Report of Experimentation with Adaptive Streaming

Yiming Yang
yiming.2.yang@aalto.fi

***Abstract*—In this report, I deployed Dash and Hsl protocol Adaptive Streaming on AWS server. And simulated different network situation to observe the reaction of each protocol.**

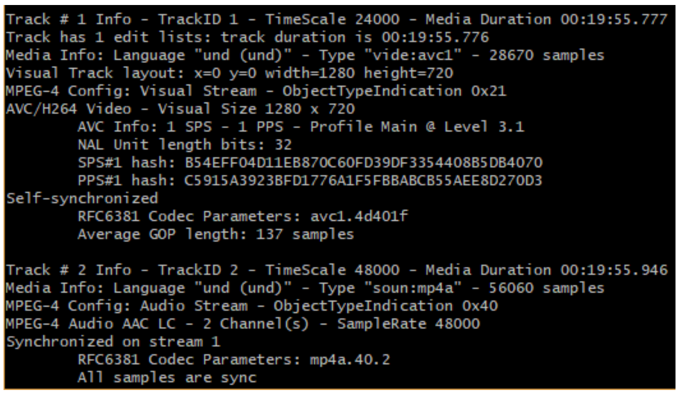
I. Introduction

The Adaptive Streaming contents are prepared according to the requirement of the protocols and deployed on AWS server, where a http server is run. Some simple web pages are set, so that the Streaming video can be accessed from a local Windows System Desktop. A simple application is used on local system to simulate lags.

II. Streaming Server Setting Up

A. Tools and Materials

- AWS server (Ubuntu Server 16.04 LTS)
- npm http-server
- original source video (detail information below)



```
Track # 1 Info - TrackID 1 - TimeScale 24000 - Media Duration 00:19:55.777
Track has 1 edit lists: track duration is 00:19:55.776
Media Info: Language "und (und)" - Type "vide:avc1" - 28670 samples
Visual Track layout: x=0 y=0 width=1280 height=720
MPEG-4 Config: Visual Stream - ObjectTypeIndication 0x21
AVC/H264 Video - Visual Size 1280 x 720
        AVC Info: 1 SPS - 1 PPS - Profile Main @ Level 3.1
        NAL Unit length bits: 32
        SPS#1 hash: B54EFF04D11EB870C60FD39DF3354408B5DB4070
        PPS#1 hash: C5915A3923BFD1776A1F5FBBABCB55AEE8D270D3
Self-synchronized
        RFC6381 Codec Parameters: avc1.4d401f
        Average GOP length: 137 samples

Track # 2 Info - TrackID 2 - TimeScale 48000 - Media Duration 00:19:55.946
Media Info: Language "und (und)" - Type "soun:mp4a" - 56060 samples
MPEG-4 Config: Audio Stream - ObjectTypeIndication 0x40
MPEG-4 Audio AAC LC - 2 Channel(s) - SampleRate 48000
Synchronized on stream 1
        RFC6381 Codec Parameters: mp4a.40.2
        All samples are sync
```

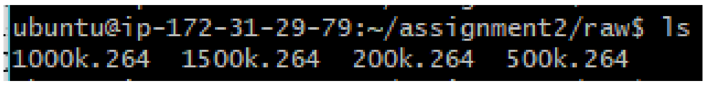
- x264 encoder
- MP4Box
- ffmpeg
- dash.js
- hsl.js

B. MPEG-DASH

Transcode original.mp4 to different bandwidth with x264 encoder and set keyframe every 48 frames:

x264 --output 500k.264 --fps 24 --preset slow --bitrate 500 --vbv-maxrate 1000 --vbv-bufsize 2000 --min-keyint 48 --keyint 48 --scenecut 0 --no-scenecut --pass 1 --video-filter "resize:width=1280,height=720" original.mp4

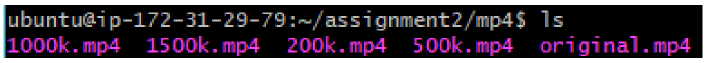
Repeat this command with different bandwidth settings. Then we get raw video streams of 4 bandwidths: 200k/500k/1000k/1500k.



```
ubuntu@ip-172-31-29-79:~/assignment2/raw$ ls
1000k.264  1500k.264  200k.264  500k.264
```

