Seminar 3

Sebastian R. Ovelar Anderson **NIA:206384**

1) **Remember you have the BBB video converted into these?**

720p
480p
360x240
160x120

·Convert them into VP8, VP9, h265 & AV1. You can use the script that allows you to transform, or create a new script.

I did this using the following commands, for example for the file of size 160x120:

```
ffmpeg -i BBB_160X120.avi -c:v libvpx -b:v 1M -c:a libvorbis BBB_160X120_vp8.webm"

“ffmpeg -i BBB_160X120.avi -c:v libvpx-vp9 -b:v 2M BBB_160X120_vp9.webm"

"ffmpeg -i BBB_160X120.avi -c:v libx265 -crf 26 -preset fast -c:a aac -b:a 128k BBB_160X120_h265.mp4"

"ffmpeg -i BBB_160X120.avi -c:v libaom-av1 -crf 30 -b:v 0 -strict experimental BBB_160X120_av1.mkv"
```

Where with the option -c:v we specify the codec of the video -c:a the codec of the audio and -b:a is the bitrate. For libvpx(vp8) we have a variable bit rate that will try to reach 1Mbit/s on average, for vp9 this bit rate to reach is 2Mbits/s. -crf is the constant quality mode the lower the number the more bits we have for the file and thus we have a bigger bitrate at the expense of file size. I save all the commands to run to the terminal in E1.py. Now we have all the videos asked.

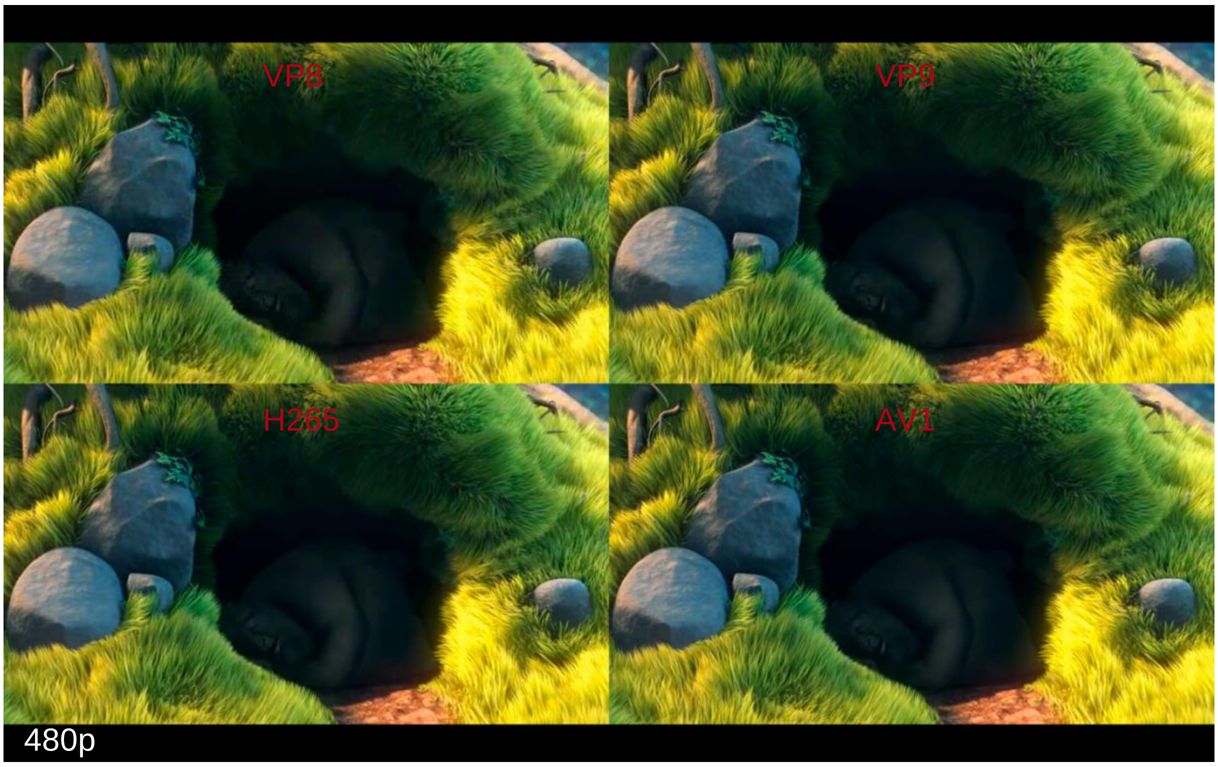
2) **Once you have the ‘mandanga’, try to create a new video as the one of the 4 videos at the same time we saw in class, and please analyze by yourself and comment how these codecs work at each bitrate.**

