Ivan Kouzmine

ECE 50863 – Sring 2020

ABR Algorithms with Dash.js

github.com/ivanscode/abrlab

Introduction:

This is an exploration of available ABR algorithms that are currently out in the world. As the backbone, two formats are used for file encoding, DASH and HLS. We will be exploring DASH for now. The project is meant to help prototype and test new ABR algorithms as well as demonstrate the power of existing ones like the ones based on the Throughput Rule and BOLA.

Implementation:

The current setup is very simple but working. Using python's very simple built-in web-server, a single page is hosted along with the content to be viewed. Behind the scenes, there is a simple script that uses ffmpeg and mp4box to convert an input mp4 video file into the segmented DASH format. These converted files get placed in the server directory for the demo. The server can be started with "python -m http.server <port>". Since GitHub does not really like uploading large files, and a 1080p mp4 file is substantially large, demo files are not included with the project. Instructions on setup and conversion can be found on the GitHub page.

The heart of the demo is in how Dash.js allows for custom ABR algorithms to be implemented. A sample custom algorithm is included in the project as well as its implementation in the demo. The idea is simple, using whatever metrics Dash.js has on hand, select a bitrate for a segment. This is accomplished by applying a Rule class in the settings of the player and can be seen in the main.js file. As of the writing of this report, only one sample is provided, but many more can be implemented in short order.

Furthermore, the demo site allows for further ABR parameter tweaking on the fly, specifically for bitrate and buffer settings as they are readily made available by dash.js' API.

Implementation of Dash.js ABR algorithms

Dash.js has some very contradicting documentation, but I have found their implementations of popular algorithms, and they can be easily added into the demo. As of right now, however, the demo uses what they call a dynamic ABR algorithm, something that switches between the ThroughputRule and BOLA. Forcing one or the other is possible and will be added for the final revision.

Implementation of Custom ABR algorithms

As mentioned earlier, implementing custom algorithms is possible by following dash.js' logic for all of their default implementations. Their references provided a sample custom algorithm that forces the player into the lowest bitrate, but its structure is very similar to their ThrouputRule one. The idea for this part is to break down what goes on in the Rule classes for

Dash.js to show someone who wants to use this project the steps necessary to select a bitrate whether the information is accumulated or current.

Plans for future

The only true obstacle left is creating a completely custom ABR algorithm or using existing proposed algorithms to be converted into usable ones by Dash.js.

In addition to the custom implementation, I have a couple of options for lightweight network simulators that can artificially congest the stream for proper testing. In combination with this, I would like to add some form of metric tracking, most likely in some graphing form if time permits. Certainly, I will add a csv output of important metrics during the stream.