SAE 5.ROM.03

Projet : Serveur MPEG-DASH

Sommaire

Introduction

Le terme MPEG-DASH désigne une méthode de diffusion. L'acronyme DASH signifie ici « Dynamic Adaptive Streaming over HTTP » (diffusion dynamique à débit adaptatif sur HTTP). Comme ce format se fonde sur le protocole HTTP, n'importe quel serveur d'origine peut être configuré de manière à diffuser des flux MPEG-DASH. Le format MPEG-DASH est similaire au HLS (un autre protocole de diffusion) en ce qu'il décompose les vidéos en petits morceaux et encode ces morceaux à différents niveaux de qualité. Cette façon de faire permet de diffuser les vidéos à différents niveaux de qualité et de passer d'un niveau de qualité à un autre en cours de lecture.

Compatibilité

Le format MPEG-DASH s'appuie sur le protocole HTTP. Il s'agit là d'un avantage considérable, car la plus grande partie d'Internet utilise déjà ce protocole. Lors d'une diffusion en HTTP, le flux se dirige vers un port standard (le port 80 ou 443), presque toujours ouvert. Ceci permet de garantir que le flux se retrouve rarement intercepté par un pare-feu, susceptible de bloquer les protocoles de diffusion utilisant des ports spécialisés ou inhabituels. De plus le format MPEG-DASH permet l'utilisation de n'importe quelle norme d'encodage.

Les principales étapes du processus de diffusion MPEG-DASH :

1)Encodage et segmentation

Le serveur d'origine divise le fichier vidéo en segments plus petits, d'une durée de quelques secondes. Le serveur utilise un fichier.mpd, une sorte de table des matières pour les segments vidéo. Les segments sont ensuite encodés, c'est-à-dire formatés de manière à ce que plusieurs appareils puissent les interpréter.

2)Diffusion

Lorsque les utilisateurs commencent à regarder le flux, les segments vidéo encodés sont envoyés aux appareils clients via Internet.

3)Décodage et lecture

Lorsque l'appareil d'un utilisateur reçoit les données diffusées, il décode les données et lit la vidéo. Le lecteur vidéo adapte automatiquement la qualité de l'image aux conditions du réseau. Ainsi, si la connexion de l'utilisateur ne présente qu'une très faible bande passante, la vidéo sera lue à un niveau de qualité inférieur, afin de consommer moins de bande.

HLS ou DASH : quelles sont les principales différences ?

Le HLS constitue un autre protocole de diffusion largement utilisé de nos jours. Les protocoles MPEG-DASH et HLS se révèlent similaires à bien des égards. Les deux protocoles fonctionnent en HTTP, utilisent TCP comme protocole de transport, décomposent les vidéos en segments accompagnés d'un fichier d'index et permettent la diffusion à débit adaptatif.

1)Formats d'encodage

Le format MPEG-DASH permet l'utilisation de n'importe quelle norme d'encodage. Le HLS, d'autre part, nécessite l'utilisation des codecs H.264 ou H.265.

2)Prise en charge des appareils

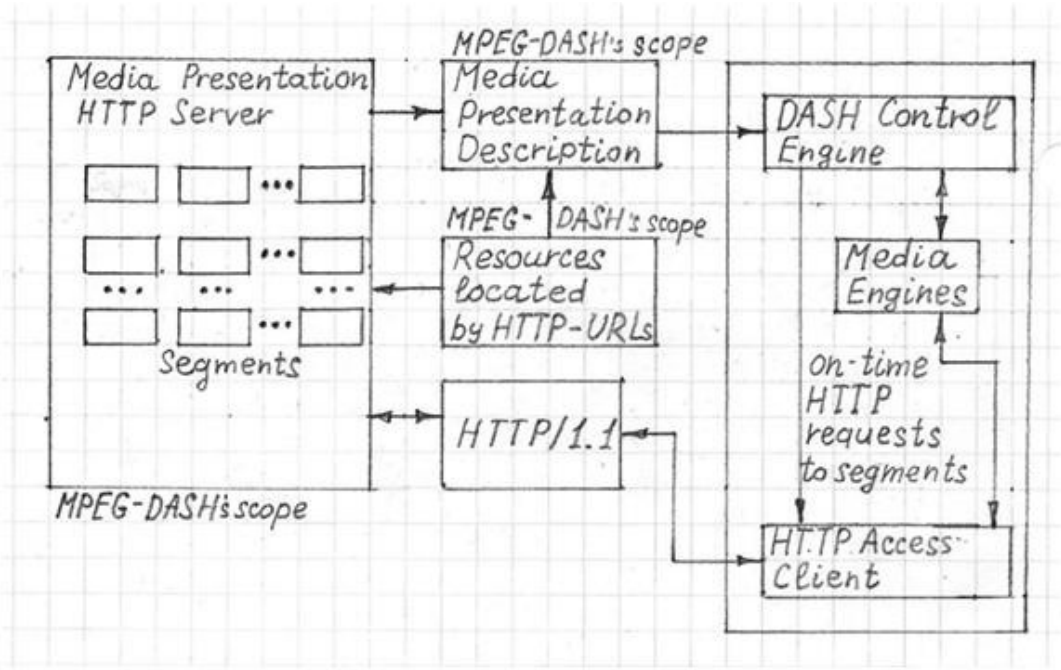
Le HLS était longtemps le seul format pris en charge nativement par les appareils Apple. Ce pendant des acteurs majeurs tels que YouTube et Netflix ont poussé le marché à s'adapter.

