Intro: Mohamed
Welcome to the Third and final workshop of the Advanced Web Technologies Project. under the Topic Video Streaming Mixer Library.

We will present to you the following contents of our Agenda

Wrap up: Problem Statement and Schedule,Output Manifest,Library Component and Final Demo

/
Wrap up Problem Statement and Schedule: Mohamed

What has been done so far:

We have implemented two strategies that select compatible streams in regards to resolution in order to join these streams into a single master playlist.. The first strategy consists of matching resolutions against the first element of the input array ,and the second strategy consists of an intersection of resolutions. We use as tools the hls-parser js library and node js as a development environment.

Output Manifest: Poonam

After implementing both strategies, we have as a result an array with the compatible streams and an array with the resolutions that matched.

Having the streams resulting from the filtering of the strategies, we have to obtain the matching variants for each resolution of the final output.

With the help of a dictionary with key resolution, we gather all the variants that matched with it according to the strategy. A stream can have multiple variants with the same resolution but with different bandwidths. In order to remove these duplicates, we choose the variant with the maximum bandwidth value.

For each matching resolution, we fetch the media playlist of all the variants to obtain their segments which will be joined in an array to create a new Media Playlist object that will represent the Variant playlist in the master manifest. We convert this new Media Playlist object into string representation in a .m3u8 file identified with the corresponding resolution.

Then, we create a new Variant object that has as uri the filename of the media playlist m3u8 file mentioned before, its resolution and bandwidth attributes, which is the maximum of all the variants joined. This new variant object will be saved in the variants dictionary mentioned before in the corresponding resolution key.

Finally, we create a new MasterPlaylist object joining all the variants in the dictionary in an array to assign the variants attribute to the master playlist. We convert this playlist into string representation too and write it on a master.m3u8 file.

Library Component: Mohamed

We created an npm package that has in the file utils.js two methods: algorithmA() and algorithmB(), where both receive as parameters an array of string URLS, along with other utility functions like joinSegments(), makeRepresentationsDict(), createMasterPlaylist(). These two methods are then exported in the index.js from which an external node app can import as a dependency, installing this package with the following command:

npm i awt-pj-ss22-video-streaming-mixer-library-1.

Final Demo: Yuni

For our demo here we can see our node package which is the video mixer library, in the utils.js file we can find all the methods that were necessary to implement both strategies, algorithmA and algorithmB which both receive as a parameter a string made of url addresses. In the end of this file we indicate which methods are to be exported in the library which are algorithmA and algorithmB.

We also have our own node testing application which will serve us as our external application which imports the video mixer library as a dependency and we find it here in the node_modules folder. In the main.js file this file imports the two strategies, algorithmA and B.

To test the first strategy which consists of having as an output a final manifest which will have the representations present only for the first element of the input array. In this case we made our own HLS file for the testing of this strategy which is hosted in our computer. We have our localhost server running and we're going to test the first strategy and we can see the first urls resolutions, which are the needed resolutions, are 1280x720, it is not present in the second stream but it is in the third one. Our final manifest as an output form the algorithm will have only one variant which is 1280x720 and makes a reference to the media playlist file that was created for this representation. The playlist file has all the segments from the two streams concatenated in a single list. If we want to test this we have to play it on Safari browser and we can see the first stream being played and the second one which starts there. We can see these two streams playing sequentially in a single stream using algorithmA.

Now we want to test the second strategy which consists of having as a final output the intersection of resolutions from all the streams in the input array. If we run the code we can see that the intersection from these four streams is the representation 640x360 which is present in every stream that was given in the input array, and we have as an output files the master manifest and the 640x360 media playlist file. If we want to check them we can see the master manifest has again only one variant which is the only one present in each of the input streams in the array and we can see the media paylist 640x360 which has all the segments or clips from the four streams concatenated in a single playlist. If we want to check them, we can see here the first stream, the second, third and fourth.

Also, if we want tot est our library with more than one representation present in the final output, we are going to first clean these outputs and we are going to test with other streams for the second algorithm. We can see that for these three streams we have four representations resulting from the intersection. The representations 640, 960, 1280 and 1920 which are present in these three streams. We have as output files four representations along with the master manifest. We can see that our master manifest has now four variants, each of them kaing a reference to the media playlist file which has the name of the resolution. We have all the segments or clips concatenated for the three streams. We refresh the player and we can see the first, the second and third which is a short ad. Thank you very much.