We package them into MP4 container using MP4Box, we also keep the original video file for its audio track:

MP4Box -add 500k.264 -fps 500k.mp4



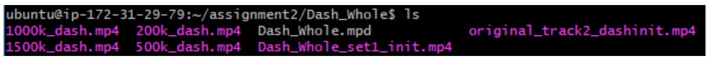
```
ubuntu@ip-172-31-29-79:~/assignment2/mp4$ ls
1000k.mp4  1500k.mp4  200k.mp4  500k.mp4  original.mp4
```

Now we need to enable segmentation for Streaming. Each segment is set to be 4 second with random access point. From here, we have 2 options. One is to store all segments in a fragmented-MP4 file. The other one is to store each segment separately in M4S file. I implemented them both:

1. fragmented-MP4

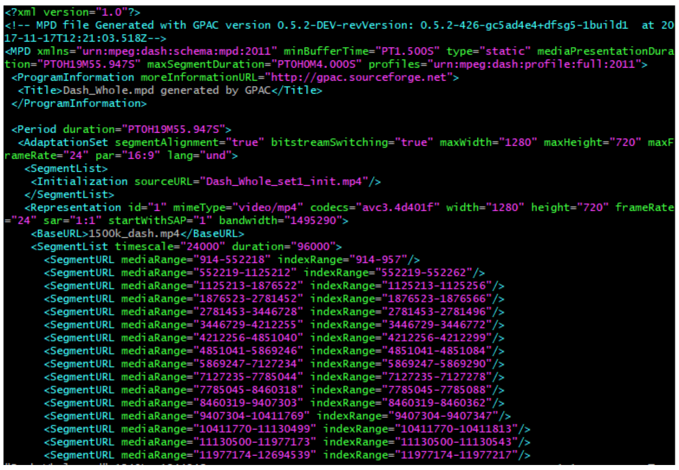
   MP4Box -dash 4000 -rap -frag-rap -out ../Dash_Whole/Dash_Whole.mpd 1500k.mp4 1000k.mp4 500k.mp4 200k.mp4 original.mp4#audio

   Each Video/Audio track is stored in a fragmented-MP4 file, the MPD file is automatically formed by MP4Box as manifest file for Streaming.



```
ubuntu@ip-172-31-29-79:~/assignment2/Dash_Whole$ ls
1000k_dash.mp4  200k_dash.mp4  Dash_Whole.mpd             original_track2_dashinit.mp4
1500k_dash.mp4  500k_dash.mp4  Dash_Whole_set1_init.mp4
```

