d'incorporació de vídeo capaç de detectar la velocitat de connexió a internet de l'espectador i triar el fitxer de vídeo adequat en funció d'aquesta velocitat d'enllaç, juntament amb la mida de la pantalla i la reproducció, capacitats del navegador, proporcionant així a cada visor diferent la millor resolució i velocitat de bits que pot utilitzar.

Per escollir els valors d'esgraonat en el VoD, s'ha de tenir en compte varis factors, com seria l'ample de banda del servidor, l'ample de banda del client, en quin o quins dispositius es reprodueix el vídeo (Mòbil, Televisor, Ordinador...) i en cas de no saber-ho, com passa en la majoria de vegades, es pot utilitzar la tècnica ABR, per tal que s'adapti als diferents formats, quin contingut s'envia (vídeos molt carregats, amb molt detall, estàtics...) i per últim a quina plataforma o CDN s'envia el contingut.

El canvi de resolució es fa seguint l'esgraonat juntament amb la taxa de bits. Això és degut a que és necessita establir un llindar, el qual una vegada es sobrepassat, es faci el canvi de resolució. Hi ha molt tipus d'esgraonats, i depenent de la capacitat del servidor i de la demanda, els valors poden variar.

Per exemple, aquest passat estiu, degut a l'augment d'usuaris durant el confinament, Netflix va fer un nou esgraonat en el seu servidor i van determinar un bitrate més baix per a resolucions més altes, acció que va fer enfadar als seus usuaris, doncs la qualitat que veien prèviament en les sèries i pel·lícules havia disminuït notablement [26].

Per això és important que abans de la publicació d'un nou sistema, es revisi amb deteniment l'esgraonat que es farà servir i així oferir la màxima resolució i taxa de bits als clients.

El valor exacte per a la resolució i per al bitrate (esgraonat) que s'utilitzarà en aquest treball prové de la següent formula: Per a cada resolució es farà servir una taxa de bits que és el tall sensible més baix del 64% (80% del 80%) d'una velocitat d'enllaç d'Internet comuna ([27]). A continuació, en la Taula 3.1 es mostren els esgraonats amb les resolucions més comunes i el seu respectiu 64% d'ample de banda.

| Resolució (Píxels x Píxels) | Connexió (Mbps) | Bitrate (Mbps) | Vídeo (Kbps) |
|---|---|---|---|
| **424x240** | 1.0 | 0.64 | 576 |
| **640x360** | 1.5 | 0.96 | 896 |
| **768x432** | 1.8 | 1.15 | 1088 |
| **848x480** | 2.0 | 1.28 | 1216 |
| **1024x576** | 3.0 | 1.92 | 1856 |
| **1280x720** | 4.0 | 2.56 | 2496 |
| **1920x1080** | 8.0 | 5.12 | 4992 |

*Taula 3.1: Taula amb els valors corresponents de bitrate per a cada nivell de resolució. Taula original de Lighterra* [28].

Però també hi ha altres configuracions d'esgraonat populars que servirien per utilitzar en aquest treball, com per exemple l'oficial d'Apple que fa servir amb el protocol HLS, que tal i com descriu en la seva web, uns valors correctes per a la configuració de resolució i bitrate serien els següents [29].

| Resolució (Píxels x Píxels) | Vídeo (Kbps) |
|---|---|
| **424x240** | 145 |
| **640x360** | 365 |
| **768x432** | 730 |
| **848x480** | 1100 |
| **1024x576** | 2000 |
| **1280x720** | 3000 |
| **1920x1080** | 6000 |

*Taula 3.2: Taula amb els valors de bitrate més comuns per a cada resolució segons Apple* [30].

## 3.5. Contenidor (Muxer)

Un dels conceptes claus a la hora d'emmagatzemar un vídeo és el contenidor, també conegut en el món audiovisual com a Muxer, el qual no deixa de ser un tipus d'arxiu que té la funció de guardar informació d'àudio, d'imatges, de vídeo, subtítols o metadades, sincronitzats per un rellotge, en un determinat format. Aquest format pot ser estandarditzat i pot estar regit per un tipus concret de compressió, tot i així, en general, un contenidor multimèdia no implica la utilització d'un còdec determinat.

Hi ha contenidors específics per un determinat tipus d'informació, mentre que altres són compatibles a la combinació de diversos tipus. Caldrà, per tant, fer-ne una diferenciació a l'hora d'escollir-ne un.

Una vegada està tota la informació multimèdia guardada en un d'aquests contenidors, per poder reproduir-ho serà suficient utilitzar un reproductor multimèdia, com VLC o en el cas d'aquest treball Ffplay, que a partir de la demultiplexació dels fluxos de dades que contingui, permetrà la reproducció d'aquests i es podrà gaudir de tot el contingut a l'hora, ja sigui per aplicacions de cinema, videoclips, música, fotografies...

Dins l'àmbit multimèdia, un contenidor és una especificació sobre la manera com s'ordenen dins un arxiu diferents tipus de contingut multimèdia codificat. Aquests continguts solen ser, sobretot, vídeo, àudio i text.

Si es pensa en el cas d'una pel·lícula en format digital, aquesta es troba en un arxiu, el qual quan es obert s'executa un reproductor i es pot començar a mirar la pel·lícula. Per tant, aquesta pel·lícula, té el seu vídeo, àudio, subtítols, informació de l'autor, informació de la durada, etc. Per a comoditat de l'usuari, s'agrupen tots aquests elements dins un sol arxiu i l'ordre en què es desen en aquest arxiu és l'especificació del contenidor [31].

En aquest treball, s'utilitza el contenidor mp4, que és un dels contenidors més utilitzat i pertany al grup MPEG. S'utilitza comunament per a arxius d'àudio i vídeo i permet tenir tota la informació en una mida compacte i fàcil de transportar ja que la majoria de serveis són compatibles amb aquest contenidor [32].

## 3.6. Playback chain

Aquest treball comporta el desenvolupament d'una plataforma per a la distribució de VoD, i l'esquema més bàsic de la plataforma juntament amb els processos que hi ha darrera, es troba a la Figura 3.7.

Al principi es rep o es grava un vídeo, el qual s'ha de normalitzar i així tenir-ho amb un còdec i un contenidor adequat.

Una vegada està normalitzat es procedeix a fer la codificació del mateix. En aquest moment es creen les còpies del vídeo (variants) cadascuna codificada a diferents resolucions i bitrates, i a la vegada cadascuna amb els seus segments.

Després, s'ha de gestionar els vídeos codificats per a poder-ho enviar-ho a un CDN o publicar-ho a una pàgina web. En aquest projecte, és aquí on entra el mòdul de Kaltura que permet configurar el servidor Ngnix per a elements multimèdia i així ser enviat posteriorment a la web.

A més, amb l'ajuda de Ffmpeg i HLS, s'utilitzarà la tècnica ABR per poder gestionar la reproducció segons l'estat de la xarxa del client.

Més endavant es podrà veure més a fons com funciona cada punt de la figura i quin software o quines comandes s'han utilitzat en cada moment.

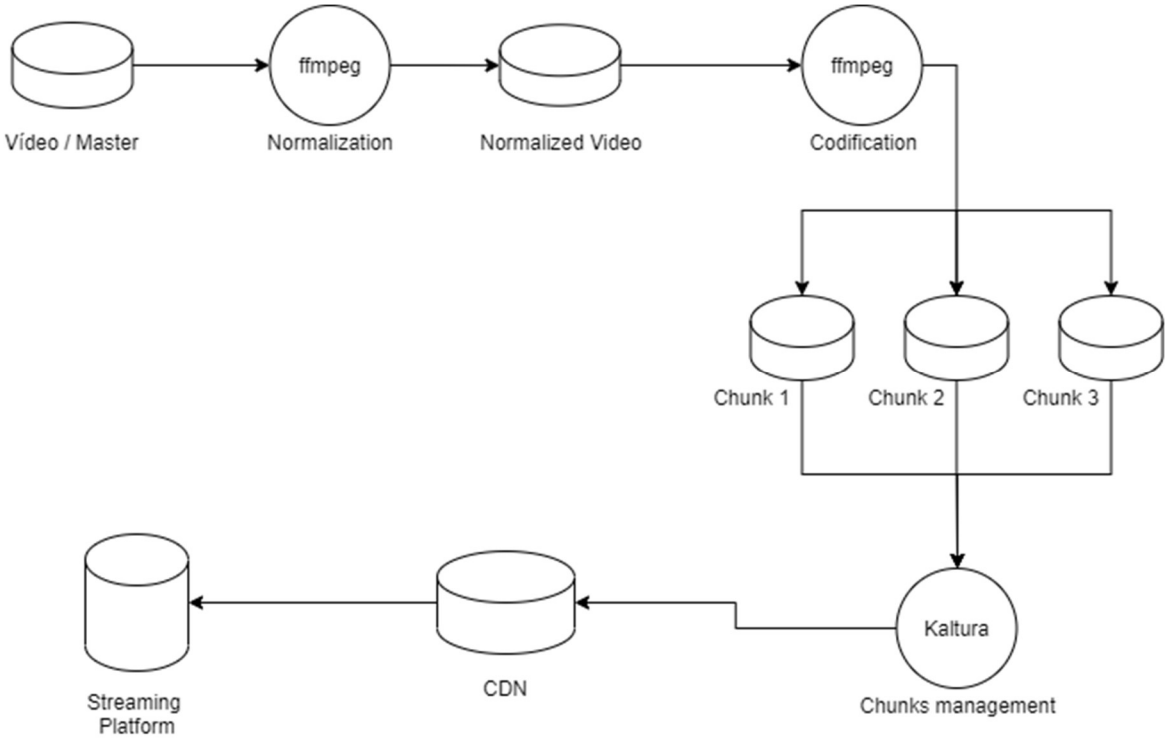



*Figura 3.7: Esquema bàsic d'una cadena de vídeo*

# 4. Desenvolupament pràctic

## 4.1. Requeriments

En aquest treball s'ha fet us de programari de codi obert per a fer totes les proves i codificacions. I s'ha treballat amb el sistema operatiu de Linux-Ubuntu, per poder utilitzar un sistema basat en Unix.

També, per fer les proves de codificació, s'ha utilitzat un vídeo open-source de la Blender Foundation: Big Buck Bunny [33].

A continuació s'explicarà resumidament quins són els programes que s'han fet servir per al desenvolupament d'aquest treball. Es seguirà l'esquema realitzat en el punt 3.1, per tant el primer pas serà la creació del servidor d'on sorgirà l'arxiu màster, l'arxiu manifest i les variables.

## 4.2. Programari seleccionat

Com ja s'ha mencionat, tot el programari escollit és de codi obert, i els dos programes principals són els següents.

### 4.2.1. Nginx

En primer lloc es troba Nginx, un programa lliure, que pot actuar com a servidor principal, servidor intermediari invers, equilibri de càrrega, servidor intermediari de correu i memòria cau HTTP [34].

En aquest treball s'ha fet servir aquest com a servidor principal dins el sistema de Linux. Per tant, Nginx s'encarrega de guardar tant pàgines web, com els arxius multimèdia principals, les variables amb els seus segments i tota la informació que sigui necessària per tal que el client pugui posteriorment reproduir contingut multimèdia des d'un reproductor basat en HLS.

Nginx disposa de diversos mòduls externs que amplien la seva funcionalitat. Molts d'aquests mòduls estan fets per terceres parts. En aquest projecte s'ha fet servir un mòdul específic per VoD desenvolupat per Kaltura.

Kaltura és el proveïdor líder de tecnologia de vídeo i creador de l'única plataforma de vídeo com a servei del món. És una plataforma de vídeo oberta altament fiable, escalable i flexible, que proporciona centenars de milers d'experiències de vídeo i fluxos de treball en indústries de més de 100 països de tot el món.

En aquest treball, s'ha utilitzat aquest servei com a mòdul addicional dins el servidor Nginx. D'aquesta manera, la tasca de crear les variables i els segment recau principalment en aquest mòdul especialitzat en vídeo sota demanda.

### 4.2.2. Ffmpeg

Per últim, com a software de codificació, hi ha Ffmpeg, un altre projecte de software lliure i de codi obert que consisteix en un gran conjunt de biblioteques i programes per gestionar arxius i fluxos de vídeo, àudio i altres fitxers multimèdia. S'utilitza àmpliament per a la transcodificació de formats, l'edició bàsica, l'escala de vídeo, els efectes de postproducció de vídeo i el compliment d'estàndards (SMPTE, ITU).

Ffmpeg forma part del flux de treball de centenars d'altres projectes de programari, i les seves biblioteques són una part fonamental dels reproductors multimèdia de programari, com ara VLC, i s'ha inclòs en el processament bàsic per a YouTube i iTunes. S'inclouen còdecs per a la codificació i / o descodificació de la majoria de formats de fitxers d'àudio i vídeo, cosa que el fa molt útil per a la transcodificació de fitxers multimèdia comuns i poc comuns en un únic format comú [35].

Les tres comandes que s'han utilitzat més son:

- `ffmpeg`: Comanda que converteix formats d'àudio o vídeo, que també pot capturar i codificar en temps real a partir de diverses fonts de maquinari i programari.
- `ffplay`: Reproductor multimèdia que utilitza les biblioteques FFmpeg i SDL (Simple DirectMedia Layer) que és un conjunt de llibreries desenvolupades amb el llenguatge C que proporcionen funcions bàsiques per realitzar operacions multimèdia d'entre altres.
- `ffprobe`: Comanda per mostrar informació dels arxius d'entrada, tal i com faria la comanda de mediainfo.

En resum, Nginx té la funció de servidor, Kaltura és un mòdul especialitzat en contingut multimèdia que funciona sota Nginx i s'encarrega de fer la segmentació i per últim Ffmpeg que actua com a codificador. A continuació és mostra el procés d'instal·lació i configuració del servidor.

## 4.3. Servidor de VoD

En aquest apartat es mostrarà el procés seguit per instal·lar i configurar el servidor Nginx. Tant el servidor com el mòdul de Kaltura s'ha clonat des de repositoris de codi obert que es troben a Github.