I’ve tried to create a python script for this exercise but there is an error when creating the files they don’t seem to work when I tried to open them. So, I end up doing it manually in the terminal with ffmpeg. The command for each file with different qualities was:

```
ffmpeg -i BBB_160X120_vp8.webm -i BBB_160X120_vp9.webm -i BBB_160X120_h265.mp4 -i
BBB_160X120_av1.mkv -filter_complex
"[0:v][1:v]hstack[top]; [2:v][3:v]hstack[bottom];
[top][bottom]vstack,format=yuv420p[v];
[0:a][1:a][2:a][3:a]amerge=inputs=4[a]"
-map "[v]" -map "[a]" -ac 2 output_160X120.mp4
```

This basically creates the files and uses a filter to split the screen in four and stack the video inputs first horizontally, the vp8 next to the vp9 at the top, and the h265 and av1 at the bottom and then stack the top and bottom vertically in this way we have the videos at the same time, and we can compare them.

In the next figure we can clearly see that the worst quality video is the one with vp8 codec this is not surprising as this is the less effective codec VP9 was develop after VP8 and it is a better version. We can see that at the same bitrate VP9 has much better quality than VP8, we can also say the same for h265 and AV1 comparing it to VP8.



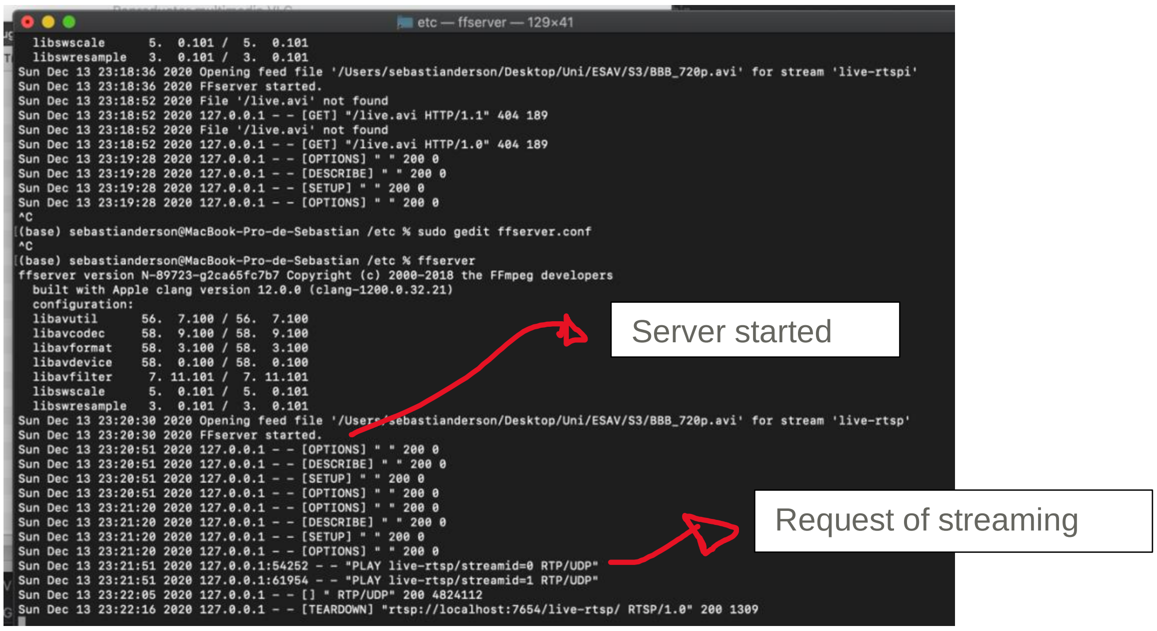
Now comparing the 3 frames that look with better quality, these are VP9, H265 and AV1 we cannot see a substantial difference in quality maybe the AV1 seem smoother and with less distortion if you look in the skin of the bunny. But there are some other aspects to consider. For example, the time it took to code. In my case it was just a 10 seconds video but specially for AV1 it took many minutes to finish coding the videos. The codec that consume the most the bit rate is VP9 following H.265 and then AV1.