The MPD file looks like:


```
<?xml version="1.0"?>
<!-- MPD file Generated with GPAC version 0.5.2-DEV-revVersion: 0.5.2-426-gc5ad4e4+dfsg5-1build1  at 20
17-11-17T12:21:03.518Z-->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.500S" type="static" mediaPresentationDura
tion="PT0H19M55.947S" maxSegmentDuration="PT0H0M4.000S" profiles="urn:mpeg:dash:profile:full:2011">
 <ProgramInformation moreInformationURL="http://gpac.sourceforge.net">
  <Title>Dash_Whole.mpd generated by GPAC</Title>
 </ProgramInformation>

 <Period duration="PT0H19M55.947S">
  <AdaptationSet segmentAlignment="true" bitstreamSwitching="true" maxWidth="1280" maxHeight="720" maxF
rameRate="24" par="16:9" lang="und">
   <SegmentList>
    <Initialization sourceURL="Dash_Whole_set1_init.mp4"/>
   </SegmentList>
   <Representation id="1" mimeType="video/mp4" codecs="avc3.4d401f" width="1280" height="720" frameRate
="24" sar="1:1" startWithSAP="1" bandwidth="1495290">
    <BaseURL>1500k_dash.mp4</BaseURL>
    <SegmentList timescale="24000" duration="96000">
      <SegmentURL mediaRange="914-552218" indexRange="914-957"/>
      <SegmentURL mediaRange="552219-1125212" indexRange="552219-552262"/>
      <SegmentURL mediaRange="1125213-1876522" indexRange="1125213-1125256"/>
      <SegmentURL mediaRange="1876523-2781452" indexRange="1876523-1876566"/>
      <SegmentURL mediaRange="2781453-3446728" indexRange="2781453-2781496"/>
      <SegmentURL mediaRange="3446729-4212255" indexRange="3446729-3446772"/>
      <SegmentURL mediaRange="4212256-4851040" indexRange="4212256-4212299"/>
      <SegmentURL mediaRange="4851041-5869246" indexRange="4851041-4851084"/>
      <SegmentURL mediaRange="5869247-7127234" indexRange="5869247-5869290"/>
      <SegmentURL mediaRange="7127235-7785044" indexRange="7127235-7127278"/>
      <SegmentURL mediaRange="7785045-8460318" indexRange="7785045-7785088"/>
      <SegmentURL mediaRange="8460319-9407303" indexRange="8460319-8460362"/>
      <SegmentURL mediaRange="9407304-10411769" indexRange="9407304-9407347"/>
      <SegmentURL mediaRange="10411770-11130499" indexRange="10411770-10411813"/>
      <SegmentURL mediaRange="11130500-11977173" indexRange="11130500-11130543"/>
      <SegmentURL mediaRange="11977174-12694539" indexRange="11977174-11977217"/>
```

(…)

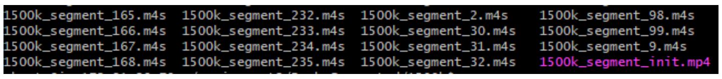
2. separate files

   We generate segments and MPD file separately for each Video/Audio track:

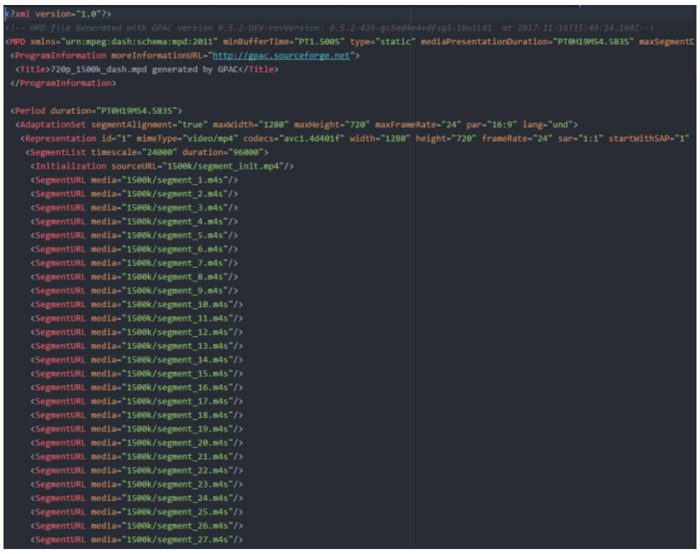
   MP4Box -dash 4000 -frag 4000 -rap -segment-name 1500k/1500k_segment_ 1500k.mp4

   M4S files are created to store each segment, a MPD file is also generated for each track.

(..skipped)


```
1500k_segment_165.m4s  1500k_segment_232.m4s  1500k_segment_2.m4s   1500k_segment_98.m4s
1500k_segment_166.m4s  1500k_segment_233.m4s  1500k_segment_30.m4s  1500k_segment_99.m4s
1500k_segment_167.m4s  1500k_segment_234.m4s  1500k_segment_31.m4s  1500k_segment_9.m4s
1500k_segment_168.m4s  1500k_segment_235.m4s  1500k_segment_32.m4s  1500k_segment_init.mp4
```