A continuació es mostra les comandes que s'han seguit, tenint en compte que alguna carpeta o nom d'arxiu pot variar segons el sistema operatiu o la versió que s'utilitzi. En aquest projecte, s'ha procedit a fer la instal·lació d'aquesta manera ja que s'ha treballat amb el sistema operatiu Linux Ubuntu, versió 16.04 de 64 bits.

El primer pas a fer és instal·lar el servidor i després el mòdul multimèdia. Les comandes necessàries per fer-ho són les següents.

### 4.3.1. Instal·lació Servidor Nginx:

El primer pas a fer abans d'instal·lar el servidor Nginx és instal·lar git, un software encarregat de clonar repositoris que es troben a la pàgina web de Github. Per fer-ho serà necessari tenir privilegis de super usuari (*root*) i actualitzar el sistema (*Update* i *Upgrade*). Una vegada ja estigui fet es pot procedir amb el servidor.

```
sudo -i
apt install git
git clone https://github.com/nginx/nginx.git
```

### 4.3.2. Instal·lació mòdul de segmentació Kaltura:

En quant al mòdul multimèdia, encarregat de la segmentació de vídeo, es seguirà el mateix procés que abans. S'instal·la des de Git, el mòdul de Kaltura, i les llibreries necessàries per a la seva compilació. Aquestes s'especificaran a l'hora de la instal·lació i depèn del sistema seran unes o altres.

```
git clone https://github.com/kaltura/nginx-vod-module.git
```

Una vegada s'ha clonat el mòdul, s'ha de seguir els passos indicats a la pàgina de GitHub, els quals són els següents [36].

Primer de tot, ubicar-se a la carpeta de Nginx dins el mòdul de Kaltura. I, des d'allà, instal·lar el mòdul de VoD.

--> cd <directori>

```
cd /opt/kaltura/nginx/
cp auto/configure .
./configure --add-module=../nginx-vod-module
make
make install
```

Una vegada instal·lat, s'ha de comprovar el correcte funcionament del servidor. Per fer-ho s'ha d'anar al directori d'arrencada del servidor i executar el programa. I, per comprovar que s'ha iniciat correctament, es procedeix a consultar els processos que hi ha en actiu a l'ordinador.

```
cd /usr/local/nginx/sbin
./nginx
netstat –tlnp
```

Finalment, al acabar la sessió, es procedeix a aturar el servidor abans d'apagar el sistema.

```
./nginx –s stop
```

### 4.3.3. Configuració del Servidor de VoD

Al instal·lar tant el servidor Nginx com el mòdul de Kaltura, es creen arxius de configuració amb algunes variables establertes per defecte necessàries per fer proves bàsiques, però el que s'ha fet en aquest treball, és canviar aquestes variables i afegir-ne de noves per deixar el servidor preparat, de manera que fos el més òptim per a una plataforma de VoD.

En primer lloc s'ha d'anar al directori on es troba per defecte l'arxiu de configuració del servidor. En el sistema operatiu en el que s'està treballant, aquest directori es sol ubicar a */usr/local/nginx/conf*

Una vegada dins, es procedeix a canviar les variables de l'arxiu anomenat *nginx.conf*, ja que aquí es on es guarda la informació bàsic del servidor.

A continuació s'especifica quines variables s'han establert dins el fitxer de configuració i quina funció te cadascuna en relació a la plataforma de VoD.

Un incís important per entendre millor com es reparteix la informació dins el fitxer de configuració és, que a l'hora d'executar-lo, hi ha algunes variables que depenen del servidor Nginx i altres que depenen del mòdul Kaltura, per això a continuació s'han separat depenent de a quin software fan referencia.

En el cas del servidor, la majoria de variables tenen a veure amb l'emmagatzematge de memòria cau. Aquesta configura una memòria on s'hi pot trobar descriptors de fitxers oberts, les seves mides i temps de modificació, informació sobre l'existència de directoris, errors de cerca de fitxers, com ara "fitxer no trobat" o "sense permís de lectura" i molta més informació què és útil per agilitzar el procés de carrega, ja que permet al client haver de fer tantes peticions al servidor.

En canvi, les variable de Kaltura, són les que configuren el mòdul de Vídeo sota Demanda i tenen més relació amb la segmentació de vídeo i configuracions del protocol HLS.

La informació sobre les variables es pot trobar a la pàgina de nginx.org [37]. Però a continuació s'expliquen les que són més importants per al servidor.

Es començarà l'explicació amb les variables corresponents al servidor Nginx.

**· Variables Nginx:**

`Client_max_body_size 128M` = Representa la mida de la petició per part del client. Si aquest supera el valor establert, es retorna un missatge d'error al client (413, Request Entity Too Large). Serveix per limitar la mida de càrrega de fitxers d'usuari a Nginx. La restricció de la mida de càrrega de fitxers és útil per evitar alguns tipus d'atacs de *denial-of-service* (DoS) i molts altres problemes relacionats.

Es pot configurar tant en la part de *http* per tal que afecti a tots els servidors, en la part de servidor per tal que afecti a un servidor en concret o a la part d'ubicació que afecta a un directori que es troba sota el lloc web.

`Access-Control-Allow-Origin * always` = D'habitual, quan s'accedeix a una pàgina web, no es pot carregar dades d'es d'un servidor extern, però, amb aquesta comanda s'indica al navegador que si que està permès per als orígens que

s'especifiquin, tant sigui un en concret, o acceptar qualsevol recurs amb el símbol d'asterisc (*) i, per tant, qualsevol domini pot carregar els recursos [38].

`Cache-Control no-cache always` = Indica a les memòries cau que aquest recurs no es pot reutilitzar sense comprovar prèviament si el recurs ha canviat al servidor d'origen. Això significa que la memòria cau farà un viatge al servidor per assegurar-se que la resposta no ha canviat i verificar si serà necessari descarregar el recurs o no.

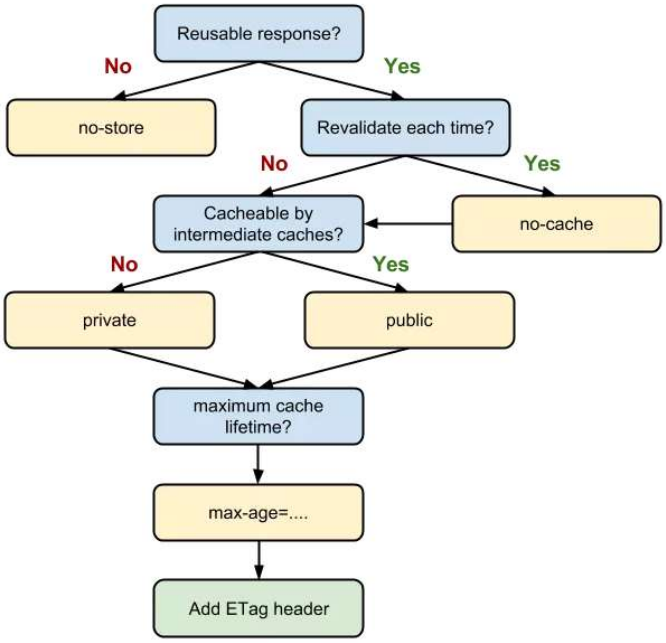



*Figura 4.1: Circuit que segueix la memòria cache. Imatge original de web.dev* [39].

`Open_file_cache max = 1000 inactive = 5m` = Configura dos paràmetres de la memòria cau.

El primer valor estableix el nombre màxim d'elements que es guarden, i en cas de desbordament, s'eliminen els LRU (Least Recently Used), que fa referencia als elements menys utilitzats recentment i per tant els que seran descartats quan la memòria s'ompli [40].

El segon valor defineix el temps després del qual es retira un element de la memòria si no s'hi ha accedit durant aquest temps.

Amb aquests dos valor es manté la memòria cau actualitzada i limitada en emmagatzematge.

`Open_file_cache_valid 2 m` = Estableix el temps després del qual s'han de validar els elements de la comanda *open_file_cache*.

`Open_file_cache_min_uses 1` = Estableix el nombre mínim d'accessos a fitxers durant el període configurat pel paràmetre inactiu de la comanda *open_file_cache*, necessari perquè un descriptor de fitxers romangui obert a la memòria cau.

`open_file_cache_errors on` = Habilita o deshabilita l'opció d'errors de cerca de fitxers de open_file_cache.

A continuació, es procedeix a explicar les variables que fan referència al mòdul de segmentació Kaltura.

**· Variables Kaltura:**

`vod_metadata_cache metadata_cache 2048m` = Configura la mida i el nom de l'objecte de memòria compartida de la memòria cau de metadades de vídeo. Per als fitxers MP4, aquesta memòria cau conté l'àtom moov.

`vod_response_cache response_cache 128m` = Configura la mida i el nom de l'objecte de memòria compartida de la memòria cau de respostes. La memòria cau de respostes conté manifestos i altres continguts que no són de vídeo (com ara el segment d'inici DASH, la clau de xifratge HLS, etc.). Els segments de vídeo no es guarden a la memòria cau.

`vod_last_modified_types *` = Estableix els tipus MIME per als quals s'hauria d'establir la capçalera de l'última modificació. El valor especial * coincideix amb qualsevol tipus MIME.

`vod_hls_master_file_name_prefix playlist` = El nom del fitxer de llista de reproducció principal HLS (està implícita una extensió m3u8).

`vod_segment_duration 6000` = Estableix la durada del segment en mil·lisegons i representa la duració dels segments individuals que s'envien al reproductor. Es recomana utilitzar una durada de segment que sigui múltiple a la durada del GOP. En cas que no fos múltiple i estigues activat *vod_align_segments_to_key_frames*, hi podria haver diferències significatives entre la durada del segment que s'informa al manifest i la durada del segment real. Això podria conduir a l'aparició de segments buits dins del flux i per tant errors de visualització en el vídeo. Apple recomana que aquests segments tinguin una duració de 6 segons, tal i com es pot veure a la seva web en el punt 7.5 ([41]).
Aquest valor és degut a que si els segments són molt petits, es requereix un número més gran de demandes per part de la web que el servidor ha de gestionar i per tant això gasta recursos. Per aquest motiu s'ha arribat a la conclusió que 6 segons permet una connexió persistent en qualsevol moment.

A continuació es mostra una gràfica que comprova que 6 segons és el valor ideal per a connexions no persistents i un valor correcte per a les connexions persistents.

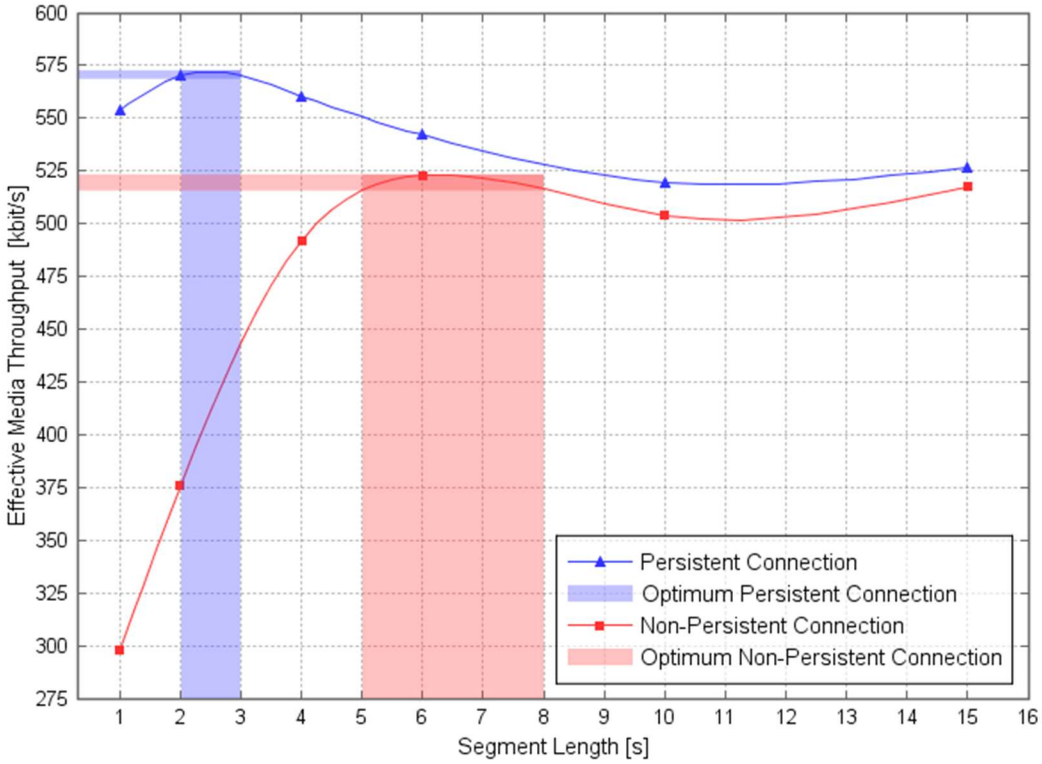



*Figura 4.2: Impacte de la mida del segment en el rendiment de la xarxa per a connexions persistents i no persistents. Imatge original de StreamingLearningCenter* [5].

`vod_manifest_segment_durations_mode accurate` = Configura el mode de càlcul de les durades del segment dins de les sol·licituds de manifest. El mode *accurate* informa de la durada exacta del segment, tenint en compte les durades del fotograma, p. per a una velocitat de fotogrames de 29,97 i 10 segons de segment, el primer segment serà el 10,01. el mode precís també té en compte l'alineació del marc clau, en cas que *vod_align_segments_to_key_frames* estigui activat

`vod_align_segments_to_key_frames on` = Quan està activat, el mòdul obliga a tots els segments a començar amb un marc clau (IDR).

## 4.4. Selecció i Codificació de les Variants

Una vegada ja s'ha configurat correctament el servidor de streaming, es procedeix a fer la codificació dels vídeos amb els paràmetres desitjats.

