Video Streaming Mixer Library, Workshop 1 Transcript
Poonam Roy, Mohamed Mesto, Yuni Quintero

Introduction: Mohamed

Welcome to the Advanced Web Technologies Project. Our Topic handles the Video Streaming Mixer Library.

Agenda: Mohamed

We will present to you the following contents about Video Streaming Mixer Library, Problem Statement, Paper and Technology Review, Possible Solutions, Schedule and Next Steps.

Problem statement: Yuni

We are definitely in the streaming era. Live shows, movies, TV programs, music and any other multimedia content is being delivered and consumed continuously from a source over the internet. The user can start playing the content without having to wait for it to be downloaded or totally transmitted. Adaptive bitrate streaming standards, like HLS, are being used nowadays to enhance the user experience while consuming video streams over the internet, changing and adapting dynamically the resolution of the video source to avoid buffering and bandwidth bottlenecks.

HLS provides different versions or copies of a video stream with lower, medium and higher quality of resolution and bitrate. HLS splits a video source into little segments for each available resolution. Each resolution is a playlist itself containing all those split videos of their respective resolution and bitrate. A user, for example, might want to enjoy multiple streams as one single playlist to avoid pausing each TV episode.

In order to combine these streams into a main one, they have to be compatible with each other in terms of video resolution and bitrate. A video stream can have multiple resolutions available thanks to HLS, but not all streams share the same representations. Two video streams might not be compatible with each other because they may have different source resolutions and bitrates from the HLS initial formatting.

We seek to resolve this issue by creating a playlist consisting of multiple video streams that already exist, making sure that the different video stream sources are all compatible in regards of resolution and bitrate. We need to identify the multiple representations available from each video stream that is intended to be in the playlist. Then, having this data, an algorithm is needed to handle and filter out the video streams that are not compatible in order to output a single playlist with the final matching representation that includes multiple video stream sources.

Paper and technology review: Poonam

We all are using progressive video streams everyday, after the covid pandemic, the use of streaming video has increased over the internet.

There are two major problems with progressive streaming.

The first is quality
In this diagram it shows the journey of a progressive video from Server to User.
Here the video file size is 1280 x 720 pixels, and that same file will simply be stretched to fit the various screen sizes that it plays on.
This format of video file will never play at correct quality levels on a screen that is 1920 x 1080px. It will be stretched and as a result we see pixelation or blurry images.
The second is buffering. If the user has a low-quality internet connection,
and cannot download the video stream fast enough to keep the video playing. This is very common, especially on mobile devices, where the connection can vary depending on the user's location.

This is why video quality may get better or worse in the middle of a video as a user is watching it. This feature is known as "adaptive bitrate streaming," and without it, slow network conditions can stop a video from playing altogether.

More the length we require more bandwidth or resolution and time for downloading is required,
Irrespective of the length of the video by using Adaptive bitrate streaming will help users to switch back and forth between video qualities.

For this we require some technology such as

HTTP Live Streaming (HLS) is an adaptive streaming communications protocol created by Apple. It uses the m3u8 manifest format. The manifest file contains the references to the available segments and their respective bitrates and any other attribute.
HTTP Live Streaming provides a reliable, cost-effective means of delivering continuous and long-form video over the Internet. It allows a receiver to adapt the bit rate of the media to the current network conditions in order to maintain uninterrupted playback at the best possible quality.

Adding to that

hls-parse js library
That is A Javascript library to parse and edit HLS playlists. it Provides synchronous functions to read/write HLS playlists. The library creates a programmatically editable Object from the manifest.

Node.js is intended to run on a dedicated HTTP server and to employ a single thread with one process at a time. Node.js applications are event-based and run asynchronously.

Code built on the Node platform does not follow the traditional model of receive, process, send, wait, receive. Instead, Node processes incoming requests in a constant event stack and sends small requests one after the other without waiting for responses.

This is a shift away from mainstream models that run larger, more complex processes and run several threads concurrently, with each thread waiting for its appropriate response before moving on.

R. Seeliger, D. Silhavy, Dr. S. Arbanowski “Dynamic ad-insertion and content orchestration workflows through manifest manipulation in HLSand MPEG-DASH” is an example of how the hls manifest can be manipulated to output a different video stream. It is a proof of concept that demonstrates how manifest manipulation enables dynamic ad insertion and flexible over-the-top streaming workflows . Content manipulation and dynamic ad insertion is realized through non-video-intrusive technologies operating on manifest level in a highly flexible and scalable system architecture.

Possible solutions: Yuni/Mohamed

We will parse the HLS manifests from the multiple video streams using hls-parse JS library. Having the respective object representations for further manipulation and handling, we will implement two algorithms that will identify the matching representations based on resolution and bitrate. The first strategy consists of identifying which video streams have matching representations from the first element of the input array. One video stream might have more resolutions than the first input so those would have to be left out.

The second strategy consists of finding an intersection between the variant playlists in regards to their resolution and bitrate. For both strategies, the final output should be the master HLS manifest of a playlist that includes the concatenated segments from all inputs matching the respective representation. Using the hls-parser library we will create and manipulate this final HLS master manifest.
Here we can see the diagram flow that shows how the multiple video streams manifest as input are going to be parsed through the hls-parse library to obtain each object representation. Then, the two strategies will be applied to this set of objects to find the matching and compatible representations in orden to output a final master playlist using, again, the hls-parse library to create/write the master manifest.

Roadmap: Mohamed

In this slide I’d like to highlight our project’s Roadmap:
So today the 17th of May is the 1st workshop taking place: As my group mates Yuni, Poonam and I clarified the three main points: Identify the problem statement, review the Paper and Technology and Identify Possible Solutions.
Until the 14th of Juni we will achieve the following work elements: Implement the first Strategy, Study of HLS Apple and HLS Parser Documentation ,Setup the development environment, the repository, deployment and Parse the HLS manifest into Object representation using the HLS JS Library. Which is referred to as workshop 2.
Moreover, the 3d workshop which will take place on 19th of July include: Output final master HLS manifest with the matching streams, setup as a dependency Library and Implement the second strategy. Nevertheless the project will be finished on the 31th of July by Submitting the final code and the Scientific paper.

Next steps: Mohamed
The next steps (workshop 2) are represented in: Implement the first strategy (First Demo.), Setup the development environment, Setup the repository & the deployment, Study of HLS Apple and HLS Parser Documentation and Review of Use cases and graphical representation.