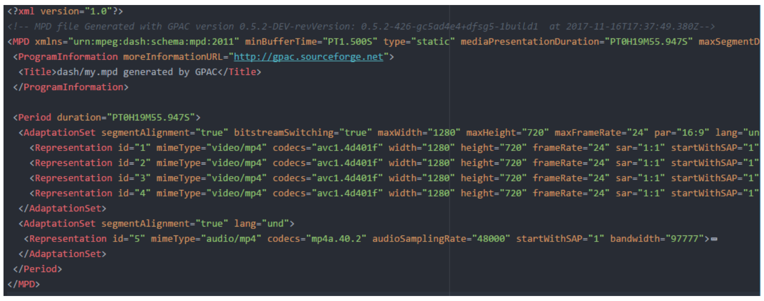
The MPD file for a single track looks like:


```
<?xml version="1.0"?>
<!-- MPD file Generated with GPAC version 0.5.2-DEV-revVersion: 0.5.2-426-gc5ad4e4+dfsg5-1build1  at 2017-11-16T15:49:14.160Z-->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.500S" type="static" mediaPresentationDuration="PT0H19M54.583S" maxSegmentD
 <ProgramInformation moreInformationURL="http://gpac.sourceforge.net">
  <Title>720p_1500k_dash.mpd generated by GPAC</Title>
 </ProgramInformation>

 <Period duration="PT0H19M54.583S">
  <AdaptationSet segmentAlignment="true" maxWidth="1280" maxHeight="720" maxFrameRate="24" par="16:9" lang="und">
   <Representation id="1" mimeType="video/mp4" codecs="avc1.4d401f" width="1280" height="720" frameRate="24" sar="1:1" startWithSAP="1"
    <SegmentList timescale="24000" duration="96000">
     <Initialization sourceURL="1500k/segment_init.mp4"/>
     <SegmentURL media="1500k/segment_1.m4s"/>
     <SegmentURL media="1500k/segment_2.m4s"/>
     <SegmentURL media="1500k/segment_3.m4s"/>
     <SegmentURL media="1500k/segment_4.m4s"/>
     <SegmentURL media="1500k/segment_5.m4s"/>
     <SegmentURL media="1500k/segment_6.m4s"/>
     <SegmentURL media="1500k/segment_7.m4s"/>
     <SegmentURL media="1500k/segment_8.m4s"/>
     <SegmentURL media="1500k/segment_9.m4s"/>
     <SegmentURL media="1500k/segment_10.m4s"/>
     <SegmentURL media="1500k/segment_11.m4s"/>
     <SegmentURL media="1500k/segment_12.m4s"/>
     <SegmentURL media="1500k/segment_13.m4s"/>
     <SegmentURL media="1500k/segment_14.m4s"/>
     <SegmentURL media="1500k/segment_15.m4s"/>
     <SegmentURL media="1500k/segment_16.m4s"/>
     <SegmentURL media="1500k/segment_17.m4s"/>
     <SegmentURL media="1500k/segment_18.m4s"/>
     <SegmentURL media="1500k/segment_19.m4s"/>
     <SegmentURL media="1500k/segment_20.m4s"/>
     <SegmentURL media="1500k/segment_21.m4s"/>
     <SegmentURL media="1500k/segment_22.m4s"/>
     <SegmentURL media="1500k/segment_23.m4s"/>
     <SegmentURL media="1500k/segment_24.m4s"/>
     <SegmentURL media="1500k/segment_25.m4s"/>
     <SegmentURL media="1500k/segment_26.m4s"/>
     <SegmentURL media="1500k/segment_27.m4s"/>
```

(…)

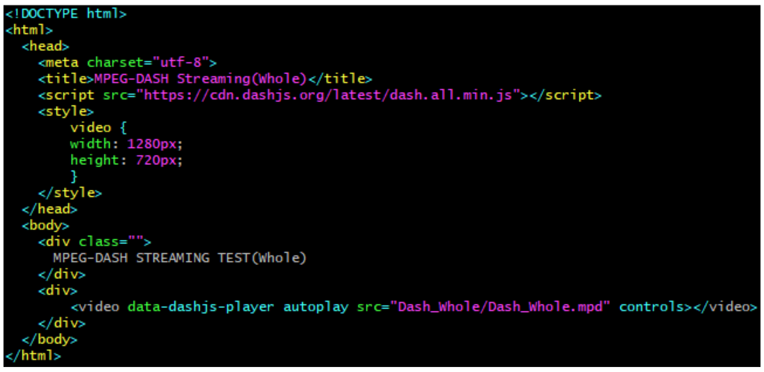
Here we need to manually combine the MPD files together into one MPD file as the manifest file. This could be simply done by copy Representation (each for every bandwidth) tags and AdaptationSet (each for Audio/Video) tags.