### 4.4.1. Normalització còpia màster

Primer de tot, quan es té la còpia màster del vídeo, s'ha de comprovar quines son les seves propietats per poder començar a treballar amb aquesta. Això és degut a que un arxiu màster pot estar configurat amb un còdec, un contenidor o un format determinat, però que a la hora de treballar amb ell, aquests paràmetres no seguin els desitjats. Per tant per poder procedir amb una correcta codificació, s'haurà primer de decidir les variables que es volen utilitzar.

Per això, el primer pas a fer amb qualsevol arxiu és veure les seves propietats i posteriorment convertir-ho al format que es desitgi. Aquest procés s'anomena normalització de vídeo i d'aquí sorgeix totes les futures variables i segments. Gràcies a aquest arxiu màster normalitzat s'estalvien problemes amb possibles arxius corruptes, ja que al tenir els mateixos formats tant a l'inici com al final del procés, és més complicat que sorgeixin problemes durant la codificació.

Durant aquest treball, tal i com s'ha explicat en la part teòrica, es vol treballar amb el còdec H.264 i amb el contenidor mp4 ja que són els més comuns i més acceptats per a la majoria de reproductors, per això, amb el vídeo que s'utilitza de prova (Big Buck Bunny), es comprovaran les seves característiques per veure la seva configuració de vídeo i àudio i, en cas que sigui necessari, realitzar els canvis oportuns.

Per poder veure les característiques d'un fitxer multimèdia es pot utilitzar la comanda mediainfo. I, respecte a l'arxiu de vídeo que es té, la comanda queda així:

```
--> mediainfo <arxiu>
```

```
mediainfo BBBunny.mp4
```

```
General
Complete name                    :          BBBunny.mp4
Format                           :          MPEG-4
File size                        :          263 MiB
Duration                         :          10m 34s
Overall bit rate                 :          3481 kbps
Movie name                       :          Big Buck Bunny, Sunflower version
Performer                        :          Blender Foundation 2008
Composer                         :          Sacha Goedegebure

Video
Format                           :          AVC
Format/Info                      :          Advanced Video Codec
Format profile                   :          High@L4
Format settings, CABAC           :          Yes
Format settings,ReFrames         :          4 frames
Codec ID                         :          avcl
Codec ID/Info                    :          Advanced Video Coding
Duration                         :          10m 34s
Bit rate                         :          3000 kbps
Maximum Bit Rate                 :          16.7 Mbps
Width                            :          1 920 pixels
Height                           :          1 080 pixels
Display aspect ratio             :          16:9
Frame rate mode                  :          Constant
Frame rate                       :          30.000 fps
Color space                      :          YUV
Chroma subsampling               :          4:2:0
Bit depth                        :          8 bits
Scan type                        :          Progressive
Bits/(Pixel*Frame)               :          0.048
Stream size                      :          227 MiB (86%)
Writing library                  :          x264 core 115
```

*Llista 4.1: Sortida de la comanda mediainfo amb la informació del vídeo original.*

Amb aquesta comanda es pot veure que l'arxiu màster ja està codificat amb un còdec apropiat i amb un format de contenidor que ja interessa per al que es necessita fer, per tant, la única variable que s'ha de canviar és el bitrate.

En el cas que el fitxer màster ben codificat d'origen, s'hauria de fer la comanda apropiada per determinar el còdec i contenidor. Per exemple:

```
ffmpeg -i vid_original.avi -c:v libx264 vid_master.mp4
```

On l'entrada es troba en format .avi i el còdec de vídeo que s'especifica a la sortida és l'H264.

Però, com aquestes dues variables ja son correctes, la comanda ffmpeg que s'ha aplicat és la següent:

```
ffmpeg -i BBBunny.mp4 -preset medium -maxrate 15m -bufsize 30m
-threads 0 -filter:v fps=30 BBBmaster.mp4
```

Els valors que s'han donat tenen la següent interpretació:

`preset mèdium` = Una configuració predeterminada (*preset*) és un conjunt d'opcions que proporcionen una certa velocitat de codificació a la relació de compressió. Un pre-ajust més lent proporcionarà una millor compressió. Per defecte, com més baix és el valor, més temps trigarà en codificar-se però millor serà el resultat. A continuació hi ha una gràfica per veure els valors que pren el temps de codificació respecte al bitrate utilitzant el còdec H264 [42]. Les unitats de bitrate de la gràfica estàn en kbps.

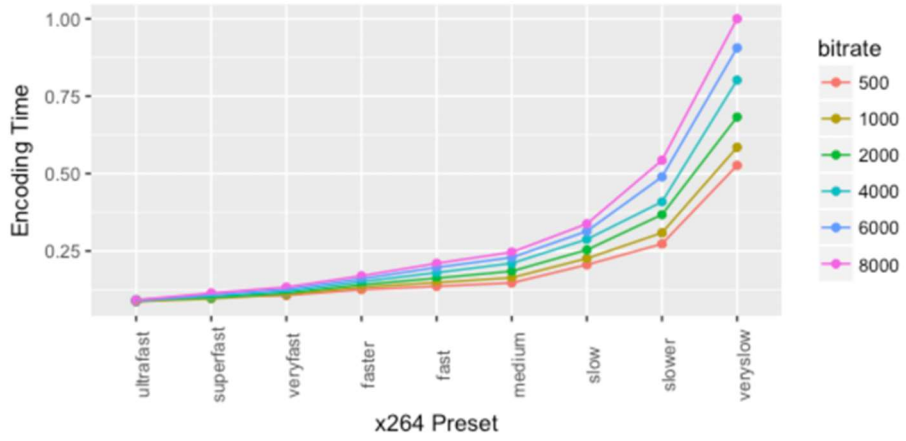



*Figura 4.3: Gràfica valorant el temps de codificació segons el preset seleccionat en la comanda ffmpeg i el còdec H.264. Imatge original de Ffmpeg* [42].

`maxrate 15m` = Es dona un valor màxim a la velocitat de bits, per tal que no es superi aquesta marca de 15Mbps, així es podrà seguir tenint un bitrate variable amb tap (Capped VBR) per evitar que no hi hagi un pic destacat en algun moment donat del vídeo.

`bufsize 30m` = Defineix la mida de la memòria intermèdia i pot ser de 1-2 segons per a la majoria d'escenes i de fins a 5 segons per als continguts més estàtics. Per tant, en aquest cas, com es vol una mida de 2 segons, s'ha de donar un valor que sigui el doble que el bitrate màxim. És a dir 30Mbps [43].

`threads 0` = El valor de *threads* (fils), ve determinat pel numero de fils que ffmpeg consumirà del processador. És a dir, aquesta comanda varia segons s'utilitza en un ordinador més potent o en un més senzill, i com més alt sigui el número, més recursos consumirà. Per defecte el valor és 22, però en molts casos, depenent de la qualitat de vídeo desitjada i els fotogrames per segon que es vulguin, un valor molt més baix serà suficient. Per exemple en un vídeo de 720:60p amb unes característiques determinades, es pot comprovar que a partir dels 8 threads ja no hi ha canvis i per tant es consumeix energia de més innecessària [44].

En aquest projecte, el valor es deixarà a 0. D'aquesta manera el propi software de codificació escull el valor més òptim segons les condicions de l'ordinador en el que es troba. Igualment, un valor correcte seria entre 4 i 8 fils.

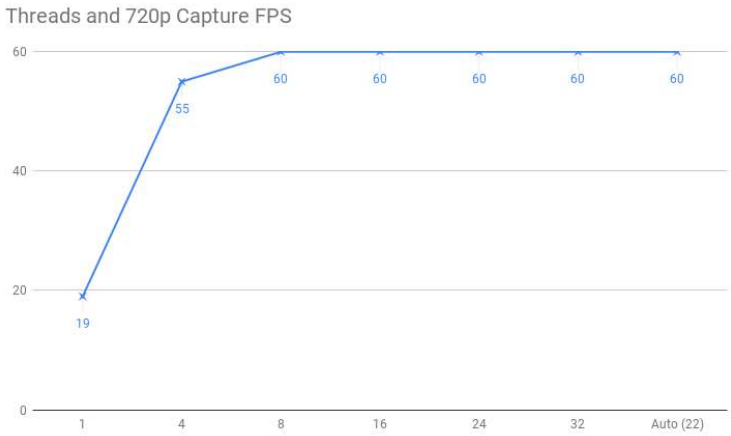



*Figura 4.4: Gràfica mostrant els fotogrames per segon que s'aconsegueixen segons el número de fils de processador que s'utilitzin durant la codificació. Imatge original de SLC* [44].

`-filter:v fps=30` = Obliga al vídeo a tenir un *frame rate* de 30. Així no hi ha problemes durant la codificació ja que s'ha establert un GOP de 60.

Per acabar, es torna a mirar quina és la configuració del fitxer de vídeo normalitzat, després d'haver realitzat la comanda anterior amb tots els canvis. Per fer-ho es torna a executar la comanda mediainfo

```
mediainfo BBBmaster.mp4
```

```
General
Complete name                    :          BBBmaster.mp4
Format                           :          MPEG-4
File size                        :          235 MiB
Duration                         :          10m 34s
Overall bit rate                 :          3110 kbps
Movie name                       :          Big Buck Bunny, Sunflower version
Performer                        :          Blender Foundation 2008
Composer                         :          Sacha Goedegebure

Video
Format                           :          AVC
Format/Info                      :          Advanced Video Codec
Format profile                   :          High@L4
Format settings, CABAC           :          Yes
Format settings,ReFrames         :          4 frames
Codec ID                         :          avcl
Codec ID/Info                    :          Advanced Video Coding
Duration                         :          10m 34s
Bit rate                         :          2973 kbps
Maximum Bit Rate                 :          15.0 Mbps
Width                            :          1 920 pixels
Height                           :          1 080 pixels
Display aspect ratio             :          16:9
Frame rate mode                  :          Constant
Frame rate                       :          30.000 fps
Color space                      :          YUV
Chroma subsampling               :          4:2:0
Bit depth                        :          8 bits
Scan type                        :          Progressive
Bits/(Pixel*Frame)               :          0.048
Stream size                      :          225 MiB (96%)
Writing library                  :          x264 core 148
```

*Llista 4.2: Sortida de la comanda mediainfo del arxiu de vídeo normalitzat.*

Com es pot veure en la última taula, l'arxiu màster normalitzat te ara uns valors diferents de bitrate que l'arxiu original però ha mantingut tant el còdec com el contenidor

Un detall important és el valor de les variables bitrate general i bitrate màxim. En el vídeo màster, la taxa de bits general era 3.481 kbps i la màxima 16.700 kbps. En canvi, en el vídeo normalitzat, el bitrate general és 3.110 kbps i el màxim 15.000 kbps. La diferència no és molt abismal, però demostra que durant la codificació s'han limitat aspectes que afecten a la qualitat de vídeo i per tant a la seva mida.

Amb aquest punt acaba la normalització del vídeo i ja es te disponible la còpia màster, per tant es pot procedir amb la codificació de les variants, tant utilitzant el mètode CBR com el mètode VBR.

### 4.4.2. Codificació de les variants

El següent pas a fer, una vegada es té el vídeo normalitzat, és la codificació per extreure les seves variants. L'objectiu en aquest treball és crear 4 variants principals de 4 resolucions diferents, de manera que aplicant la tècnica d'ABR mencionada anteriorment, a l'hora de reproduir el vídeo, el reproductor pugui escollir la que millor s'adapti al seu estat de la xarxa.

Per codificar existeixen varies tècniques i configuracions disponibles, en aquest cas s'ha utilitzat la codificació de dos passos, també coneguda com a codificació de diversos passos. Aquesta consisteix en una estratègia de codificació de vídeo que s'utilitza per conservar la millor qualitat durant la conversió.

Durant el primer pas de codificació, les dades d'entrada del vídeo d'origen s'analitzen i s'emmagatzemen en un fitxer de registre. A la segona passada, les dades recollides de la primera passada s'utilitzen per aconseguir la millor qualitat de codificació. A la codificació de vídeo, la codificació de dos passos normalment es controla mitjançant la configuració de la taxa de bits mitjana o mitjançant la configuració de l'interval de velocitat de bits (taxa de bits mínima i màxima permesa) o per la configuració de la mida del fitxer de vídeo objectiu, és a dir amb la codificació VBR. Aquesta tècnica de dues passades és gairebé dues vegades més lenta que la codificació d'un pas. Per tant, si es té poc temps, s'ha de seleccionar la solució d'un pas [45].

En aquest treball s'ha utilitzat aquesta tècnica i, després de la primera passada, s'obté un arxiu de sortida nul que conté valors de configuració de vídeo bàsics per a les variants i, a continuació, en la segona passada, agafant la informació que té l'arxiu de sortida anterior, es repeteixen alguns valors de codificació de vídeo per millorar la seva qualitat i s'afegeix la codificació d'àudio. Tot això dona com a resultat 4 arxius de sortida, ja que durant el procés s'especifica que es repetirà quatre vegades, cadascun amb una resolució i taxa de bits diferent.

Tant per la part de CBR, com la de VBR s'ha utilitzat aquesta tècnica de codificació i per tant la única diferència entre aquests és el valor de la taxa de bits, que en el primer cas ha de tenir un valor igual tant en el bitrate mínim, màxim i mitjà. I, en el segon cas, els valors han de ser distints per a les tres etiquetes. A continuació és mostra una part del script de codificació, de manera que es pugui veure quines son les etiquetes utilitzades i quina importància tenen per al resultat final. I més endavant, quins valors s'ha donat segons es desitjava obtenir una codificació constant o variable.

Aquesta és la primera part del script, amb la configuració general del vídeo:

```
FFMPEG_ADV_COD = "-c:a aac -ac 2 -strict -2 -b:a 128k -c:v
libx264 -pix_fmt yuv420p -preset fast -tune film -profile:v
main -b:v \$bitrate  -minrate \$bitrate -maxrate \$bitrate -
bufsize \$buffsize -s:v:0 \$size -sc_threshold 0 -g 60 -
keyint_min 30 -coder 1 -bf 3 -refs 4 -movflags +faststart -
map_chapters -1 -avoid_negative_ts 1 -shortest -vsync 1 -f
mp4"
```

Cada etiqueta utilitzada té el següent significat:

`c:v libx264` = Indica que s'ha d'utilitzar el còdec de vídeo H.264.