3)Longueur du segment

La différence entre les protocoles s'avérait plus marquée avant 2016, car la longueur du segment par défaut s'élevait à 10 secondes pour le HLS. Cette durée s'élève aujourd'hui à 6 secondes pour le HLS. Les segments encodés en MPEG-DASH affichent généralement une longueur comprise entre 2 et 10 secondes, la durée optimale étant de 2 à 4 secondes.

4)Normalisation

Le format MPEG-DASH constitue une norme internationale, qui a été développé par le Moving Picture Experts Group (MPEG), qui est un groupe de travail de l'Organisation internationale de normalisation (ISO). Développé par Apple, le protocole HLS n'a pas été publié en tant que norme internationale, même s'il bénéficie d'une vaste prise en charge.



1 Schéma Interaction Server Client MPEG-DASH

Installation de FFMPEG sur Linux Debian 12

Il y a 2 options :

Première option, la plus simple :
apt install ffmpeg

Deuxième option

Sur le site officiel de ffmpeg : https://ffmpeg.org/download.html

Choisir le source code, le fichier ‘ffmpeg-6.1.tar.xz’

Extraire le fichier

tar -xf ffmpeg-6.1.tar.xz

cd ffmpeg-6.1

./configure

make install

ffmpeg est installé

Pour vérifier son fonctionnement :

Toujours dans le terminal lancé la commande : ffmpeg

Vérifier les caractéristiques d'une vidéo.

ffmpeg -i votrevideo.mp4

-i votrevideo.mp4 : Cela spécifie le fichier d'entrée, "votrevideo.mp4" dans ce cas. C'est le fichier vidéo que nous souhaitons traiter.

Pour raccourcir le temps d’une vidéo

ffmpeg -i votrevideo.mp4 -ss 00:00:00 -to 00:02:10 -acodec copy -vcodec copy votrevideoout.mp4

La durée de la vidéo est exprimée de la sorte hh:mm:ss. Les options « acodec » et « vcodec » avec l’argument copy, indiquent que nous ne touchons pas au codec audio et vidéo.

Pour créer une copie de votre vidéo dans la résolution souhaiter.

-s permet de spécifie la résolution
ffmpeg -i video4K.mp4 -s 1920x1080 videoTHD.mp4

Ici nous avons choisir une vidéo native 4k pour après la convertir dans une résolution inférieure. Nous ne touchons pas au codec vidéo/audio et au bitrate de la vidéo.

Création de copie des vidéos avec différentes résolutions

```
ffmpeg -i video.mp4 -c:v libvpx-vp9 -keyint_min 150 \
-g 150 -tile-columns 4 -frame-parallel 1  -f webm -dash 1 \
-an -vf scale=160:90 -b:v 250k -dash 1 video_160x90_250k.webm \
-an -vf scale=320:180 -b:v 500k -dash 1 video_320x180_500k.webm \
-an -vf scale=640:360 -b:v 750k -dash 1 video_640x360_750k.webm \
-an -vf scale=640:360 -b:v 1000k -dash 1 video_640x360_1000k.webm \
-an -vf scale=1280:720 -b:v 1500k -dash 1 video_1280x720_1500k.webm \
-an -vf scale=2560:1440 -b:v 3000k -dash 1 video_2560x1440_3000k.webm \
-an -vf scale=3840:2160 -b:v 8002k -dash 1 video_3840x2160_8002k.webm
```

