Setup

I setup the OBS as explained in the Medium article. Then, I created a **config** file to manage all variables from one file. After that, I created the **ffmpeg.sh** to start ffmpeg command. Then, I located the **server.sh** script to start node server. Finally, I was able to play segments on Dash.js player with default settings of keyint=120, seg_duration=4. As a result, the latency was about **4 seconds**. Since the segment duration is 4 seconds, I was expecting the latency about 9-10 seconds (glass to glass e.g. including the media capturing, media encoding, ffmpeg segmenting and network delivery). However, server responds requests using tmp file that is been currently writing by ffmpeg.

As a second step, I added timestamps to the OBS scene as it can be seen from the Figure 1. The most upper one is from browser source as suggested in the Medium article. The timestamp below that, starting with the "QR" and the QR code at the right is from generated from python script. Ffmpeg has a feature to put QR code to the frames but it is only available on OSX. Therefore, I manually created the QR codes using python. However, it is not effectively as I expected. It is not updated at every timestamp change since opencv writes the QR code to png file slower than the QR code generation.

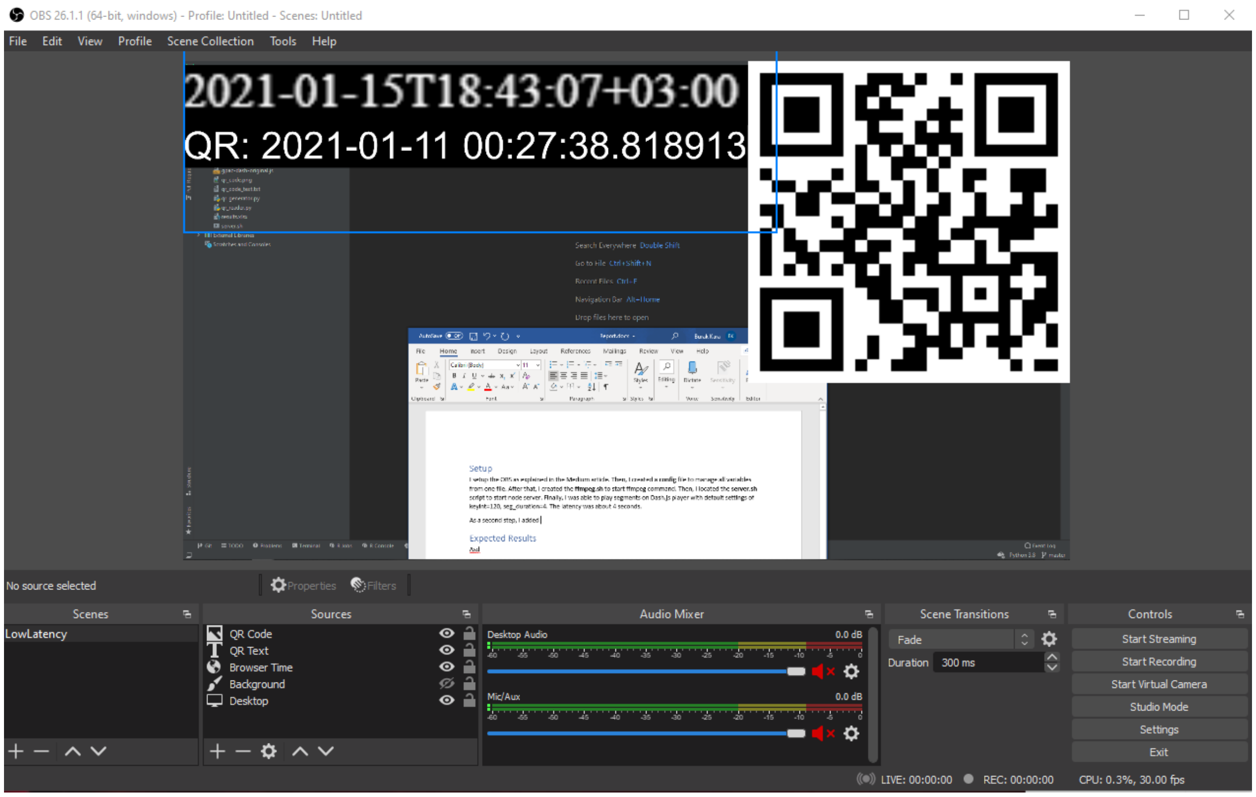
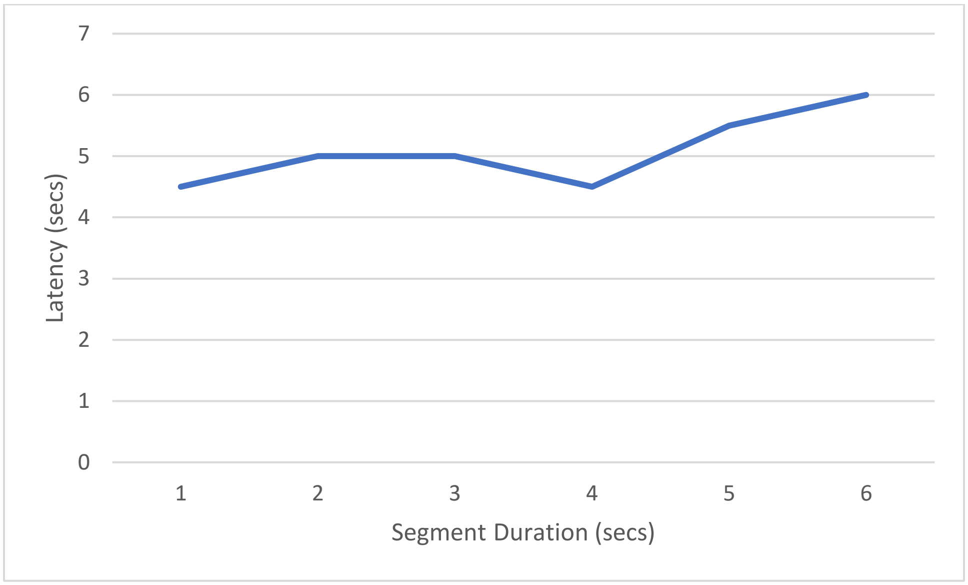