`pix_fmt yuv420p` = El format dels píxels és YUV4:2:0 progressiu.

`b:v` = Taxa de bits del fitxer de sortida.

`preset fast` = Proporciona la relació de velocitat de compressió de la codificació. S'ha explicat també en el punt 4.4.1, durant la normalització de vídeo.

`s:v:0` = Determina la mida del marc (resolució) en píxels.

bufsize = És un valor a partir del qual el còdec comprova el vídeo per si hi ha canvis i per si s'ha d'adaptar a un nou valor de bitrate. Idealment aquest ha de ser el doble que el valor de bitrate i així es fan les comprovacions cada 2 segons. [43]

`keyint_min 30` = Estableix l'interval de frames mínim entre marcs IDR.

`tune film` = Estableix el tipus de contingut. Pot tenir el valor per defecte, pantalla o pel·lícula. En aquest cas, s'utilitza l'últim.

`sc_threshold 0` = Estableix el llindar per la detecció de canvi d'escena. En aquest cas, es desactiva per evitar problemes en l'interval de fotogrames clau, tal i com s'explica a continuació.

`g 60` = Defineix la mida del Gop a 60 marcs, el doble del valor de keyint_min per així tenir un interval de 2 segons.

La raó d'això és que quan es codifica un vídeo per a una transmissió adaptativa, l'interval de fotogrames clau (keyframe) s'ha de dividir uniformement en la mida del segment per garantir un fotograma clau (IDR) al començament de cada segment. En conseqüència, com més gran sigui la mida del segment, més llarg serà l'interval de fotogrames clau potencial. És a dir, si la mida del segment és de dos segons, l'interval de fotograma clau més llarg que es pot suportar és de dos segons. Els intervals de fotogrames clau més llargs solen resultar en una qualitat de vídeo superior, perquè els fotogrames clau, que es codifiquen sense fer referència a cap altre fotograma, són el fotograma menys eficient. Tot i això, la diferència és bastant modesta per a la majoria de fitxers, tal com es pot veure a la següent gràfica, que mostra els valors PSNR dels fitxers 1080p codificats a la mateixa velocitat de bits.

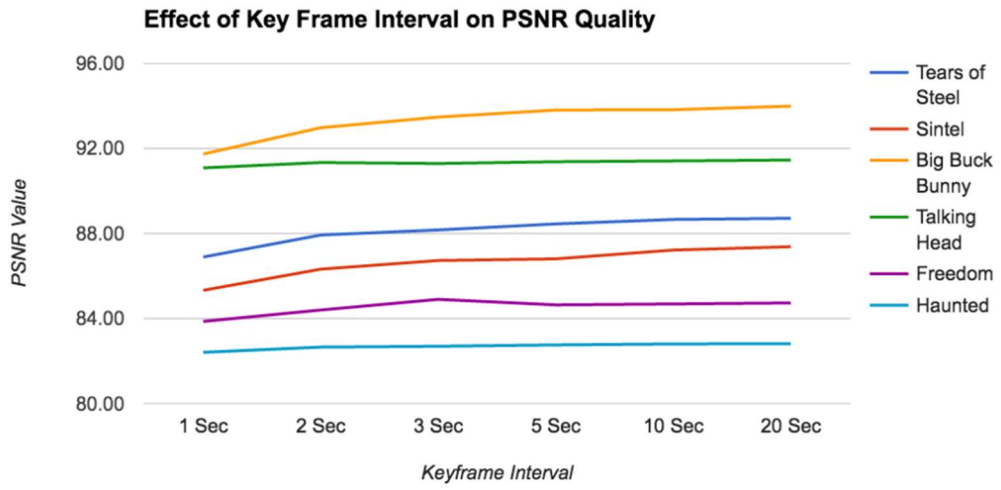



*Figura 4.5: Efecte de l'interval de fotogrames clau en relació al PSNR, avaluat en dB* [5].

Es mostra la diferència PSNR més gran entre els intervals d'un fotograma clau d'entre un i vint segons, però en el cas del vídeo Big Buck Bunny la diferència de qualitat només és del 2,4% en total. La majoria dels altres vídeos mostren molt menys diferencial, amb la major part de la diferència recuperada en canviar d'un segon a dos segons. Més enllà de dos segons, la pendent de millora es va aplanant i és casi impossible notar la diferència de qualitat entre dos i vint segons fins i tot per a l'espectador més exigent.

`bf 3` = Estableix el límit de marcs de tipus B.

`refs 4` = Una de les funcions més útils de H.264 és la capacitat per referenciar els marcs diferents del que és immediatament anterior al marc actual. Aquest paràmetre permet especificar quantes referències es poden utilitzar, fins a un màxim de 16. Augmentar el nombre de referències augmenta el requisit de DPB (Decoded Picture Buffer), el que significa que els dispositius de reproducció de maquinari sovint tenen límits estrictes al nombre de referències que poden gestionar. En les fonts d'acció real, hi ha més referències que tenen un ús limitat més enllà del 4-8, però sovint és útil en fonts de dibuixos animats fins al valor màxim de 16.

`movflags faststart` = Executa una segona passada movent l'índex al començament del fitxer. Aquesta operació pot trigar una estona i no funcionarà en diverses situacions, com ara la sortida fragmentada, de manera que no està habilitada per defecte.

`map_chapters -1` = Copia els capítols del fitxer d'entrada amb l'index input_file_index al següent fitxer de sortida. Si no s'especifica cap assignació de capítols, aquests es copien del primer fitxer d'entrada amb almenys un capítol. I, com en aquest cas, si s'utilitza un índex de fitxers negatiu, serveix per desactivar la còpia de capítols.

`avoid_negative_ts 1` = Aquest valor significa *make_non_negative*, és a dir que es canviïn les marques de temps a un valor positiu.

`Shortest` = Indica que s'acabi el procés de codificació quan s'acaba el flux d'entrada més curt.

`vsync 1` = Mètode de sincronització de vídeo. Quan el valor és 1 indica que s'accepta *HTML5* i mètodes *Flash*.

`f mp4` = Força el format d'entrada o de sortida.

A continuació és mostra la codificació de dues passades, que es realitza just després de la comanda anterior. Aquí es pot veure com en la primera passada només es tracta el vídeo i que l'arxiu de sortida te un valor nul i com la segona passada tracta tant el vídeo com l'àudio i ja inclou l'arxiu de sortida final amb l'etiqueta "*pass 2*".

```
FFMPEG_CODING_PASS1 = "-c:v libx264 -pix_fmt yuv420p -
profile:v baseline -preset veryslow -tune film -g 60 -s \$size
-b:v \$bitrate -maxrate \$bitrate -bufsize \$buffsize -pass 1
-f null /dev/null"


FFMPEG_CODING_PASS2 = "-c:a aac -ac 2 -b:a 128k -c:v libx264
-c:v libx264 -pix_fmt yuv420p -profile:v baseline -preset
veryslow -tune film -g 60 -s \$size -b:v \$bitrate -maxrate
\$bitrate -bufsize \$buffsize -pass 2"
```

Per tant, una vegada s'ha entès com està configurat l'arxiu de codificació, es procedeix a la seva execució.

Per fer-ho, s'ha d'estar en tot moment amb privilegis de super usuari.

Dins la carpeta del servidor Nginx, es crea un nou directori (en cas que no estigui creat prèviament) per guardar els vídeos. Aquest directori ha de rebre el nom que s'hagi assignat en l'arxiu de configuració del servidor:

Dins d'aquest directori es crearan les variants de l'arxiu normalitzat. Per fer-ho tant amb la tècnica de CBR com la de VBR, la comanda necessària és la següent, i realitzar-ho trigarà un cert temps, depenent de la potencia de l'ordinador:

```
./encode_CBR.sh BBBmaster.mp4
./encode_VBR.sh BBBmaster.mp4
```

Les diferències entre el dos arxius executables són les següents:

### *CBR*

En el cas de l'arxiu *encode_CBR.sh*, on es vol una taxa de bits constant, les variables de bitrate, minrate i maxrate hauran de tenir el mateix valor. D'aquesta manera s'obté una taxa de bits igual tant per inferiorment com superiorment, mantenint sempre el mateix nivell.

Els valors seleccionats per a cada resolució són els que es mostren a la Taula 3.1.

### *Capped VBR*

En canvi, en l'arxiu *encode_VBR.sh*, on es vol una taxa de bits variable, s'ha de donar un valor diferent a cadascuna de les variables mencionades anteriorment.

Els valors que s'han establert per a la variable de bitrate, són els que es mostren a la Taula 3.1, on hi ha cada resolució amb el seu valor corresponent

I, en el cas de la variable maxrate, que determina el bitrate màxim, s'ha donat un valor igual al doble que el bitrate.

I l'etiqueta del bitrate mínim té un valor establert de 200kbps per a qualsevol de les resolucions.

Amb aquesta configuració s'aconsegueix tenir per a cada resolució una taxa de bits diferents. De manera que es podrà operar amb el protocol ABR per tenir una reproducció adaptativa al client.

Una vegada es tenen codificades les variables, es pot procedir a fer les proves del sistema, enviant una petició per part del client i una resposta amb el vídeo seleccionat per part del servidor.

## 4.5. Client

Com s'ha vist en l'esquema de l'inici, es necessita un mètode per entregar el contingut multimèdia del servidor al client. Hi ha diferents mètodes: Per exemple un seria utilitzar una terminal i amb una comanda com ffplay iniciar la reproducció. Un altre mètode és amb un reproductor capaç de demanar contingut per *http*, fer la petició, per exemple VLC. O un altre mètode, què és el que s'ha fet servir en aquest treball és utilitzar *Javascript* (JS) des d'un navegador web per veure el contingut del servidor.

Això es fa mitjançant codi HTML i *Javascript* emmagatzemat en el servidor que permet incrustar els vídeos a partir d'extensions *javascript*.

Per tant, per comprovar des de la banda del client, s'utilitza el següent codi JS i HTML per fer les proves des de un navegador:

```
<video id="video" controls=""  width="426" height="240">
</video>
<script>
if(Hls.isSupported()) {
var video = document.getElementById('video');
var hls = new Hls();

hls.loadSource('http://192.168.1.140/vodhls/BBBmaster_VBR_,0.
8,1.2,2.5,5,M.mp4.urlset/playlist.m3u8');
     hls.attachMedia(video);

hls.on(Hls.Events.MANIFEST_PARSED,function() {
     video.play();
     });
}
</script>
```

*Llista 4.3: Arxiu test-web.html per carregar el vídeo a la web.*

El primer punt és una etiqueta vídeo, que representa el marc en el que es reproduirà en la pàgina web el vídeo. Aquí se li dona una mida determinada per evitar que s'escali en la web.

A continuació, amb codi JS, es carrega el recurs del vídeo, enviant les 4 variables que es troben a la carpeta *media* des de la xarxa local en format m3u8.

I, seguidament, s'incrusta aquest recurs a l'etiqueta vídeo creada amb *HTML* per poder-ho reproduir.

Amb aquest codi ja es té la pàgina web disponible pe reproduir el vídeo des de un navegador extern que estigui connectat a la xarxa.

En el treball, la comanda necessària a fer per part del client és la següent:

```
http://192.168.1.140/test-web.html
```

Aquí es mostra com el client es connecta via http al servidor amb IP *192.168.1.140* i fa la petició d'un document anomenat *test-web.html* que conté el codi mostrat prèviament.

# 5. Proves i Resultats

Finalment, amb el sistema ja configurat i les variants del vídeo ben codificades, es procedeix a fer les proves necessàries per veure com funciona el servidor, quan hi ha una petició d'un client.

El procés de proves es divideix en dues parts. La primera part es tracta de reproduir el vídeo des d'un navegador extern amb unes condicions estàtiques de xarxa, és a dir amb un valor d'ample de banda constant, i la segona part amb unes condicions variables de xarxa a mesura que el reproductor de vídeo es troba en funcionament per comprovar si la tècnica ABR actua correctament i s'adapta de manera eficient als valors de bitrate establerts durant la codificació.

Ambdues proves es realitzaran amb el vídeo codificat amb CBR i en VBR, de manera que es pugui determinar quin dona millor resultat i és més òptim per al VoD.

## 5.1. Condicions estàtiques de xarxa

Per fer-ho s'utilitzarà la comanda *wondershaper*, que permet donar un valor de carrega i descàrrega determinat a la interfície de xarxa que s'especifiqui.

Per tant, abans de començar a utilitzar aquesta comanda, s'ha de comprovar quines són les interfícies de xarxa connectades a la màquina mitjançant alguna comanda, per exemple, *ifconfig*. A partir d'aquí, ja es pot conèixer en quina superfície es troba el servidor i per tant en quina es vol configurar l'ús de l'amplada de banda:

En el cas d'aquest treball la xarxa del servidor rep el nom *enp0s3*.

Per tant ara ja es pot procedir a fer les proves de xarxa, donant en una primera instància un valor constant i en segon lloc un valor variable.

Com a primera prova, s'executarà la comanda amb un valor constant de 4000 kbps tant de pujada com de baixada a la xarxa a la que pertany el servidor. La comanda queda d'aquesta manera.

```
wondershaper enp0s3 4000 4000
```

Una vegada limitat, es procedeix a la prova de reproducció utilitzant un navegador extern, accedint al fitxer *HTML* creat prèviament. Alhora, utilitzant la terminal, es visualitza el document acces.log ubicat dins el servidor per veure com es realitzen les peticions de les variants amb els seus segments per part del client.