-c:v libvpx-vp9 : Spécifie le codec vidéo à utiliser pour la compression vidéo. Dans ce cas, c'est le codec VP9.
-keyint_min 150 : Définit le nombre minimum d'images entre deux images clés (I-frames). Une image clé est une image complète et indépendante qui peut être décodée sans aucune référence aux images précédentes.
-g 150 : Définit la fréquence d'images de référence. Cela spécifie le nombre d'images entre les images clés successives
-tile-columns 4 : Divise la sortie vidéo en colonnes de tuiles. Cela peut améliorer les performances lors de la lecture sur des appareils compatibles.
-frame-parallel 1 : Active le traitement parallèle des trames pour améliorer les performances de l'encodage.
-f webm : Spécifie le format de sortie, qui est WebM dans ce cas.
-dash 1 : Active DASH (Dynamic Adaptive Streaming over HTTP) pour permettre le streaming adaptatif.
-an : Désactive la piste audio dans la sortie.
-vf scale=160 :90 : Redimensionne la vidéo à une résolution de 160x90 pixels.
-b: v 250k : Définit le débit binaire vidéo cible à 250 kbps.
-video_160x90_250k.webm: Nom du fichier de sortie.

En résumé, cette commande prend une vidéo en entrée, la compresse en utilisant le codec vidéo VP9, redimensionne la vidéo, désactive la piste audio, et crée un fichier WebM avec la possibilité de streaming adaptatif en utilisant le format DASH. Nous avons choisi le format WebM, car WebM est un format de fichier multimédia ouvert, principalement destiné à un usage sur le web. Il regroupe des flux vidéo codés en VP8/VP9 et des flux audios codés en Vorbis. Ce format fait partie des formats vidéo proposés pour la balise <video> de HTML5.

Création d'un fichier audio extrait de la vidéo

```
ffmpeg -i video.mp4 -vn -acodec libvorbis -ab 128k -dash 1 my_audio.webm
```

-vn : Désactive la vidéo, indiquant à FFmpeg de ne pas traiter la vidéo.
-acodec libvorbis : Spécifie le codec audio à utiliser, dans ce cas, c'est le codec Vorbis pour l'audio.
-ab 128k : Définit le débit binaire audio cible à 128 kbps.
-dash 1 : Active la création d'un fichier DASH pour le streaming adaptatif. Cela peut être utile pour le streaming audio adaptatif.
-my_audio.webm : Nom du fichier de sortie pour l'audio WebM.

Création du fichier manifest.mpd

On sélection les différents segments de vidéos créer plutôt avec le fichier audio.

```
ffmpeg \
 -f webm_dash_manifest -i video_160x90_250k.webm \
 -f webm_dash_manifest -i video_320x180_500k.webm \
 -f webm_dash_manifest -i video_640x360_750k.webm \
 -f webm_dash_manifest -i video_1280x720_1500k.webm \
 -f webm_dash_manifest -i video_2560x1440_3000k.webm \
 -f webm_dash_manifest -i video_3840x2160_8002k.webm \
 -f webm_dash_manifest -i my_audio.webm \
 -c copy \
 -map 0 -map 1 -map 2 -map 3 -map 4 -map 5 -map 6  \
 -f webm_dash_manifest \
 -adaptation_sets "id=0,streams=0,1,2,3,4,5 id=1,streams=6" \
 my_video_manifest.mpd
```

Les entrées vidéo :
-f webm_dash_manifest -i video_160x90_250k.webm: Fichier manifeste DASH pour la vidéo en résolution 160x90 avec un débit binaire de 250 kbps.

L'entrée audio :
-f webm_dash_manifest -i my_audio.webm: Fichier manifeste DASH pour la piste audio.
Options générales :
-c copy : Copie les flux de chaque fichier d'entrée sans recompressions. Cela est possible car tous les fichiers d'entrée sont déjà au format WebM.
Spécification des flux de sortie :
-map 0 -map 1 -map 2 -map 3 -map 4 -map 5 -map 6: Indique quels flux doivent être inclus dans la sortie. Les indices (0, 1, 2, etc.) correspondent aux indices des fichiers d'entrée spécifiés précédemment.
-f webm_dash_manifest : Spécifie le format du fichier de sortie, qui est à nouveau un fichier manifeste DASH.