The combined MPD file looks like:

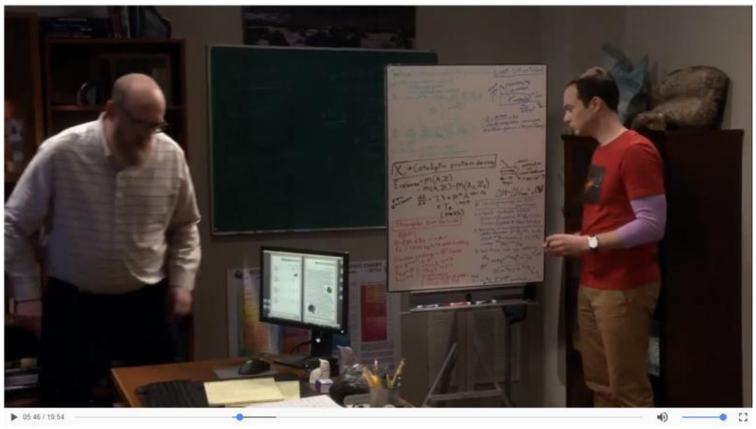

```
<?xml version="1.0"?>
<!-- MPD file Generated with GPAC version 0.5.2-DEV-revVersion: 0.5.2-426-gc5ad4e4+dfsg5-1build1  at 2017-11-16T17:37:49.380Z-->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.500S" type="static" mediaPresentationDuration="PT0H19M55.947S" maxSegmentD
 <ProgramInformation moreInformationURL="http://gpac.sourceforge.net">
  <Title>dash/my.mpd generated by GPAC</Title>
 </ProgramInformation>

 <Period duration="PT0H19M55.947S">
  <AdaptationSet segmentAlignment="true" bitstreamSwitching="true" maxWidth="1280" maxHeight="720" maxFrameRate="24" par="16:9" lang="un
    <Representation id="1" mimeType="video/mp4" codecs="avc1.4d401f" width="1280" height="720" frameRate="24" sar="1:1" startWithSAP="1"
    <Representation id="2" mimeType="video/mp4" codecs="avc1.4d401f" width="1280" height="720" frameRate="24" sar="1:1" startWithSAP="1"
    <Representation id="3" mimeType="video/mp4" codecs="avc1.4d401f" width="1280" height="720" frameRate="24" sar="1:1" startWithSAP="1"
    <Representation id="4" mimeType="video/mp4" codecs="avc1.4d401f" width="1280" height="720" frameRate="24" sar="1:1" startWithSAP="1"
  </AdaptationSet>
  <AdaptationSet segmentAlignment="true" lang="und">
   <Representation id="5" mimeType="audio/mp4" codecs="mp4a.40.2" audioSamplingRate="48000" startWithSAP="1" bandwidth="97777">
  </AdaptationSet>
 </Period>
</MPD>
```

After we generated the content for Streaming and MPD files. We finally can set up for Streaming.

We use Video tag in HTML5 for playing. Also, we use Dash.js as HTML5 player in order for better compatibility.