Des del navegador extern, s'accedeix a la URL que conté l'arxiu de prova HTML i que es troba ubicat dins la carpeta *html* del servidor:

```
http://192.168.1.140/test-web.html
```

Des de la terminal del servidor de streaming, s'analitza el fitxer de registre:

```
cd /usr/local/nginx/logs
tail -f accés.log
```

El resultat obtingut del registre d'entrada és el següent:

```
[10:18:23] GET /vodhls/bbb_master_VBR_2.5M.mp4/index-v1-a1.m3u8
[10:18:23] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-1-v1-a1.ts
[10:18:23] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-2-v1-a1.ts
[10:18:23] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-3-v1-a1.ts
[10:18:23] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-4-v1-a1.ts
[10:18:23] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-5-v1-a1.ts
[10:18:29] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-6-v1-a1.ts
[10:18:35] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-7-v1-a1.ts
[10:18:41] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-8-v1-a1.ts
...
```

*Llista 5.1: Sortida de la comanda accés.log a mesura que es reprodueix el vídeo en un cercador extern amb la banda limitada a 4000 kbps. Condicions estàtiques*

Gràcies al fitxer de registre, es pot veure com donades les condicions de xarxa estàtiques, el reproductor decideix agafar una resolució de 720p que equival a un taxa de bits de 2,5M, doncs l'ample de banda no pot sobrepassar els 4M. Quan, en condicions òptimes de la xarxa, el reproductor hauria d'escollir la màxima resolució disponible, en aquest cas 1080p amb una taxa de bits de 5M.

També, aquesta llista mostra cada quan s'envia un segment al client. A l'inici de tot, es descarrega l'arxiu manifest i un conjunt de segment de cop. Després, a partir de la configuració que s'ha fet en aquest treball, s'envia un segment nou cada 6 segons al client amb la part de vídeo que correspongui.

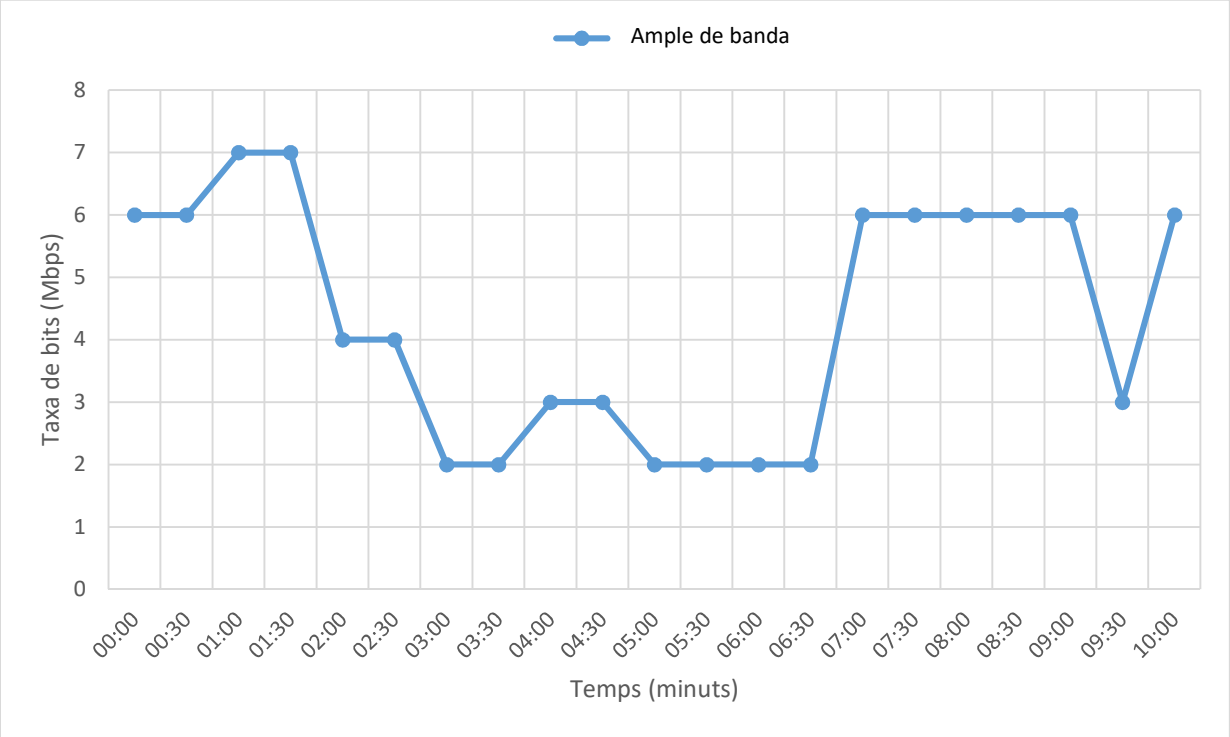
Amb aquesta prova bàsica, es comprova que amb condicions de xarxa amb un valor constant, el reproductor escull correctament la resolució màxima. Però ara, s'ha de veure que passa quan la xarxa te un valor diferent durant la reproducció del vídeo.

## 5.2. Condicions variants de xarxa

El següent pas es provar si variant l'estat de la xarxa durant la reproducció el sistema canvia de qualitat de vídeo i no entra en un estat d'espera, de manera que el client pugui veure el vídeo complet sense pauses i amb la millor qualitat possible.

Aquest és el cas més comú en la majoria de clients, doncs l'estat de la xarxa en qualsevol lloc sol ser variant, sobretot si s'utilitza una connexió mitjançant *wi-fi* i per tant qualsevol reproductor hauria de poder adaptar-se a aquestes senyals sense problemes.

Per realitzar aquestes provés s'ha creat un arxiu executable (*script*) que permet automatitzar la comanda *wondershaper* i donar valors prèviament establerts cada cert temps. D'aquesta manera, el que s'ha fet és canviar el valor d'ample de banda cada 1 minut per veure com reacciona el servidor i quan triga en donar l'ordre de canvi de taxa de bits al client. Aquest script varia el seu valor d'ample de banda seguint el patró que es veu en la següent gràfica. Els valors establert es troben dins els marges de desenvolupament de la plataforma. És a dir que si la taxa de bits màxima a la que es pot reproduir el vídeo és 5Mbps, l'ample de banda no tindrà un valor molt per sobre. En l'exemple que es descriu, el màxim es troba en 7Mbps.




La comanda necessària per executar el script és la següent:

```
./test-bw.sh
```

Des del navegador extern, es torna a accedir a la URL que conté l'arxiu de prova HTML:

```
http://192.168.1.140/test-web.html
```

I, des de la terminal del servidor, es torna a analitzar l'arxiu de registre:

```
cd /usr/local/nginx/logs
tail -f accés.log
```

El resultat obtingut en l'arxiu accés.log en els primers instants de vídeo, és el següent:

```
[18:44:37] GET /vodhls/bbb_master_VBR_5M.mp4/index-v1-a1.m3u8
[18:44:37] GET /vodhls/bbb_master_VBR_5M.mp4/seg-1-v1-a1.ts
[18:44:37] GET /vodhls/bbb_master_VBR_5M.mp4/seg-2-v1-a1.ts
[18:44:37] GET /vodhls/bbb_master_VBR_5M.mp4/seg-3-v1-a1.ts
[18:44:37] GET /vodhls/bbb_master_VBR_5M.mp4/seg-4-v1-a1.ts
[18:44:37] GET /vodhls/bbb_master_VBR_5M.mp4/seg-5-v1-a1.ts
[18:44:43] GET /vodhls/bbb_master_VBR_5M.mp4/seg-6-v1-a1.ts
[18:44:49] GET /vodhls/bbb_master_VBR_5M.mp4/seg-7-v1-a1.ts
[18:44:55] GET /vodhls/bbb_master_VBR_5M.mp4/seg-8-v1-a1.ts
...
```

*Llista 5.2: Sortida de la comanda accés.log a mesura que es reprodueix el vídeo en un cercador extern amb la banda limitada a 5000 kbps. Condicions variables*

I, a partir dels dos minuts de vídeo, com que el script test-bw.sh va variant el valor d'ample de banda, l'arxiu de peticions té aquesta forma:

```
[18:47:13] GET /vodhls/bbb_master_VBR_5M.mp4/seg-31-v1-a1.ts
[18:47:19] GET /vodhls/bbb_master_VBR_5M.mp4/seg-32-v1-a1.ts
[18:47:25] GET /vodhls/bbb_master_VBR_2.5M.mp4/index-v1-a1.m3u8
[18:47:25] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-33-v1-a1.ts
[18:47:31] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-34-v1-a1.ts
[18:47:37] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-35-v1-a1.ts
[18:47:43] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-36-v1-a1.ts
[18:47:49] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-37-v1-a1.ts
[18:47:55] GET /vodhls/bbb_master_VBR_2.5M.mp4/seg-38-v1-a1.ts
...
```

*Llista 5.3: Sortida de la comanda accés.log a mesura que es reprodueix el vídeo en un cercador extern amb la banda limitada a 4000 kbps. Condicions variables*

Com es mostra en la Llista 5.2 i en la Llista 5.3, on es veu l'arxiu de peticions del servidor, hi ha dos conceptes importants.

El primer es troba en la primera línia de la

Llista 5.2, que correspon a la descarrega de l'arxiu manifest d'aquella resolució. Aquest arxiu és sempre el primer en aparèixer, ja que com s'ha explicat en la part de tòria, és

on hi ha la informació de les variables, els segments i cada quan canviar d'un a l'altre. Per tant, quan hi ha un canvi de resolució per primera vegada, també s'hi troba el fitxer manifest corresponent, com es pot veure en la tercera línia de la

Llista 5.3. D'aquesta manera el reproductor coneix quan ha de fer el canvi d'un segment a l'altre de la nova resolució.

I, el segon concepte, es pot veure en la resta de línies, ja que a mesura que es reprodueix el vídeo, el reproductor va descarregant segments de la qualitat màxima permesa per l'ample de banda cada 6 segons, tal i com està configurat en el punt 4.3.3. I, per tant, quan hi ha un canvi d'ample de banda, el client demana una resolució més baixa ja que la màxima no la pot suportar. En aquest exemple s'ha fet un canvi de valor en l'ample de banda i s'ha establert una velocitat de 4Mbps, per tant el client demana una qualitat de vídeo amb una taxa de bits inferior a aquest valor. El que vindria a ser 2.5Mbps d'acord amb les resolucions disponibles en aquest treball, ja que la següent, que es troba un esgraó per sobre, té un valor de 5Mbps.

Al final del script de prova d'amplada de banda, es torna a l'estat per defecte. És a dir, es crida la comanda *wondershaper clear*. Per això, els últims minuts de vídeo es poden tornar a visualitzar en la màxima qualitat (5M).

Amb aquestes proves, es comprova que el sistema fa un bon ús de la xarxa disponible i s'adapta de manera correcte als canvis de velocitat. De manera que el client pugui visualitzar el vídeo sense haver de tenir moments d'espera, i oferint sempre la qualitat màxima de vídeo disponible.

Però, encara no s'ha vist quin sistema te un millor rendiment, si el CBR o el VBR. Per tant ara, es procedeix a fer una comprovació sobre l'impacte que té la codificació en CBR i VBR Capped i si afecten a la qualitat de visualització quan es varia l'ample de banda.

## 5.3. Comparació CBR i VBR Capped

Per veure quina tècnica de codificació és més bona per a VoD, tot i que teòricament, cop ja s'ha dit en el punt 2, per a Vídeo sota Demanda és millor utilitzar VBR Capped en comptes de CBR. S'ha de mirar si a la pràctica és així o no hi ha una diferencia molt notable en utilitzar una tècnica o l'altre.

Cal precisar que totes les proves que es realitzen són seguint un únic sistema de VBR Capped, el que s'explica a la part teòrica. Però a la pràctica, aquesta codificació te paràmetres que poden ser canviats i adaptats a les necessitats que es requereixin i obtenir resultats diferents.

Una vegada s'ha aclarit aquest matis, el primer pas a fer és tenir els arxius en un mateix format per a poder comparar-los. Per tant, primerament, s'ha de convertir la playlist que es genera automàticament amb la comanda ffmpeg, és a dir la que s'envia des del servidor al client i que es troba en format m3u8, a un format que serveixi compatible per fer comparacions i gràfiques, per exemple mp4.

Per aconseguir fer aquesta conversió, s'utilitza la següent comanda en el cas de CBR:

```
sudo ffmpeg -i
http://localhost/vodhls/BBBmaster_CBR_,0.8,1.2,2.5,5,M.mp4.urlset/playlist.m3u8 -c copy -bsf:a aac_adtstoasc outputCBR.mp4
```

I, el mateix per al vídeo què és VBR Capped:

```
sudo ffmpeg -i
http://localhost/vodhls/BBBmaster_VBR_,0.8,1.2,2.5,5,M.mp4.urlset/playlist.m3u8 -c copy -bsf:a aac_adtstoasc outputVBR.mp4
```

Una vegada ja es tenen els dos vídeos en un format compatible per a la comparació, es procedeix a veure les propietats internes de cadascun i comparar-les.

Per fer-ho es procedirà a realitzar 3 tècniques diferents. En primer lloc s'analitzarà el Bitrate, en segon lloc la capacitat d'adaptació a l'ample de banda, és a dir la tècnica ABR, i per últim es calcularà el valor de *PSNR* per tenir una referència numèrica de qualitat.

### 5.3.1. Bitrate Viewer

Amb l'aplicació *Bitrate Viewer* es comparen els dos arxius convertits prèviament: El de sortida del servidor amb codificació CBR i el de sortida del servidor amb codificació VBR Capped. D'aquesta manera es veu com actua el bitrate en cada moment del vídeo i es compren millor cada tècnica.