-adaptation_sets "id=0,streams=0,1,2,3,4,5 id=1,streams=6": Configure les ensembles d'adaptation dans le fichier manifeste. L'ensemble d'adaptation 0 inclut les flux vidéo (indices 0, 1, 2, 3, 4, 5), et l'ensemble d'adaptation 1 inclut le flux audio (indice 6).
-my_video_manifest.mpd : Nom du fichier manifeste de sortie.

En résumé, cette commande combine plusieurs fichiers WebM de différentes résolutions vidéo et d'une piste audio pour créer un fichier manifeste DASH qui permet un streaming adaptatif en fonction des capacités du lecteur et du réseau.

Intégration de la vidéo dans l'index.html

```
<script src="https://cdn.dashjs.org/latest/dash.all.min.js"></script>
<video id="videoPlayer" controls></video>
<script>

        const videoPlayer = dashjs.MediaPlayer().create();
        videoPlayer.initialize(document.querySelector("#videoPlayer"), "/my_video_manifest.mpd" ,
        true);

</script>
```

Intégration de l'API Dash.js dans une page HTML pour permettre la lecture du flux vidéo DASH spécifié dans le fichier manifeste "my_video_manifest.mpd".

<script src="https://cdn.dashjs.org/latest/dash.all.min.js"></script> :
Inclut le fichier JavaScript de Dash.js depuis un CDN. Dash.js est une bibliothèque JavaScript qui implémente le lecteur DASH dans le navigateur.

<video id="videoPlayer" controls></video> : Crée une balise vidéo avec l'ID "videoPlayer" et l'attribut "controls", ce qui permet d'afficher les contrôles de lecture standard (boutons play, pause, etc.).

<script> : Débute la section du code JavaScript.

const videoPlayer = dashjs.MediaPlayer().create(); :
Crée une instance du lecteur Dash.js et l'assigne à la variable videoPlayer.

videoPlayer.initialize(document.querySelector("#videoPlayer"), "/ my_video_manifest.mpd" , true); :
Initialise le lecteur Dash.js avec les paramètres suivants :
document.querySelector("#videoPlayer") : Spécifie l'élément HTML (balise vidéo avec l'ID "videoPlayer") dans lequel le lecteur vidéo doit être intégré.

"/BBB4K/Big Buck Bunny 4K Demo/my_video_manifest.mpd" : l'URL du fichier manifeste DASH qui décrit la structure du flux vidéo.

true : Active le mode de lecture automatique du lecteur, ce qui signifie que la lecture commencera dès que le lecteur sera prêt.

En résumé, ce code utilise Dash.js pour créer un lecteur vidéo DASH dans une page HTML. Il crée balise vidéo avec des contrôles de lecture, puis initialise le lecteur Dash.js avec l'élément vidéo et l'URL du fichier my_video_manifeste.mpd. Une fois que cette page est chargée, le lecteur Dash.js gère la lecture du flux vidéo DASH de manière adaptative en fonction des capacités du lecteur et du réseau.

Contrôle du flux Réseaux

Pour contrôler le flux réseaux, nous utilisons wondershaper.
Wondershaper permet de contrôler la bande passante réseau entrante et sortante pour des interfaces spécifiques.

Installation :
git clone https://github.com/magnific0/wondershaper.git

Dans l'exemple suivant, notre interface « Ethernet 0 » est limitée à un téléchargement de 4 Mbps et en envoie de 8 Mbps.
./wondershaper -a eth0 -u 4096 -d 8192

./wondershaper -s -a eth0 (-s Afficher l'état actuel de l'interface eth0)

./wondershaper -c -a eth0 (-c supprimer les règles mise en place sur l'interface eth0)

Nous contrôlons notre bande passante réseau pour tester le flux adaptatif, en réduisant notre bande passante, nous constatons le changement de résolution.

Conclusion

Pour conclure, l'adoption du protocole MPEG-DASH pour la diffusion de vidéos sur un site web représente une approche moderne et efficace, offrant une expérience utilisateur optimale. Grâce à son streaming adaptatif, MPEG-DASH s'adapte dynamiquement aux conditions du réseau, garantissant une qualité de visualisation fluide. Sa compatibilité multi-appareils, sa gestion simplifiée du côté serveur avec des fichiers manifestes, et son économie de bande passante font de MPEG-DASH un choix optimal pour les fournisseurs de contenu cherchant à fournir des vidéos de haute qualité de manière évolutive et efficiente sur une diversité d'appareils connectés.