Servidor HLS

En este respositorio está contenido el servidor para el Trabajo 2 de la asignatura Redes y Sistemas de Nueva Generación, de la Universidad Pública de Navarra.

Ha sido desarrollado por el Grupo 5, compuesto por Idoia Cerro, Xabier Dendarieta y Sonia Elizondo. Puede verse el vídeo explicativo en Youtube.

El servidor es una modificación del repositorio de T. Mullen.

Requisitos previos

- Máquina con Linux.
- Paquetes `npm` y `nodejs-legacy` instalados.

Inicialización

En primer lugar, habrá que instalar las dependencias de `npm`:

- `npm install`

Con esto, ya estaría todo listo para ejecutarse. Aunque falta obtener varios archivos de stream adicionales.

Crear un archivo de streaming

Se necesita previamente tener instalado `ffmpeg`, que es una librería de tratamiento de audio y vídeo. Generalmente se utiliza para conversión de formatos, como haremos en nuestro caso, aunque incluye más funcionalidades.

Es importante que, dentro de los parámetros que vamos a utilizar para convertir los vídeos a streams, tengamos siempre `pix_ftm yuv420p`, ya que el protocolo HLS parece sensible a este tipo de codificación de píxel.

Preprocesado de vídeo con anterioridad

Para convertir un **archivo MP4** (parámetros entre corchetes opcionales):

- `ffmpeg -i archivo.mp4 [-vf scale=640:480] -f hls -c:v h264 -profile:v baseline -pix_fmt yuv420p -hls_time 5 -hls_list_size 0 stream.m3u8`

El parámetro `hls_time` controla el tiempo medio de duración de los segmentos temporales que se vayan generando. Es un parámetro importante a tener en cuenta para el rendimiento del servidor: cuanto más grande, más tarda en servir; cuanto más pequeño, más veces tiene que servir.

Preprocesado de vídeo en directo

Para grabar directamente desde la **cámara web**:

- Primero, listar los dispositivos con: `ffmpeg -list_devices true -f dshow -i dummy`
- Luego, para comenzar, utilizar el comando: `ffmpeg -f dshow -i video= "<dispositivo de la lista>" -vf scale=640:480 -f hls -c:v h264 -profile:v baseline -pix_fmt yuv420p -hls_time 1 stream.m3u8`

En nuestro caso en lugar de utilizar una webcam, hemos optado por grabar el escritorio de la máquina virtual, lo que nos pareció una opción más viable que el conseguir conectar una cámara a la propia máquina virtual.

- Para grabar la pantalla, habrá que adaptar el parámetro `video_size` por la resolución de la máquina donde se ejecute: `ffmpeg -video_size 1440x900 -f x11grab -i :0.0 -f hls -pix_fmt yuv420p -vf scale=900:500 stream.m3u8`

Archivos provistos

En el repositorio se ha provisto un script en bash, llamado `getStreams.sh`, que descarga los streams extra. Si no se ejecutan, sólo se tiene el de test. Para ejecutar se necesitan `wget` y `unzip` previamente.

También se ha provisto de un vídeo para falso directo, obtenible con el script `getFakeLive.sh`, con la intención de generar el stream conforme se van sirviendo los segmentos temporales. Este archivo habrá que ir transformándolo en stream en directo mediante `ffmpeg`.

Con objetivo de hacer el procesado en directo, tanto de la grabación de pantalla como del archivo en falso directo, se ha provisto de los scripts `startLiveStream.sh` y `startFakeLiveStream.sh`. Estos archivos simplemente generan la estructura de carpetas necesaria y lanzan el comando de `ffmpeg` necesario en cada caso.

- El comando para **directo real** (grabación de pantalla) es: `ffmpeg -video_size 1440x900 -f x11grab -i :0.0 -f hls -pix_fmt yuv420p -vf scale=900:500 "src/output/more/live/stream.m3u8"`
- El comando para **falso directo** (archivo MP4) es: `ffmpeg -i "src/output/more/fake_live.mp4" -vf scale=640:480 -f hls -c:v h264 -profile:v baseline -pix_fmt yuv420p -hls_time 1 -strict experimental "src/output/more/fake/stream.m3u8"`

NOTA: en caso de que los scripts de descarga fallen o no descarguen el contenido debido, acceder a las siguientes webs desde un navegador:

- *Para los streams small y large: https://gofile.io/d/siShhl*
- *Para el video fake_live.mp4: https://gofile.io/d/L5i3VT*

Tras ese acceso, dichos scripts deberían funcionar correctamente, siempre que los archivos sigan en línea.

Poner en marcha el server

- Acceder hasta la carpeta `src`
- Utilizar el comando: `node app.js`
- Acceder a: http://localhost:8000/ o http://localhost:8000/player.html y hacer click en el stream deseado (Funciona en Opera, Firefox y Chrome sin necesidad de extensiones)
    - En caso de acceso externo al servidor, puede utilizarse la IPv6 configurada con la VPN de la asignatura: `http://[2001:720:1d10:fffe::1000:5]:8000/`

Pruebas realizadas

Para probar el protocolo HSL y su comportamiento hemos decidido probar varias situaciones: en primer lugar, con un único cliente accediendo y, en segundo lugar, con dos clientes.

Pruebas con un único cliente conectándose

- Tres accesos al **vídeo test**.
- Tres accesos al **vídeo small**.
- Tres accesos al **vídeo large**, realizando varios saltos para obligar a cargar zonas nuevas del vídeo en desorden.
- Un acceso al **stream fake live**, realizando saltos en la zona visible del reproductor.
- Un acceso al **stream live**, realizando saltos en la zona visible del reproductor.

Las pruebas con un único cliente han sido realizadas en **Firefox (x32)** y en **Chromium (x32)**, lo más parecidas posibles, con el fin de verificar si hay diferencias entre las implementaciones del protocolo HLS entre ambos navegadores.

Pruebas con dos clientes simultáneos

- Tres accesos al **vídeo test**.
- Tres accesos al **vídeo small**.
- Tres accesos al **vídeo large**, realizando varios saltos para obligar a cargar zonas nuevas del vídeo en desorden.
- Un acceso al **stream fake live**, realizando saltos en la zona visible del reproductor.
- Un acceso al **stream live**, realizando saltos en la zona visible del reproductor.

En el caso de las pruebas para dos clientes simultáneos, solo se han realizado en **Firefox (x32)**, ya que en este caso se buscaba ver el comportamiento del servidor ante esta situación. También se ha procurado que haya un pequeño margen entre la petición del primer cliente y la del segundo, para poder resaltar más la posible diferencia.

Registro de accesos

Con intención de facilitar el análisis de los resultados y su representación, el servidor está preparado para resgitrar en *logs* la información que se va generando sobre las peticiones en un archivo llamado `log.csv`.