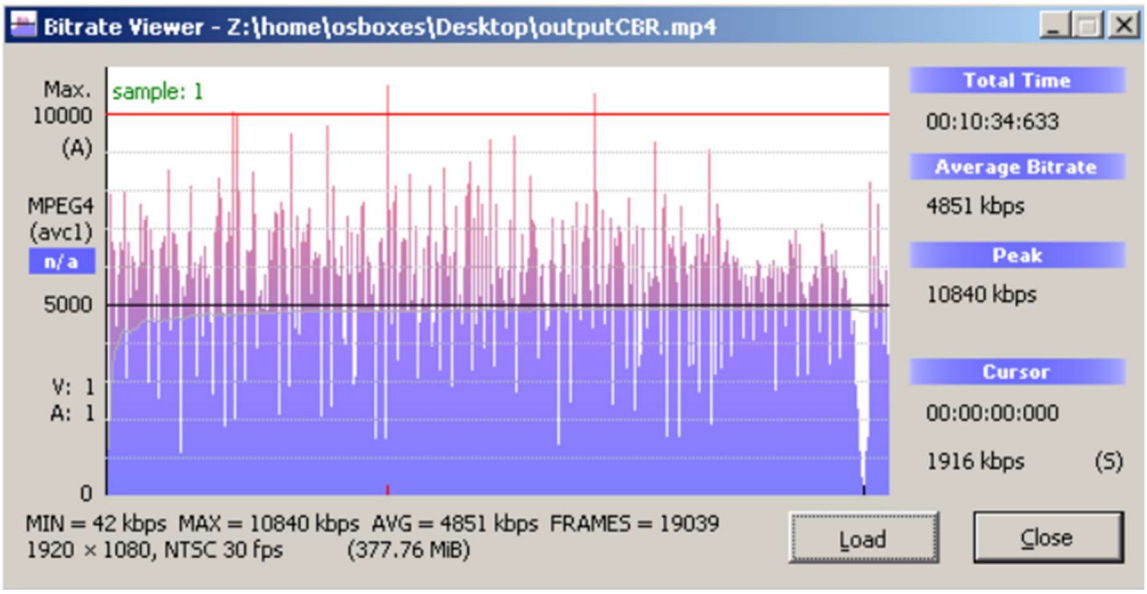



*Figura 5.1: Bitrate Viewer amb arxiu de vídeo utilitzant CBR.*

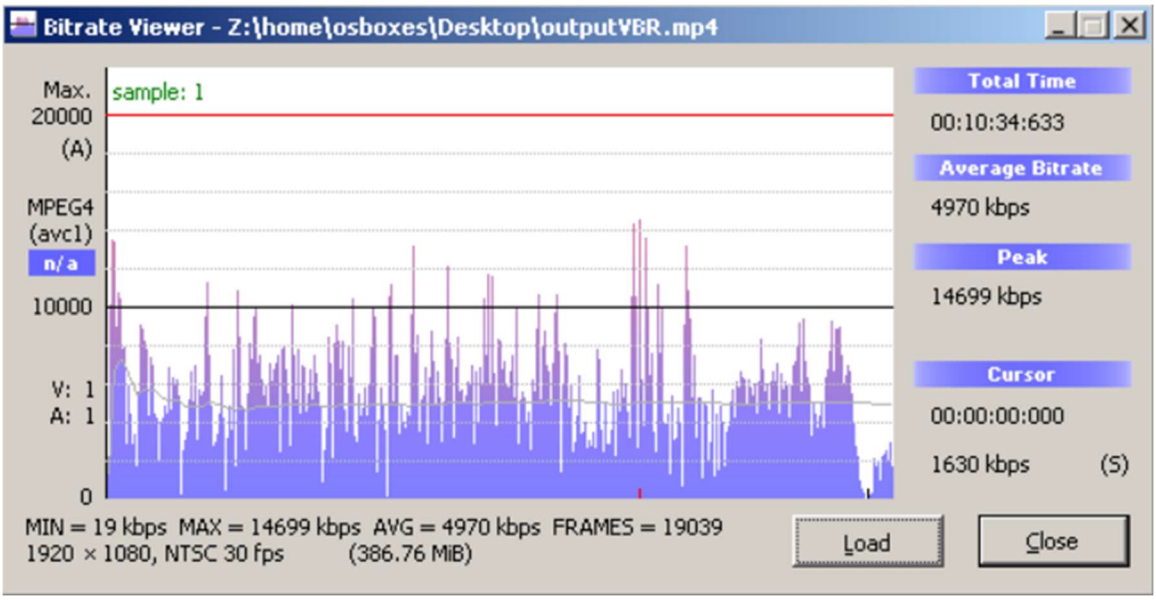



*Figura 5.2: Bitrate Viewer amb arxiu de vídeo utilitzant VBR Capped.*

La principal diferència al veure aquestes dues figures és bastant evident. En la Figura 5.1, la gràfica té una forma més constant que en la Figura 5.2. Costa una mica d'apreciar però en cada gràfica hi ha una línia gris que representa el nivell mig de taxa de bits durant tot el vídeo. En el primer cas aquesta línia és gairebé totalment recta, en canvi en el segon cas la línia té altures diferents segons l'escena en la que es troba, ja que en la codificació VBR, com s'ha explicat prèviament, la taxa de bits s'adapta a l'escena i permet destinar més recursos a aquells moments amb més informació, en canvi per a escenes més estàtiques dona un valor de taxa de bits molt més petit.

Per exemple, en escenes molt fosques on no cal tenir un nivell de detall molt gran, els recursos que es destinen son inferiors i, de cara al client comú, el canvi de qualitat és casi inapreciable. D'aquesta manera s'aconsegueix tenir un arxiu de dimensions semblants al que està codificat amb CBR però amb una millor qualitat.

Com es pot veure en la mateixa gràfica, hi ha un pic molt gran de 15 Mbps, ja que en aquell instant, el codificador ffmpeg ha decidit que s'ha de destinar més recursos que en la resta de vídeo. En canvi, en la gràfica de sortida CBR, el pic màxim és només de 10,8 Mbps. Ja que no diferencia entre escenes, i dona un valor més constant en tots els instants.

Tot això implica que per a una mateixa quantitat d'emmagatzematge, la tècnica VBR Capped permet tenir una millor qualitat ja que aquelles escenes que son més complexes poden tenir més recursos. En canvi, amb la tècnica CBR això no és així, i es destina el mateix per a tot el vídeo.

La tècnica CBR el que fa és variar la qualitat de vídeo per adaptar el seu nivell de taxa de bits a un valor constant. Per tant el que fa és variar el seu pas de quantificació, mentre que en el cas de VBR Capped, aquest es manté constant i varia el bitrate. Això implica que en algunes escenes el vídeo en CBR tingui una qualitat inferior a la del vídeo en VBR Capped.

L'avantatge de CBR sobre VBR Capped és la facilitat de preservació de recursos. És a dir, la tècnica VBR destina més bits a les escenes que ho requereixin. Però si en un moment donat, 5 pel·lícules emmagatzemades en el servidor necessiten entregar una taxa de bits elevada als seus clients, el servidor es pot col·lapsar. En canvi, amb CBR, això no pot passar ja que sempre té un nivell constant i hauria d'haver-hi un número molt gran de peticions per donar problemes.

Ara que s'ha entès com es comporta el bitrate en cadascun dels vídeos, es procedeix a veure com actua cadascun durant les proves de variació d'ample de banda mentre s'estan reproduint en el navegador del client, és a dir que es posarà a prova la tècnica ABR.

### 5.3.2. Tècnica ABR amb condicions variants de xarxa

Com s'ha explicat en el punt 5.2, durant les proves de condicions variables de la xarxa, s'ha realitzat el mateix procés amb el vídeo codificat amb CBR i amb el vídeo codificat amb VBR Capped.

I per mostrar el resultat s'han realitzat dues gràfiques, una per a cada tècnica de codificació.

En cada gràfica hi ha dos elements: En l'eix X s'hi troba l'instant de temps del vídeo i en l'eix Y s'hi troba la velocitat d'ample de banda. I com a dades resultants, hi ha dues línies que son variants en el temps i representen l'ample de banda disponible en la xarxa i la taxa de bits que reprodueix el vídeo. D'aquesta manera es pot veure com actua el servidor i quan temps passa des de que el client fa una petició de canvi de resolució fins que el servidor respon amb aquest canvi.

A continuació es mostren les dues gràfiques. La primera correspon al vídeo en VBR Capped i la segona al vídeo en CBR.

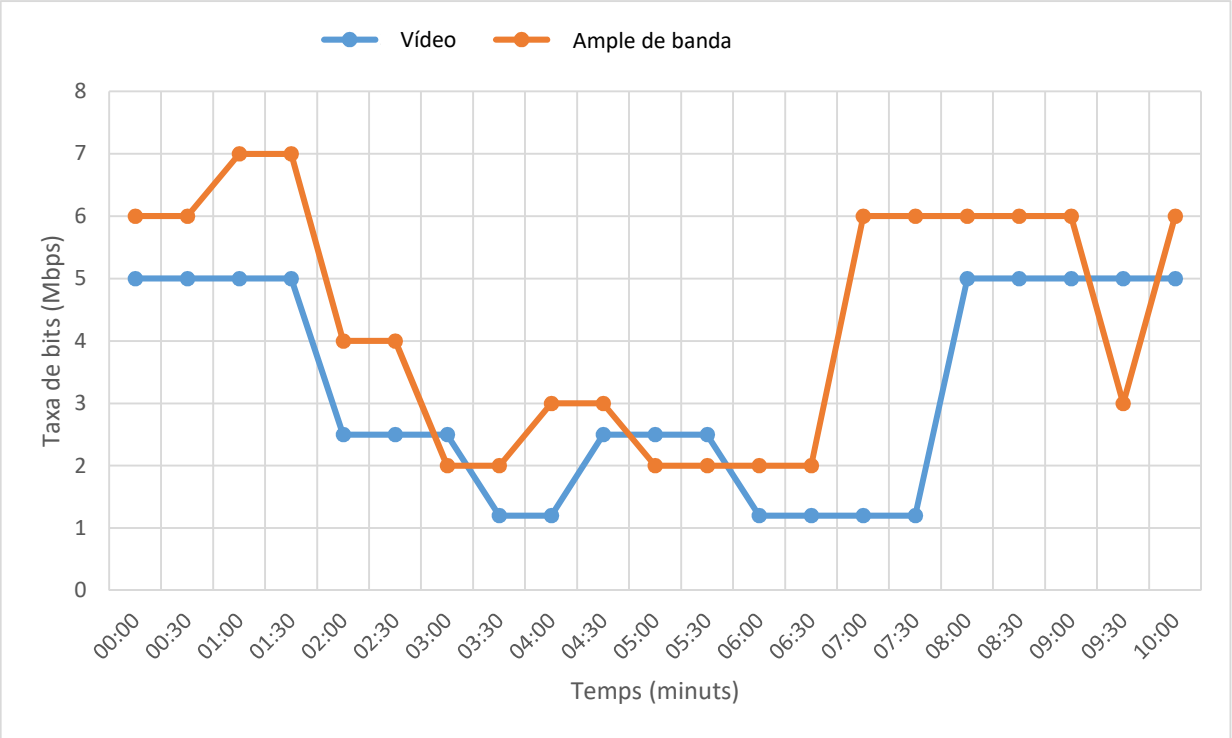



*Figura 5.3: Gràfica mostrant tècnica ABR quan l'ample de banda és variant amb vídeo codificat en VBR_Capped.*

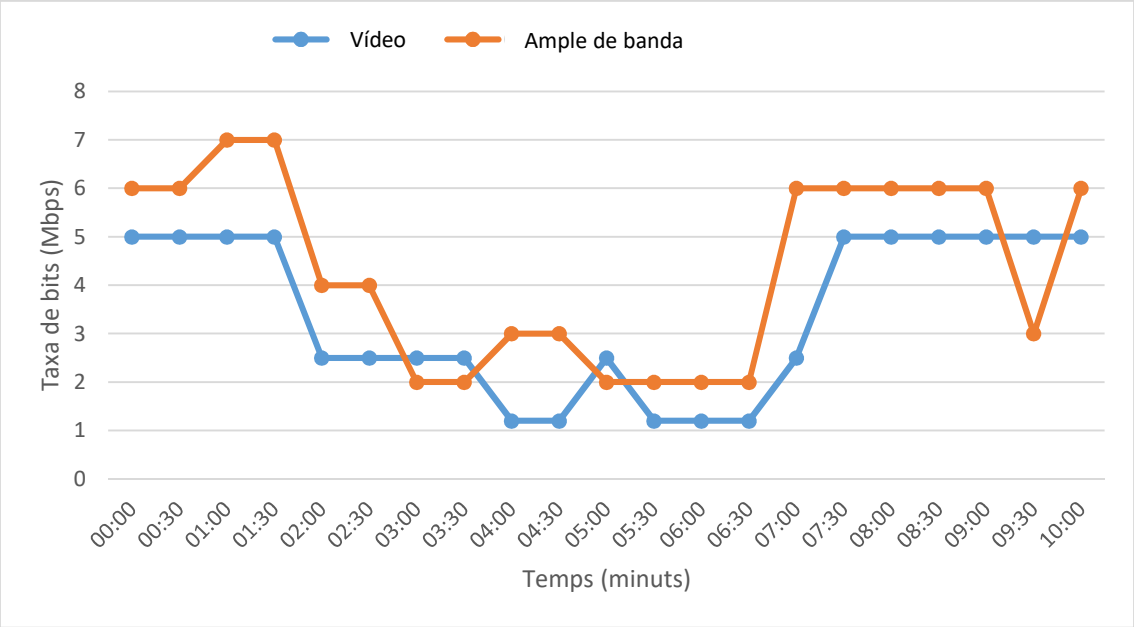



*Figura 5.4: Gràfica mostrant tècnica ABR quan l'ample de banda és variant amb vídeo codificat en CBR.*

Com es pot comprovar, la diferencia entre aquestes dues gràfiques no és molt notable.

En ambdós casos, el vídeo es correspon bastant a l'ample de banda disponible, seguint els seus valors. I, quan el client fa una petició nova (quan el valor d'ample de banda canvia) el servidor entrega primer els segmentes emmagatzemats en el buffer i després respon amb la taxa de bits nova.

Hi ha alguns moments en que els valors del vídeo no son tan fidels als de l'amplada de banda. Sobretot on hi ha canvis grans de taxa de bits.

Cap al final del vídeo, s'ha fet un canvi molt dràstic  en l'ample de banda per veure com actuava el sistema. En cap dels dos casos el vídeo s'ha vist afectat, doncs com hi havia informació emmagatzemada en el buffer no ha donat temps a reacció.