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>MPEG-DASH Streaming(Whole)</title>
    <script src="https://cdn.dashjs.org/latest/dash.all.min.js"></script>
    <style>
        video {
        width: 1280px;
        height: 720px;
        }
    </style>
  </head>
  <body>
    <div class="">
      MPEG-DASH STREAMING TEST(Whole)
    </div>
    <div>
        <video data-dashjs-player autoplay src="Dash_Whole/Dash_Whole.mpd" controls></video>
    </div>
  </body>
</html>
```

I tested on Windows(Edge+Chrome) and Android(Firefox+Chrome). The playing and random accessing all works well.



C. HLS(HTTP Live Streaming)

Since we already have original video with different bandwidth. We may directly enable segmentation for HLS protocol.

For video track:

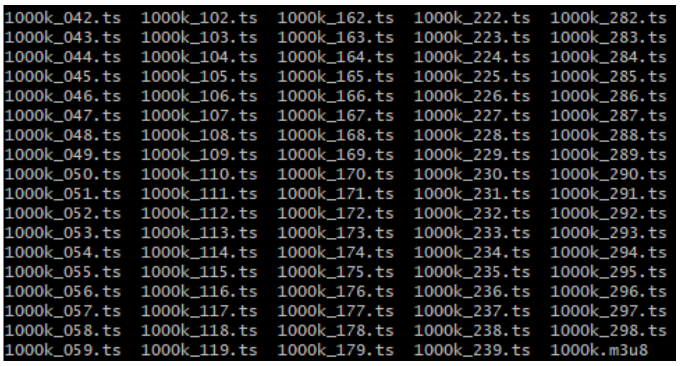
ffmpeg -i 200k.mp4 -c:v h264 -flags +cgop -g 48 -hls_time 4 -hls_list_size 0 -hls_segment_filename '200k_%03d.ts' 200k.m3u8

For audio track:

ffmpeg -i original.mp4 -vn -c:a aac -flags +cgop -g 48 -hls_time 4 -hls_list_size 0 -hls_segment_filename 'audio_%03d.ts' -strict -2 audio.m3u8

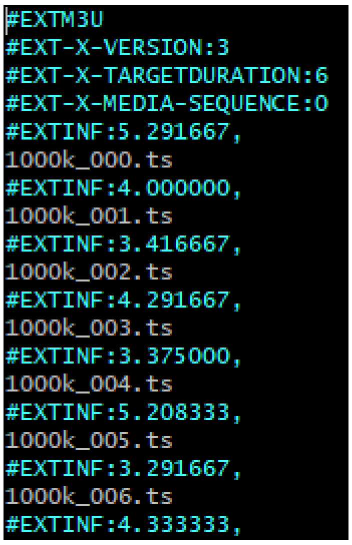
Segments are stored in TS files, m3u8 files are generated as playlist.

(..skipped)



```
1000k_042.ts  1000k_102.ts  1000k_162.ts  1000k_222.ts  1000k_282.ts
1000k_043.ts  1000k_103.ts  1000k_163.ts  1000k_223.ts  1000k_283.ts
1000k_044.ts  1000k_104.ts  1000k_164.ts  1000k_224.ts  1000k_284.ts
1000k_045.ts  1000k_105.ts  1000k_165.ts  1000k_225.ts  1000k_285.ts
1000k_046.ts  1000k_106.ts  1000k_166.ts  1000k_226.ts  1000k_286.ts
1000k_047.ts  1000k_107.ts  1000k_167.ts  1000k_227.ts  1000k_287.ts
1000k_048.ts  1000k_108.ts  1000k_168.ts  1000k_228.ts  1000k_288.ts
1000k_049.ts  1000k_109.ts  1000k_169.ts  1000k_229.ts  1000k_289.ts
1000k_050.ts  1000k_110.ts  1000k_170.ts  1000k_230.ts  1000k_290.ts
1000k_051.ts  1000k_111.ts  1000k_171.ts  1000k_231.ts  1000k_291.ts
1000k_052.ts  1000k_112.ts  1000k_172.ts  1000k_232.ts  1000k_292.ts
1000k_053.ts  1000k_113.ts  1000k_173.ts  1000k_233.ts  1000k_293.ts
1000k_054.ts  1000k_114.ts  1000k_174.ts  1000k_234.ts  1000k_294.ts
1000k_055.ts  1000k_115.ts  1000k_175.ts  1000k_235.ts  1000k_295.ts
1000k_056.ts  1000k_116.ts  1000k_176.ts  1000k_236.ts  1000k_296.ts
1000k_057.ts  1000k_117.ts  1000k_177.ts  1000k_237.ts  1000k_297.ts
1000k_058.ts  1000k_118.ts  1000k_178.ts  1000k_238.ts  1000k_298.ts
1000k_059.ts  1000k_119.ts  1000k_179.ts  1000k_239.ts  1000k.m3u8
```

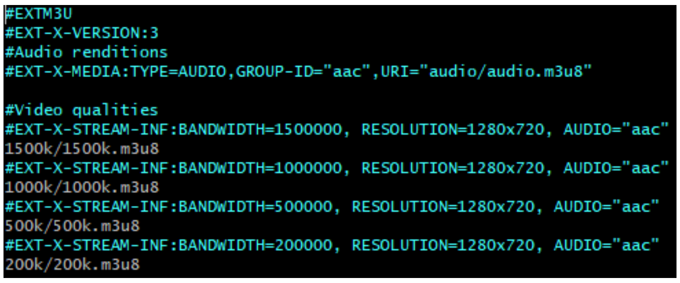
Inside the m3u8 file:



```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:5.291667,
1000k_000.ts
#EXTINF:4.000000,
1000k_001.ts
#EXTINF:3.416667,
1000k_002.ts
#EXTINF:4.291667,
1000k_003.ts
#EXTINF:3.375000,
1000k_004.ts
#EXTINF:5.208333,
1000k_005.ts
#EXTINF:3.291667,
1000k_006.ts
#EXTINF:4.333333,
```

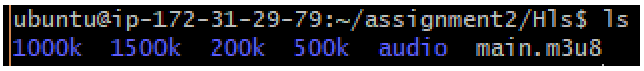
(…)

We manually create a main.m3u8 as the manifest file. This m3u8 file acts as an index and leads to other m3u8 separate files:



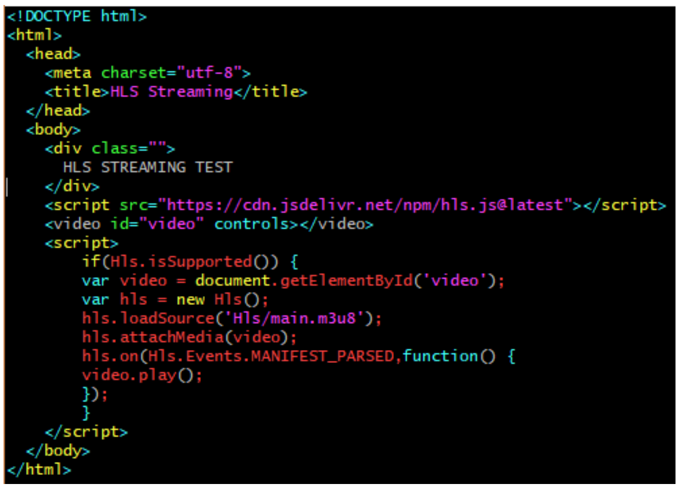
```
#EXTM3U
#EXT-X-VERSION:3
#Audio renditions
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac",URI="audio/audio.m3u8"

#Video qualities
#EXT-X-STREAM-INF:BANDWIDTH=1500000, RESOLUTION=1280x720, AUDIO="aac"
1500k/1500k.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=1000000, RESOLUTION=1280x720, AUDIO="aac"
1000k/1000k.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=500000, RESOLUTION=1280x720, AUDIO="aac"
500k/500k.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=200000, RESOLUTION=1280x720, AUDIO="aac"
200k/200k.m3u8
```

The structure of HLS Streaming content looks like:



```
ubuntu@ip-172-31-29-79:~/assignment2/Hls$ ls
1000k  1500k  200k  500k  audio  main.m3u8
```

Now, we create a web page with HTML5 Video tag and Hls.js player.

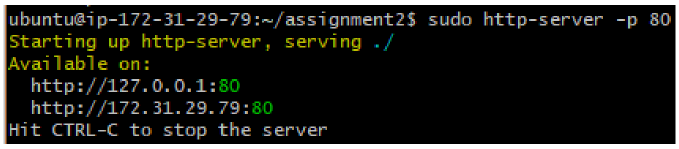


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>HLS Streaming</title>
  </head>
  <body>
    <div class="">
      HLS STREAMING TEST
    </div>
    <script src="https://cdn.jsdelivr.net/npm/hls.js@latest"></script>
    <video id="video" controls></video>
    <script>
        if(Hls.isSupported()) {
        var video = document.getElementById('video');
        var hls = new Hls();
        hls.loadSource('Hls/main.m3u8');
        hls.attachMedia(video);
        hls.on(Hls.Events.MANIFEST_PARSED,function() {
        video.play();
        });
        }
    </script>
  </body>
</html>
```

