Cristian Fernandez Moreno NIA: 229611

Sistemes de codificació de video - Practice 3 Report/Readme

- To run the exercises, I used an Uncharted 4 video that can be downloaded here.

Task 2)

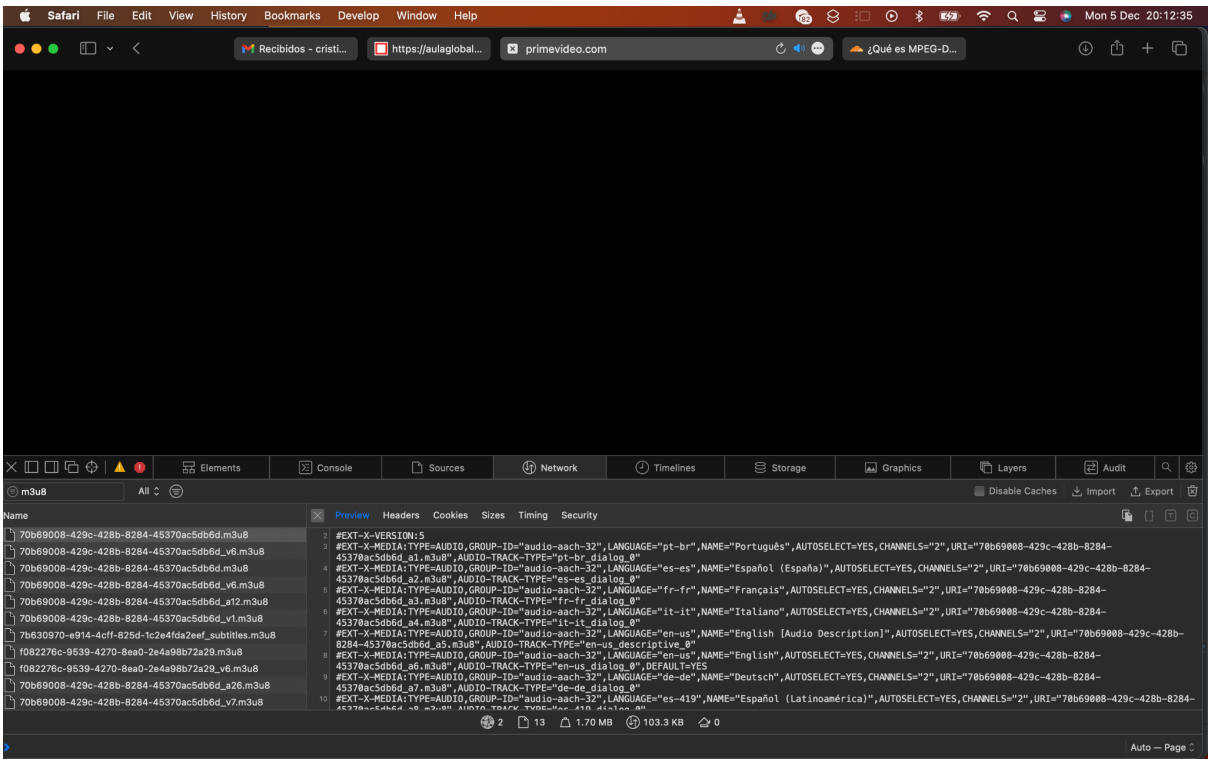
Some considerations: first, to run this task, bento4 tools should be installed, since in the .py I directly run commands through os.system() using bento4 tools such as *mp4fragment* and *mp4dash*. Secondly, I gave the option to encrypt the MPEG DASH file, but of course you cannot directly play an encrypted file, it should be unencrypted using *mp4decrypt*. In order to make a working demonstration, I created an unencrypted file and played it using a player that allows me to play .mpd files.
I made a simple HTTP server using Python to serve the files to Chrome and used this extension which is a native MPEG DASH player to be able to play them in the browser. Here you can see a demonstration of me doing this process.

Task 3)

In this task I used the RTMP (Real Time Messaging Protocol) to livestream using FFmpeg.
I used this amazing program I got from Github to create a simple minimalistic RTMP local server on my Mac, and thus I could stream content in real time using a single short FFmpeg command with the RTMP address of the server. In order to be able to stream the content from other devices in my local network I just put the local IP address from my computer in the RTMP part of the command and just like that I was able to stream the video from my phone.
Here you can see a demonstration of the process from the point of view of my computer, and here you can see a screen recording from my phone where I stream the video from the VLC app.