Un detall important durant aquestes peticions és el retard. Com es pot veure en les gràfiques prèvies, hi ha canvis de variants cada més o menys 1 minut o 1 minut i mig. Tot i que en el script de prova, el canvi de velocitat d'amplada de banda s'ha fet cada 1 minut exactament. Això és degut a que el reproductor té una memòria intermèdia anomenada buffer, que emmagatzema uns segments de més dels que està reproduint. De manera que, quan hi ha un canvi en la velocitat de connexió, en el buffer hi ha aproximadament un parell o tres de segments de la qualitat anterior que s'entreguen al client, mentre que el servidor fa el canvi per adaptar-se a la nova taxa de bits. Aquest valor de buffer varia segons la tècnica que s'utilitza per part del client. Però en aquest treball no s'ha entrat més en detall sobre aquesta informació i el seu funcionament.

Per tant, amb aquestes proves es demostra que per a la realització de la tècnica ABR, la codificació amb VBR Capped té un rendiment semblant a la codificació amb CBR. Això significa, que durant el procés d'entrega, l'elecció de VBR o CBR no és tan important com durant en el procés de codificació.

### 5.3.3. PSNR

Per acabar, s'utilitzarà una tècnica per mesurar la qualitat del vídeo rebut per part del client. Aquesta tècnica es coneix amb un terme que compara la relació entre la màxima potència possible d'un senyal i la potència de soroll que afecta la fidelitat de la seva representació. Aquest terme es conegut com a PSNR (Peak Signal-to-Noise Ratio), i s'utilitza molt com a mesura quantitativa de qualitat de reconstrucció a l'hora de comprimir imatges o vídeos.

Com que molts senyals tenen un rang dinàmic molt ampli, el PSNR sol expressar-se com una quantitat logarítmica mitjançant l'escala de decibels.

Es considera que un valor de PSNR de 45 dB o més és excel·lent per tenir una bona qualitat de vídeo. Com més alt és aquest valor, millor és la qualitat, però la diferència que s'aprecia cada vegada és menor. Valors que envolten els 38 dB es consideren correctes. Un PSNR de 30 dB ja comença a tenir un nivell de qualitat molt just. I, tots els valors que vinguin per sota, es consideren una mala qualitat de vídeo.

Per tant, com més alt sigui el valor de PSNR, millor serà la qualitat del vídeo de sortida ([46]). I, el comitè MPEG, determina un llindar informal de 0,5 dB en l'increment de l'PSNR per decidir si hi ha una determinada millora en l'algoritme de codificació, ja que es considera que aquest augment de l'PSNR és apreciable visualment [47].

Per poder veure el PSNR entre dos entrades de vídeo s'ha de fer la següent comanda en el cas del vídeo en CBR:

```
ffmpeg -i outputCBR.mp4 -i BBBmaster.mp4 -filter_complex
"psnr" -f null /dev/null
```

I la següent comanda per al vídeo en VBR Capped:

```
ffmpeg -i outputVBR.mp4 -i BBBmaster.mp4 -filter_complex
"psnr" -f null /dev/null
```

On en cada cas es compara l'arxiu de sortida obtingut en el punt 5.3, que es correspon amb el vídeo que s'envia des del servidor al client, amb el vídeo màster normalitzat, ja que aquest és el vídeo de màxima qualitat i per tant el que s'agafa com a referència.

En el cas del vídeo de sortida amb CBR, el valor de PSNR és el següent:

```
PSNR y:44.26 u:51.22 v:52.22 average:45.61 min:31.61
```

En canvi, el valor per al vídeo de sortida amb VBR Capped, és el següent:

```
PSNR y:45.55 u:52.16 v:52.97 average:46.87 min:37.26
```

Una vegada més, es demostra que el vídeo codificat amb VBR Capped ofereix un millor rendiment que el codificat amb CBR. La diferència entre ambdós és de 1,26 dB de mitja. I, seguint les instruccions del comitè MPEG, és un valor bastant significatiu i notable en l'apreciació humana.

Per tant, segons el resultat d'aquesta prova, es reafirma la premissa de que la codificació amb VBR Capped té millor rendiment que la codificació amb CBR en quan a qualitat percebuda per un espectador de VoD amb l'aplicació de la tècnica ABR.

També, aquests resultats, mostren que la qualitat en ambdós casos és semblant tot i que en VBR és veu millor. Per tant la pregunta important que un s'ha de fer a l'hora d'escollir un tipus de codificació és si aquesta diferència de qualitat, que es mostra en fotogrames puntuals li mereix la pena, tenint en compte les avantatges i desavantatges que cadascuna d'aquestes codificacions presenta.

# Conclusions

A l'inici d'aquest treball, es va establir un objectiu que es podria dividir en 3 parts i les quals s'han pogut realitzar correctament.

En primer lloc hi ha la creació de l'entorn de treball, la qual ha consistit en crear un servidor utilitzant les eines *open-source* que hi ha disponibles actualment de manera que qualsevol persona pugui realitzar el mateix procés. Com s'ha presentat durant tot el treball aquest procés consta de tres programaris principals, Nginx com a servidor, Kaltura com a eina de segmentació i ffmpeg com a codificador, que funcionen sobre un sistema Unix. L'entorn implementat permet controlar amb comoditat la gestió dels vídeos i la configuració de xarxa per a les proves que s'han realitzat.

En segon lloc hi ha la part de codificació. Aquí és on s'ha realitzat la normalització del vídeo per obtenir la còpia màster i s'han creat les variants que posteriorment s'enviaran per part del servidor. Aquest punt ha complert el seu objectiu, ja que amb els programaris anteriors es pot personalitzar pràcticament tots els paràmetres del vídeo de manera còmode.

Per últim, l'apartat de proves i resultats. On, utilitzant el servidor prèviament configurat, els vídeos codificats i un navegador extern per fer les proves de reproducció, s'ha comprovat de manera pràctica el que s'havia explicat a la part teòrica. D'aquesta manera s'ha confirmat que un sistema de VoD te un millor rendiment quan la seva configuració de taxa de bits és variable amb límit en comptes de treballar amb una configuració amb una taxa de bits constant. I també s'ha pogut veure com el còdec H.264 treballa de manera molt eficient juntament amb el protocol HLS per a transmissions adaptatives.

En aquest treball s'ha pogut crear una eina que permet comparar vídeos amb diferents valors i variables de codificació, de manera que a través de la informació dels arxius i els seus gràfics és pugui escollir la configuració més òptima en funció del tipus de vídeo i inclòs de la connexió a internet, 4G, W-Fi, satèl·lit, fibra, etc.

Dit això i veient que els resultats obtinguts són positius, hi ha moltes coses que encara es podrien realitzar per acabar de completar els conceptes d'aquest treball i fer-ho un nivell més extensiu.

Primerament, es podria afegir opcions noves de configuració de distribució del sistema per donar un ventall més ampli als clients que utilitzaran el servidor. Per exemple, en aquest treball s'ha utilitzat el protocol de streaming HLS, però n'hi ha d'altres com seria el MPEG-DASH (Dynamic Adaptative Streaming over HTTP) o el RTSP (Real Time Streaming Protocol), cadascun amb les seves avantatges i inconvenients però perfectament funcionals.

Una altra millora seria la implementació d'un protocol de comunicació per Internet segur, és a dir l'HTTPS (HyperText Transfer Protocol Secure) que serveix per protegir la integritat i les dades dels usuaris entre els ordinadors i les pàgines web. Amb aquest protocol s'integren característiques de xifrat, integritat de dades i autenticació.

En tercer lloc, com a proposta de millora i d'ampliació del treball, es troba la incorporació d'una CDN (Content Delivery Network) per distribuir de manera eficaç el contingut multimèdia a usuaris repartits per tot el món. D'aquesta manera les peticions que es reben al servidor són analitzades segons la seva localització i es busca el CDN més

proper per evitar problemes de latència. Amb això s'estalvia temps d'espera, alliberament del servidor i estalvi d'ample de banda [48].

Per últim, com a millora per al futur, es recomana ampliar el ventall de proves en l'extrem del client. És a dir, incorporar nous client al sistema per veure com actua el servidor quan té moltes peticions, buscar noves maneres de fer peticions (no només a través d'http) i així extreure conclusions sobre a quin tipus de clients el sistema ofereix un millor serveix i sota quines condicions es troben.

Per acabar amb l'apartat de conclusions, s'ha vist que en els últims anys, els serveis de VoD han crescut de manera exponencial, i que s'espera que en els propers anys sigui creixent. Per tant es pot afirmar que aquest camp no ha arribat al seu límit i que hi haurà moltes serveis que s'optimitzaran i de nous que apareixeran, oferint més alternatives al mercat i abaratint els costos de subscripció. El que està clar, és que el camí del VoD no s'acaba aquí i segueix tenint molt futur i per tant molta feina a realitzar.

# Bibliografia


[1] "¿Qué son las plataformas OTT y qué debemos saber? - Megamedia." https://megamedia.es/producto-digital/plataformas-ott/ (accessed Aug. 02, 2021).

[2] "Las plataformas de vídeo 'on demand' superarán los 400 millones de usuarios en 2020 | Transformación Digital | Tecnología | EL PAÍS." https://elpais.com/retina/2019/10/28/tendencias/1572257138_509867.html (accessed Jul. 16, 2021).

[3] "Las plataformas de streaming superan ya los 1.000 millones de usuarios a nivel global y amenazan a la televisión tradicional." https://www.puromarketing.com/18/35037/%3C!doctype html%3E%3Chtml lang= (accessed Jul. 25, 2021).

[4] "¿Qué es una CDN?" https://www.akamai.com/es/our-thinking/cdn/what-is-a-cdn (accessed Sep. 01, 2021).

[5] "Choosing the Segment Duration for DASH or HLS - Streaming Learning Center." https://streaminglearningcenter.com/blogs/choosing-the-optimal-segment-duration.html (accessed Sep. 18, 2021).

[6] "2020, el año del streaming: así está cambiando el mundo." https://www.elperiodico.com/es/activos/innovadores/20201028/2020-el-ano-del-streaming-asi-esta-cambiando-el-mundo-esta-tendencia-8176452 (accessed Sep. 18, 2021).

[7] "Adaptive Bitrate Streaming (ABR) | Video Streaming Definition." .

[8] "Adaptive bitrate streaming HLS VOD service in NodeJS | by gaurav sharma | revamp | Medium." https://medium.com/sharma02gaurav/adaptive-bitrate-streaming-hls-vod-service-in-nodejs-8df0d91d2eb4 (accessed Sep. 01, 2021).

[9] "Adaptive streaming video infrastructure encoding and HTML5 player - Bitmovin." https://www.bitmovin.com/ (accessed Sep. 24, 2021).

[10] "HLS vs. MPEG-DASH: A Live Streaming Protocol Comparison for 2021." https://www.dacast.com/blog/mpeg-dash-vs-hls-what-you-should-know/ (accessed Sep. 01, 2021).

[11] "¿Qué es el streaming en HTTP? | Streaming HLS | Cloudflare." .

[12] "Cómo usar URL de lista de reproducción 3U con IPTV." .

[13] "M3U - Wikipedia, la enciclopedia libre." .

[14] "Structure and chunk your video lecture content - LX at UTS." https://lx.uts.edu.au/collections/creating-and-managing-videos/resources/structure-chunk-video-lecture-content/ (accessed Jun. 16, 2021).

[15] "Beginner Guide to Video Streaming - AT&T Developer." https://developer.att.com/blog/video-streaming-adaptive-bitrate (accessed Apr. 26, 2021).

[16] "Creating a Primary Playlist | Apple Developer Documentation." https://developer.apple.com/documentation/http_live_streaming/example_playlists_for_http_live_streaming/creating_a_primary_playlist (accessed Aug. 03, 2021).

[17] "Còdec de vídeo - Viquipèdia, l'enciclopèdia lliure."

https://ca.wikipedia.org/wiki/Còdec_de_vídeo (accessed Jun. 17, 2021).

[18] “¿Qué es un Codec? - Tecnología Fácil.” https://tecnologia-facil.com/que-es/que-es-un-codec/ (accessed Jun. 17, 2021).

[19] “¿Qué es y para qué sirve H264? | Streaming.com.co.” .

[20] “HEVC/H.265 video format | Can I use... Support tables for HTML5, CSS3, etc.” https://caniuse.com/hevc (accessed Sep. 27, 2021).

[21] “Effective Use of Long GOP Video Codecs | The Tiliam Blog.” http://tiliam.com/Blog/2015/07/06/effective-use-long-gop-video-codecs/ (accessed Sep. 01, 2021).

[22] “Everything You Ever Wanted to Know About IDR Frames but Were Afraid to Ask - Streaming Learning Center.” https://streaminglearningcenter.com/encoding/everything-you-ever-wanted-to-know-about-idr-frames-but-were-afraid-to-ask.html (accessed Sep. 18, 2021).

[23] “Open and Closed GOPs - All You Need to Know - Streaming Learning Center.” https://streaminglearningcenter.com/blogs/open-and-closed-gops-all-you-need-to-know.html (accessed Aug. 03, 2021).

[24] “What is the Optimal Bitrate for Your Resolution? – Teradek.” https://teradek.com/blogs/articles/what-is-the-optimal-bitrate-for-your-resolution (accessed Apr. 26, 2021).

[25] “Tested: Lowering Bandwidth at Night is Good.” https://ipvm.com/reports/low-light-bandwidth-compression-test (accessed Sep. 18, 2021).

[26] “Nuevo bitrate 4K de Netflix: se ve peor y enfada a los clientes.” https://www.adslzone.net/noticias/streaming-tv/nuevo-bitrate-netflix-peor-calidad-septiembre-2020/ (accessed Sep. 27, 2021).

[27] “Video Encoding Settings for H.264 Excellence.” http://www.lighterra.com/papers/videoencodingh264/ (accessed Apr. 26, 2021).

[28] “Video Encoding Settings for H.264 Excellence.” http://www.lighterra.com/papers/videoencodingh264/ (accessed Apr. 17, 2021).

[29] “HLS Authoring Specification for Apple Devices | Apple Developer Documentation.” https://developer.apple.com/documentation/http_live_streaming/hls_authoring_specification_for_apple_devices (accessed Aug. 10, 2021).