Figure 1 OBS

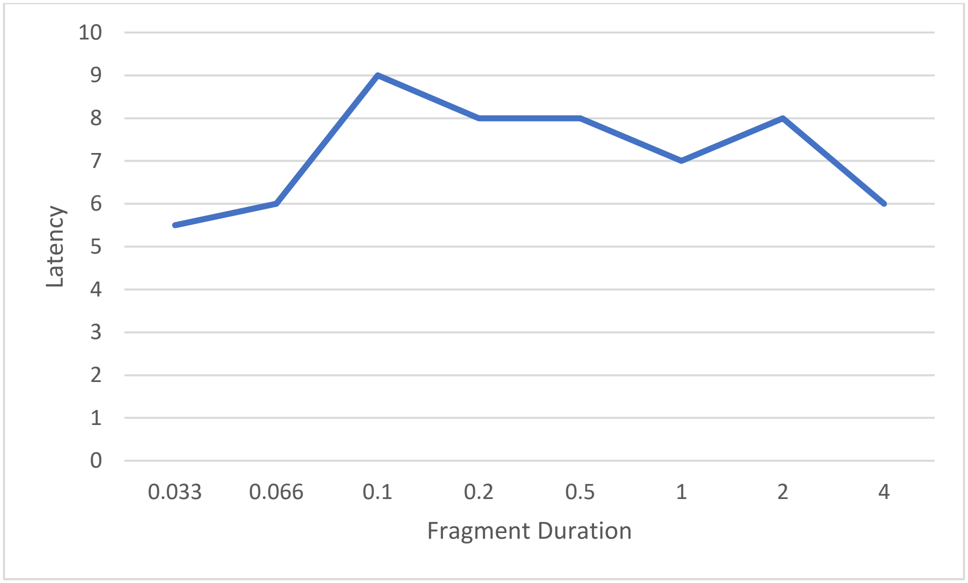
Finally, I created QR reader for my setup to measure live latency with QR code. I can scan the QR code from Dash.js player and take to time differences between scanned and current time. However, as I mentioned above, the QR code at the scene is not updated very frequently since opencv writing times.

Results

I was expecting latency results that differ depend on keyint and seg_duration settings. However, all latency results were close to **4-5 seconds**. I investigated to this and found that server designed in this way. As I mentioned in the first paragraph, while ffmpeg is writing a segment to a file (e.g. chunk-stream0-01.m4s), that file becomes temporary file and gets *tmp* extension at the end (chunk-stream0-01.m4s.tmp). Therefore, while the server is responding a coming request for chunk-stream0-01.m4s, it uses that *tmp* file (chunk-stream0-01.m4s.tmp). Therefore, I did not see any difference in the latency when I changed the segment duration since server immediately responds with tmp file that is on the fly. The different seg_duration settings only change the download duration of a segment since the duration directly affects file size.



When I changed the chunk duration, I was expecting a reduce in the latency, since the smallest value for the chunk is 0.033. When I set the frag_duration to 4 each segment contains 1 fragment exactly. Expecting all of these, setting the fragment duration in ffmpeg cause much more latency in some cases. This can be related to exceeding the default fragment size. For instance, 0.033, 0.66 and 0.1 gave the same results with not implicitly set frag duration. Also, 4 gave the similar latency with those ones. However, the other values between 0.1 and 4 gave the higher values that is not expected.


Latency
Fragment Duration