3) **Try to achieve with FFMpeg or FFServer to create a live streaming of the BBB Video. You should broadcast it into an IP address (locally of course) and open this IP or URL inside VLC Media Player.**

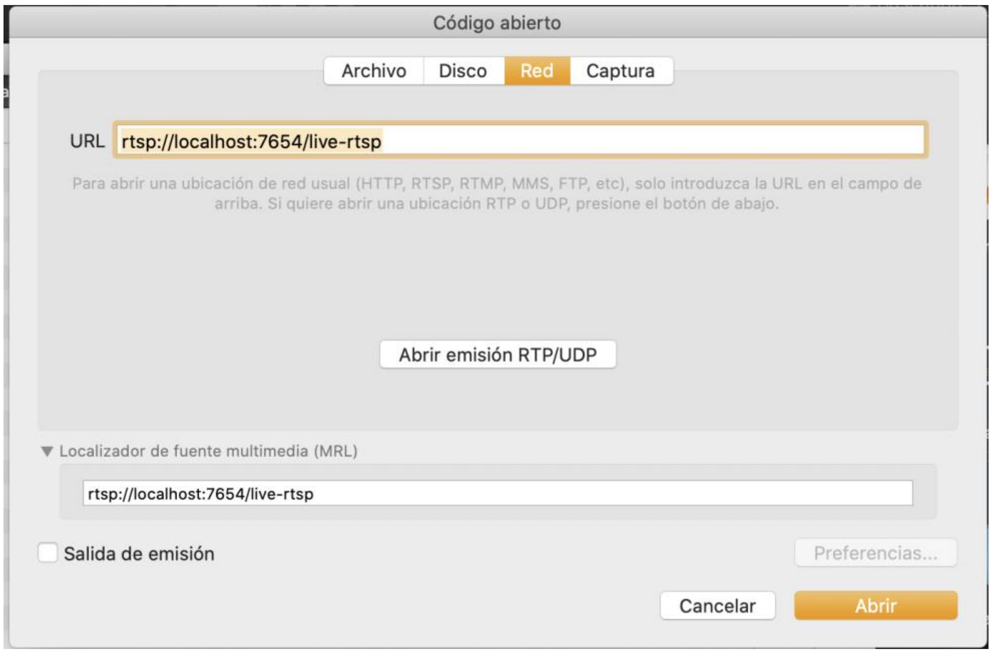
For this exercise I've used FFServer. As the functionality is not included anymore in the new version of ffmpeg I need to commit the ffmpeg to the commit 2ca65fc7b74444edd51d5803a2c1e05a801a6023 which is the last commit that included this feature for this I've used the following commands.

```
git clone https://git.ffmpeg.org/ffmpeg.git ffmpeg
cd ffmpeg
git checkout 2ca65fc7b74444edd51d5803a2c1e05a801a6023
./configure
make -j4
```