[30] “HLS Authoring Specification for Apple Devices | Apple Developer Documentation.” https://developer.apple.com/documentation/http_live_streaming/hls_authoring_specification_for_apple_devices#//apple_ref/doc/uid/TP40016596-CH4-SW1 (accessed May 18, 2021).

[31] “Ficheros de vídeo: diferencia entre códec y contenedor.” https://www.wikiversus.com/informatica/codec-vs-contenedor/ (accessed Jun. 16, 2021).

[32] “‘mp4’ | Can I use... Support tables for HTML5, CSS3, etc.” https://caniuse.com/?search=mp4 (accessed Sep. 27, 2021).

[33] “Big Buck Bunny » Download.” https://peach.blender.org/download/ (accessed Aug. 09, 2021).

[34] “Nginx - Wikipedia.” https://en.wikipedia.org/wiki/Nginx (accessed Jun. 09, 2021).

[35] “FFmpeg - Wikipedia.” https://en.wikipedia.org/wiki/FFmpeg (accessed Jun. 10, 2021).

[36] “kaltura/nginx-vod-module: NGINX-based MP4 Repackager.” https://github.com/kaltura/nginx-vod-module (accessed Jun. 17, 2021).

[37] “Module ngx_http_core_module.” http://nginx.org/en/docs/http/ngx_http_core_module.html#client_max_body_size (accessed May 13, 2021).

[38] “¿Qué es el CORS? Te explicamos el cross-origin resource sharing - IONOS.” https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/cross-origin-resource-sharing/ (accessed Aug. 19, 2021).

[39] “Prevent unnecessary network requests with the HTTP Cache.” https://web.dev/http-cache/ (accessed Sep. 01, 2021).

[40] “Oracle LRU & MRU Algorithm - Laymen’s Perspective - Nimesa.” https://nimesa.io/blogs/oracle-lru-mru-algorithm-laymen/ (accessed Aug. 20, 2021).

[41] “HLS Authoring Specification for Apple Devices | Apple Developer Documentation.” https://developer.apple.com/documentation/http_live_streaming/hls_authoring_specification_for_apple_devices#//apple_ref/doc/uid/TP40016596-CH4-SW1 (accessed Sep. 18, 2021).

[42] “Encode/H.264 – FFmpeg.” https://trac.ffmpeg.org/wiki/Encode/H.264 (accessed Aug. 08, 2021).

[43] “EncodingForStreamingSites – FFmpeg.” https://trac.ffmpeg.org/wiki/EncodingForStreamingSites (accessed Aug. 08, 2021).

[44] “FFmpeg Threads Command: How it Affects Quality and Performance - Streaming Learning Center.” https://streaminglearningcenter.com/blogs/ffmpeg-command-threads-how-it-affects-quality-and-performance.html (accessed May 24, 2021).

[45] “What is two pass encoding? - Encoding.com HelpEncoding.com Help.” https://help.encoding.com/knowledge-base/article/what-is-two-pass-encoding/ (accessed Sep. 04, 2021).

[46] “Peak signal-to-noise ratio - Wikipedia.” https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio (accessed Sep. 18, 2021).

[47] “Computation of SNR and PSNR.” https://web.archive.org/web/20071026050307/http://bmrc.berkeley.edu/courseware/cs294/fall97/assignment/psnr.html (accessed Sep. 27, 2021).

[48] “Utilizar una CDN: ¿Qué ventajas tiene para su sitio web? - OVH.” https://www.ovh.es/cdn/ventajas.xml (accessed Sep. 23, 2021).

# Annex

## Arxiu codificació CBR

```
#!/bin/bash

if [ -z "$1" ]; then
   echo  "$0 File.mp4"
   exit 1
fi

if ! [ -f "$1" ]; then
   echo "file $1 not found"
   exit 1
fi



STREAM[1]="1920x1080 4992K 5M"
STREAM[2]="1280x720 2496K 2.5M"
STREAM[3]="848x480 1216K 1.2M"
STREAM[4]="640x360 896K 0.8M"



FFMPEG_ADV_COD="-c:a aac -ac 2 -strict -2 -b:a 128k -c:v libx264 -pix_fmt
yuv420p -preset fast -tune film -profile:v main -b:v \$bitrate  -minrate
\$bitrate -maxrate \$bitrate -bufsize \$buffsize -s:v:0 \$size -
sc_threshold 0 -g 60 -keyint_min 30 -coder 1 -bf 3 -refs 4 -movflags
+faststart -map_chapters -1 -avoid_negative_ts 1 -shortest -vsync 1 -f mp4"


FFMPEG_CODING_PASS1="-c:v libx264 -pix_fmt yuv420p -profile:v baseline -
preset veryslow -tune film -g 60 -s \$size -b:v \$bitrate -maxrate
\$bitrate -bufsize \$buffsize -pass 1 -f null /dev/null"


FFMPEG_CODING_PASS2="-c:a aac -ac 2 -b:a 128k -c:v libx264  -c:v libx264 -
pix_fmt yuv420p -profile:v baseline -preset veryslow -tune film -g 60 -s
\$size -b:v \$bitrate -maxrate \$bitrate -bufsize \$buffsize -pass 2"
```

```
outname=$(basename $1)
outname="${outname%%.mp4}_CBR"


for arr_idx in $(/usr/bin/seq 1 ${#STREAM[@]})
do
    size=$(echo ${STREAM[$arr_idx]} | cut -f1 -d " ")
    bitrate=$(echo ${STREAM[$arr_idx]} | cut -f2 -d " ")
    bitrateVal=$(echo $bitrate | sed -e 's/\(^[0-9]\+\).*/\1/')
    bitrateUnit=$(echo $bitrate | sed -e 's/\(^[0-9]\+\)\(.*\)/\2/')
    buffsize="$((bitrateVal * 2))$bitrateUnit"
    suffix=$(echo ${STREAM[$arr_idx]} | cut -f3 -d " ")

    # AVC CODING
    codparms=$(eval echo "$FFMPEG_ADV_COD")
eval ffmpeg -y -hide_banner -i $1 $codparms ${outname}_${suffix}.mp4

    sleep 1
done
```

## Arxiu codificació VBR Capped

```
#!/bin/bash

if [ -z "$1" ]; then
   echo  "$0 File.mp4"
   exit 1
fi

if ! [ -f "$1" ]; then
   echo "file $1 not found"
   exit 1
fi



STREAM[1]="1920x1080 4992K 5M"
STREAM[2]="1280x720 2496K 2.5M"
STREAM[3]="848x480 1216K 1.2M"
STREAM[4]="640x360 896K 0.8M"



FFMPEG_ADV_COD="-c:a aac -ac 2 -strict -2 -b:a 128k -c:v libx264 -pix_fmt
yuv420p -preset fast -tune film -profile:v main -b:v \$bitrate  -minrate
200k -maxrate \$maxrate -bufsize \$buffsize -s:v:0 \$size -sc_threshold 0 -
g 60 -keyint_min 30 -coder 1 -bf 3 -refs 4 -movflags +faststart -
map_chapters -1 -avoid_negative_ts 1 -shortest -vsync 1 -f mp4"


FFMPEG_CODING_PASS1="-c:v libx264 -pix_fmt yuv420p -profile:v baseline -
preset veryslow -tune film -g 60 -s \$size -b:v \$bitrate -maxrate
\$maxrate -bufsize \$buffsize -pass 1 -f null /dev/null"


FFMPEG_CODING_PASS2="-c:a aac -ac 2 -b:a 128k -c:v libx264  -c:v libx264 -
pix_fmt yuv420p -profile:v baseline -preset veryslow -tune film -g 60 -s
\$size -b:v \$bitrate -maxrate \$maxrate -bufsize \$buffsize -pass 2"
```

```
outname=$(basename $1)
outname="${outname%%.mp4}_VBR"


for arr_idx in $(/usr/bin/seq 1 ${#STREAM[@]})
do
    size=$(echo ${STREAM[$arr_idx]} | cut -f1 -d " ")
    bitrate=$(echo ${STREAM[$arr_idx]} | cut -f2 -d " ")
    bitrateVal=$(echo $bitrate | sed -e 's/\(^[0-9]\+\).*/\1/')
    bitrateUnit=$(echo $bitrate | sed -e 's/\(^[0-9]\+\)\(.*\)/\2/')
    maxrate="$((bitrateVal * 2))$bitrateUnit"
    buffsize="$((bitrateVal * 3))$bitrateUnit"
    suffix=$(echo ${STREAM[$arr_idx]} | cut -f3 -d " ")

    # AVC CODING
    codparms=$(eval echo "$FFMPEG_ADV_COD")
    eval ffmpeg -y -hide_banner -i $1 $codparms ${outname}_${suffix}.mp4

    sleep 1
done
```

## Arxiu test-web.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
            <meta charset="UTF-8">
                <title>Live Streaming</title>
    </head>
    <body>


        <script src="https://cdn.jsdelivr.net/npm/hls.js@latest"></script>
        <video id="video" controls=""  width="426" height="240"></video>
        <script>
if(Hls.isSupported()) {
        var video = document.getElementById('video');
            var hls = new Hls();

hls.loadSource('http://192.168.1.140/vodhls/BBBmaster_VBR_,0.8,1.2,2.5,5,M.
mp4.urlset/playlist.m3u8');
                    hls.attachMedia(video);
                    hls.on(Hls.Events.MANIFEST_PARSED,function() {
                              video.play();


                    });
}
        </script>


    </body>
</html>
```

## Arxiu de configuració del Servidor

```
pid /run/nginx.pid;
worker_processes 1;

user root root;

events {
    worker_connections 1024;
}



http {
    include       mime.types;
    default_type application/octet-stream;
    sendfile        on;
    keepalive_timeout 65;

    server {
        listen 80;
       # server_name pav.upc.edu;



        #client_max_body_size 128M;

        add_header Access-Control-Allow-Origin * always;
        add_header Cache-Control no-cache always;


       #vod caches
        vod_metadata_cache metadata_cache 2048m;
        vod_response_cache response_cache 128m;
         vod_last_modified_types *;
        vod_hls_master_file_name_prefix playlist;
```

```
# vod variables
          open_file_cache max=1000 inactive=5m;
          open_file_cache_valid 2m;
          open_file_cache_min_uses 1;
          open_file_cache_errors on;

        # vod sites for adaptive VOD streaming



        # vod share
          location /vodhls/ {
             vod hls;
             vod_mode local;
             # vod_segment_duration: multiple of the GOP duration in
miliseconds
                vod_segment_duration 6000;
             vod_align_segments_to_key_frames on;
              # vod_manifest_segment_durations_mode accurate;

             alias /usr/local/nginx/media/;
             gzip on;
             gzip_types application/vnd.apple.mpegurl;
             vod_last_modified_types *;
             add_header Access-Control-Allow-Headers 'origin,range,accept-
encoding,referrer';
             add_header Access-Control-Expose-Headers 'Server,range,Content-
Length,Content-Range';
             add_header Access-Control-Allow-Methods 'GET, HEAD, OPTIONS';
             add_header Access-Control-Allow-Origin '*';
           add_header Cache-Control no-cache always;
             expires 100d;
             add_header Last-Modified "Sun, 19 Nov 2000 08:52:00 GMT";
          }
```

```
location /vod_status/ {
            vod_status;
         }



        # Main Document Root
        location / {
            root   html;
            index test-web.html test-web.htm;
        }
    }
}
```

## Arxiu de configuració del mòdul de Kaltura

```
# internal location for vod subrequests
          location /kalapi_proxy/ {
               internal;
               proxy_pass https://kalapi/;
               proxy_set_header Host $http_host;
          }

# base locations
include /opt/kaltura/nginx/conf/base.conf;

# serve flavor progressive
location ~ ^/p/\d+/(sp/\d+/)?serveFlavor/ {
vod none;

     directio 512;
     output_buffers 1 512k;

     include /opt/kaltura/nginx/conf/cors.conf;
}

# serve flavor HLS
location ~ ^/hls/p/\d+/(sp/\d+/)?serveFlavor/ {
     vod hls;
     vod_bootstrap_segment_durations 2000;
     vod_bootstrap_segment_durations 2000;
     vod_bootstrap_segment_durations 2000;
     vod_bootstrap_segment_durations 4000;

     include /opt/kaltura/nginx/conf/cors.conf;
}
```

```
# serve flavor DASH
location ~ ^/dash/p/\d+/(sp/\d+/)?serveFlavor/ {
      vod dash;
      vod_segment_duration 4000;
      vod_dash_manifest_format segmenttemplate;
      vod_manifest_duration_policy min;

      include /opt/kaltura/nginx/conf/cors.conf;
}

# serve flavor HDS
location ~ ^/hds/p/\d+/(sp/\d+/)?serveFlavor/ {
      vod hds;
      vod_segment_duration 6000;
      vod_segment_count_policy last_rounded;

      include /opt/kaltura/nginx/conf/cors.conf;
}

# serve flavor MSS
location ~ ^/mss/p/\d+/(sp/\d+/)?serveFlavor/ {
      vod mss;
      vod_segment_duration 4000;
      vod_manifest_segment_durations_mode accurate;

      include /opt/kaltura/nginx/conf/cors.conf;
}

# static files (crossdomain.xml, robots.txt etc.) + fallback to api
           location / {
      root   /opt/kaltura/nginx/static;
      try_files $uri @api_fallback;
}
```

```
# all unidentified requests fallback to api (inc. playManifest)
location @api_fallback {
     proxy_pass https://kalapi;
     proxy_set_header Host $http_host;
}

location /dashme {
        open_file_cache off;
        root /var/tmp;
        add_header Cache-Control no-cache;
        # To avoid issues with cross-domain HTTP requests (e.g. during
development)
        add_header Access-Control-Allow-Origin *;
}
location /hlsme {
        open_file_cache off;
        types {
                application/vnd.apple.mpegurl m3u8;
        }
        root /var/tmp;
        add_header Cache-Control no-cache; # Prevent caching of HLS
fragments
        add_header Access-Control-Allow-Origin *; # Allow web player to
access our playlist
}
```