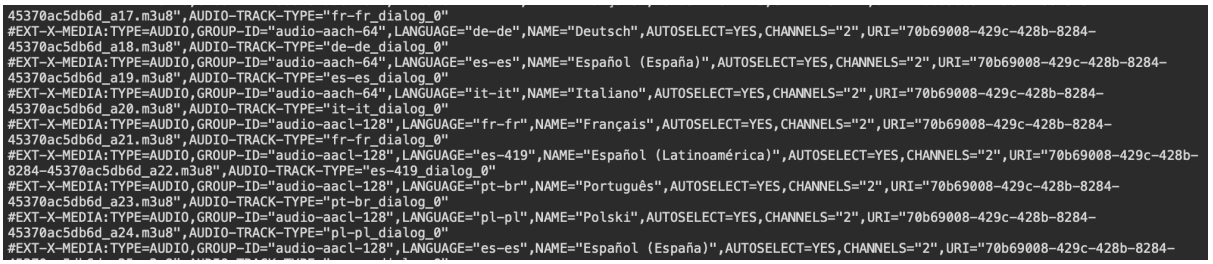
Task 4)

Since I'm subscribed to several streaming services I've tried to see the information of a couple, Amazon Prime Video and Disney+.
First, with Amazon Prime Video on Safari, the video is clearly HLS, since we can see .m3u8 files that are downloaded when the video starts.

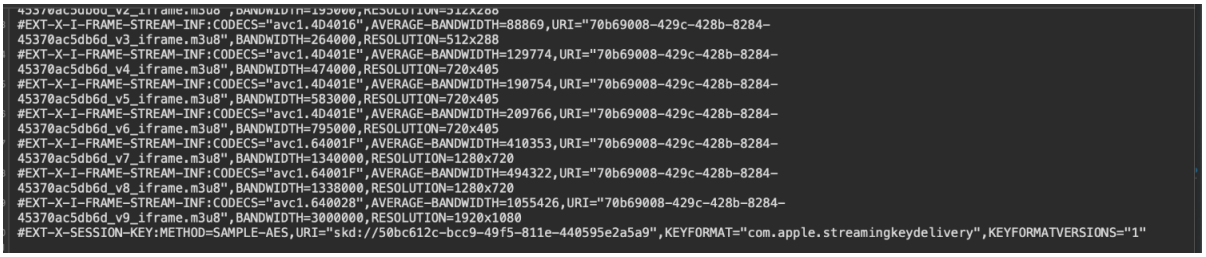


If we zoom in the information provided by the files, we can see the information for the different tracks provided.

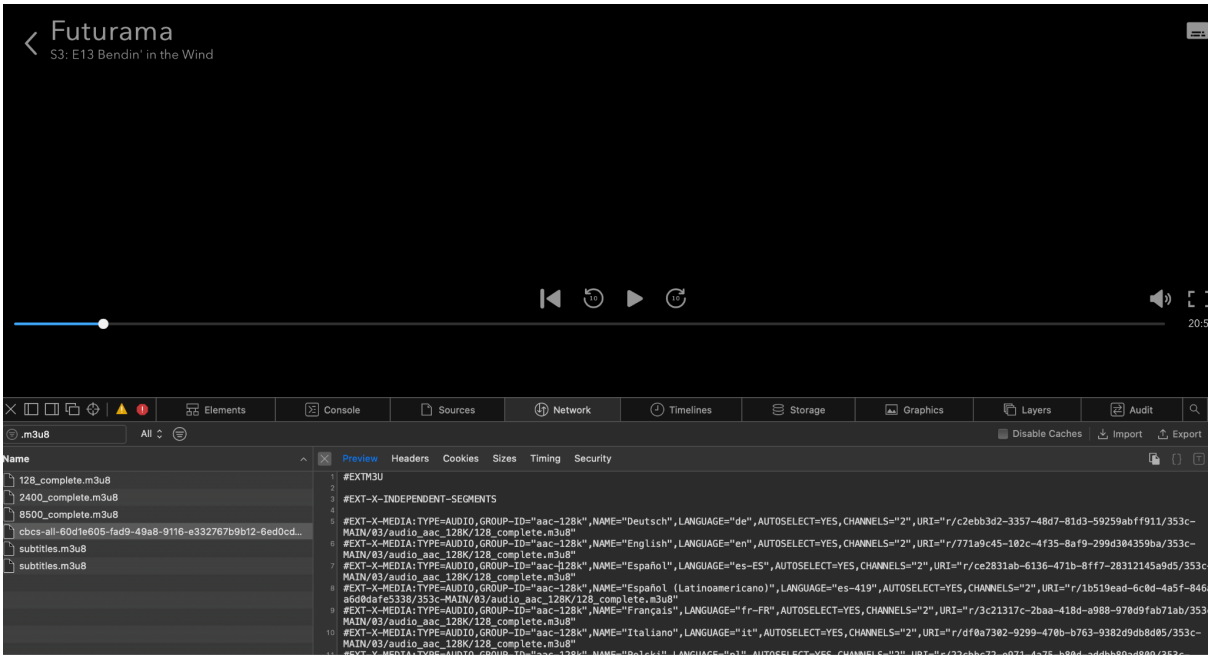
First for the audio, there is one track for each language provided, with the audio encoded in ’aach’, which corresponds to MPEG-4 High Efficiency AAC audio. The ‘32’ that accompanies the AAC-H bit I'm guessing refers to the bitrate, 32kbps.
There’s also some higher quality audio, encoded in ‘aacl’ with 128kbps, which corresponds to MPEG-4 AAC Low Delay audio, another variation of AAC:



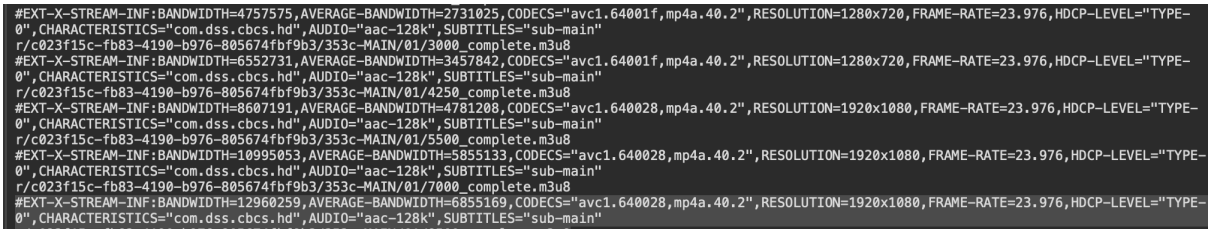
Scrolling down we can see some subtitle tracks and also the tracks for video, each with a different average bandwidth and resolution, in order to make the streaming adaptive. We can also see that they are encoded using AVC (H264).



Now looking at Disney Plus also in Safari, we can also see some .m3u8 files, which indicates HLS streaming. The audio is also AAC, with various tracks for various languages and bitrates.



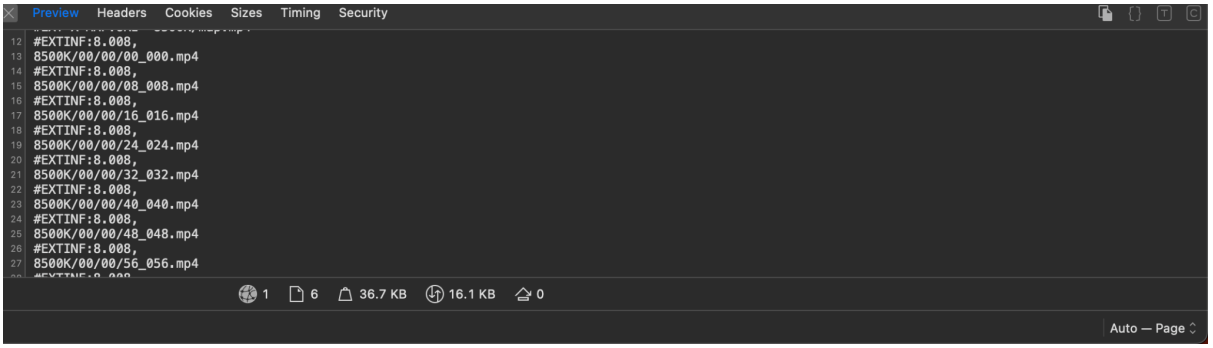
The video files are again encoded in H264:



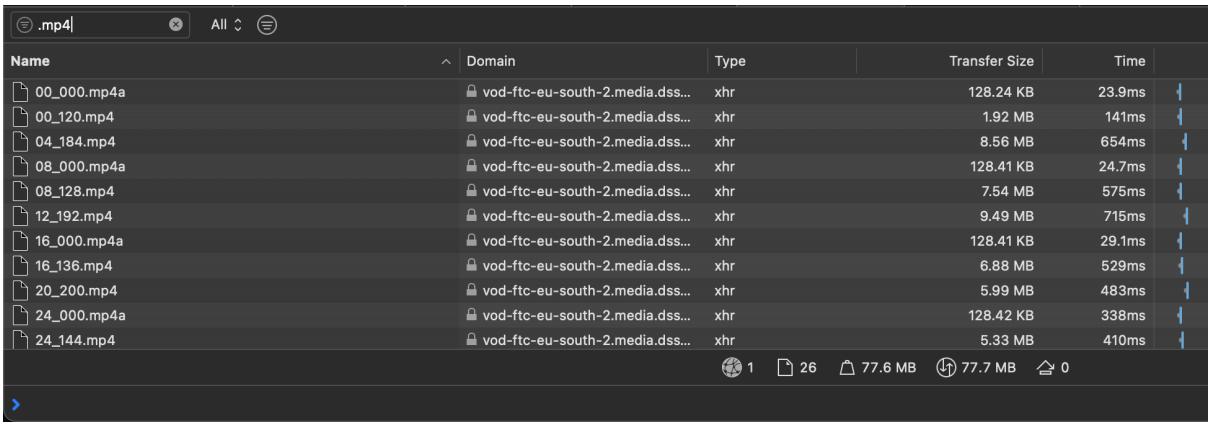
This is surprising to me since I expected to see a lot more HEVC.
We can even see at the bottom some keys likely related to DRM such as Microsoft Play Ready.

Furthermore, we can also see which tracks it decides to load, based on my network speed.

‘8500_complete.m3u8’ is the one with the highest bitrate and resolution. If we open it we can see the information playlist for the different .mp4 fragments, we can see that each fragment is 8.008 seconds long, and they’re organized in folders by hour, minute, and the final fragment has the seconds and milliseconds in the name.



These fragments are requested through HTTP requests as the video advances :



Through HLS the service would be able to switch seamlessly through different qualities if the network speed changed.