Then I needed to configure the server file ffserver.conf with some commands that specified the port and the protocol to follow and the filepath of the file that will be streamed. I did this with the help of ffserver guide on ffmpeg.org[1]. The protocol I choose is RTSP which stands for real time streaming protocol. Once I have modified correctly the ffserver.conf and it worked we used the command ffserver to start the server. We can see it worked in the following images and the server has started functioning.



```
  libswscale      5.  0.101 /  5.  0.101
  libswresample   3.  0.101 /  3.  0.101
Sun Dec 13 23:18:36 2020 Opening feed file '/Users/sebastianderson/Desktop/Uni/ESAV/S3/BBB_720p.avi' for stream 'live-rtspi'
Sun Dec 13 23:18:36 2020 FFserver started.
Sun Dec 13 23:18:52 2020 File '/live.avi' not found
Sun Dec 13 23:18:52 2020 127.0.0.1 - - [GET] "/live.avi HTTP/1.1" 404 189
Sun Dec 13 23:18:52 2020 File '/live.avi' not found
Sun Dec 13 23:18:52 2020 127.0.0.1 - - [GET] "/live.avi HTTP/1.0" 404 189
Sun Dec 13 23:19:28 2020 127.0.0.1 - - [OPTIONS] " " 200 0
Sun Dec 13 23:19:28 2020 127.0.0.1 - - [DESCRIBE] " " 200 0
Sun Dec 13 23:19:28 2020 127.0.0.1 - - [SETUP] " " 200 0
Sun Dec 13 23:19:28 2020 127.0.0.1 - - [OPTIONS] " " 200 0
^C
(base) sebastianderson@MacBook-Pro-de-Sebastian /etc % sudo gedit ffserver.conf
^C
(base) sebastianderson@MacBook-Pro-de-Sebastian /etc % ffserver
ffserver version N-89723-g2ca65fc7b7 Copyright (c) 2000-2018 the FFmpeg developers
  built with Apple clang version 12.0.0 (clang-1200.0.32.21)
  configuration:
  libavutil      56.  7.100 / 56.  7.100
  libavcodec     58.  9.100 / 58.  9.100
  libavformat    58.  3.100 / 58.  3.100
  libavdevice    58.  0.100 / 58.  0.100
  libavfilter     7. 11.101 /  7. 11.101
  libswscale      5.  0.101 /  5.  0.101
  libswresample   3.  0.101 /  3.  0.101
Sun Dec 13 23:20:30 2020 Opening feed file '/Users/sebastianderson/Desktop/Uni/ESAV/S3/BBB_720p.avi' for stream 'live-rtsp'
Sun Dec 13 23:20:30 2020 FFserver started.
Sun Dec 13 23:20:51 2020 127.0.0.1 - - [OPTIONS] " " 200 0
Sun Dec 13 23:20:51 2020 127.0.0.1 - - [DESCRIBE] " " 200 0
Sun Dec 13 23:20:51 2020 127.0.0.1 - - [SETUP] " " 200 0
Sun Dec 13 23:20:51 2020 127.0.0.1 - - [OPTIONS] " " 200 0
Sun Dec 13 23:21:20 2020 127.0.0.1 - - [OPTIONS] " " 200 0
Sun Dec 13 23:21:20 2020 127.0.0.1 - - [DESCRIBE] " " 200 0
Sun Dec 13 23:21:20 2020 127.0.0.1 - - [SETUP] " " 200 0
Sun Dec 13 23:21:20 2020 127.0.0.1 - - [OPTIONS] " " 200 0
Sun Dec 13 23:21:51 2020 127.0.0.1:54252 - - "PLAY live-rtsp/streamid=0 RTP/UDP"
Sun Dec 13 23:21:51 2020 127.0.0.1:61954 - - "PLAY live-rtsp/streamid=1 RTP/UDP"
Sun Dec 13 23:22:05 2020 127.0.0.1 - - [] " RTP/UDP" 200 4824112
Sun Dec 13 23:22:16 2020 127.0.0.1 - - [TEARDOWN] "rtsp://localhost:7654/live-rtsp/ RTSP/1.0" 200 1309
```



URL: rtsp://localhost:7654/live-rtsp

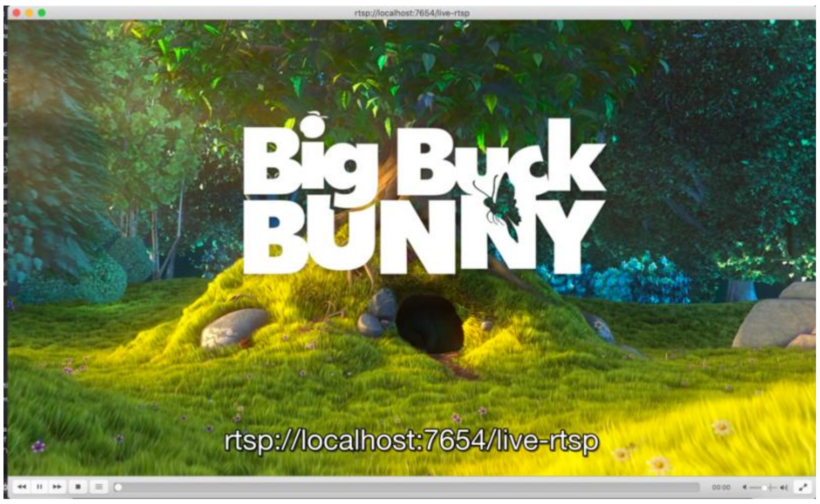
Para abrir una ubicación de red usual (HTTP, RTSP, RTMP, MMS, FTP, etc), solo introduzca la URL en el campo de arriba. Si quiere abrir una ubicación RTP o UDP, presione el botón de abajo.

▼ Localizador de fuente multimedia (MRL)

rtsp://localhost:7654/live-rtsp

☐ Salida de emisión

This image shows the URL we introduced in VLC to request the start of the streaming.


rtsp://localhost:7654/live-rtsp
Big Buck BUNNY
rtsp://localhost:7654/live-rtsp

And finally, we see the streaming and the URL at the bottom.

4) Try to generate a python script that enables/activates the online streaming.

For this exercise I did a python file to copy the ffserver.conf into the etc/ folder and start the server on file E4.py. Once you have this done you can start VLC and introduce the link that the command tells you and see the streaming on VLC. Everything is also in my repository on github: *https://github.com/sebastianderson/S3.git.*