Video playing and random access are tested on Windows(Edge+Chrome) and Android(Firefox+Chrome).

III. Test Under Bad Network Connection Situation

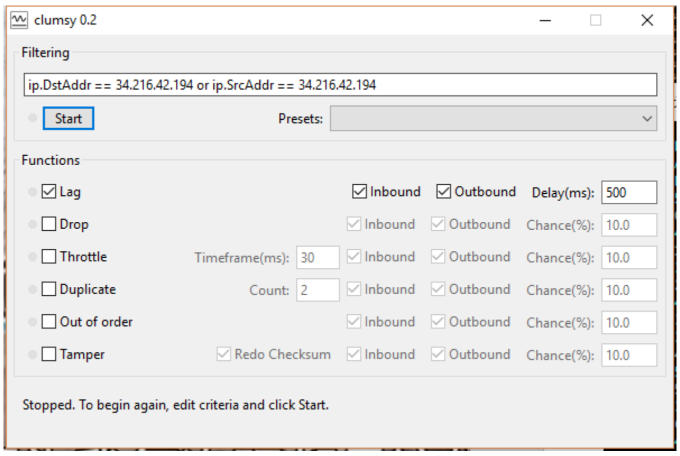
We run a basic npm http server.



```
ubuntu@ip-172-31-29-79:~/assignment2$ sudo http-server -p 80
Starting up http-server, serving ./
Available on:
  http://127.0.0.1:80
  http://172.31.29.79:80
Hit CTRL-C to stop the server
```

So we can access the test web pages through Internet.

Now we use a Windows application called ***clumsy*** to simulate bad network connection:



ip.DstAddr == 34.216.42.194 or ip.SrcAddr == 34.216.42.194

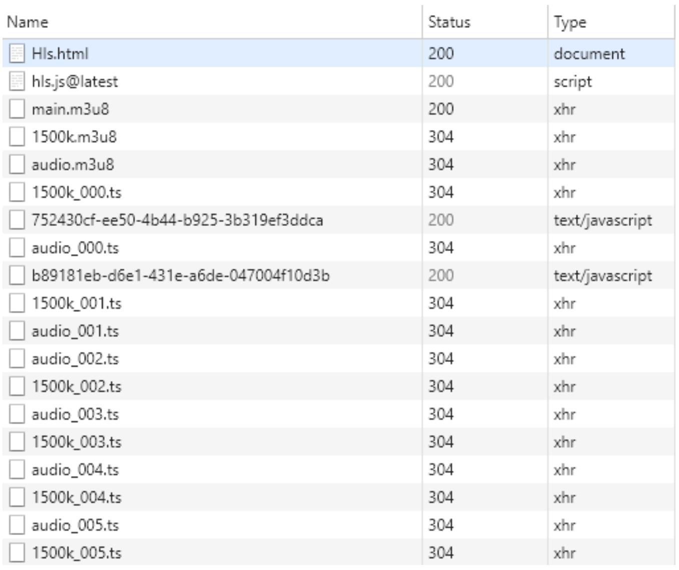
☑ Lag		☑ Inbound	☑ Outbound	Delay(ms):	500
☐ Drop		✓ Inbound	✓ Outbound	Chance(%):	10.0
☐ Throttle	Timeframe(ms): 30	✓ Inbound	✓ Outbound	Chance(%):	10.0
☐ Duplicate	Count: 2	✓ Inbound	✓ Outbound	Chance(%):	10.0
☐ Out of order		✓ Inbound	✓ Outbound	Chance(%):	10.0
☐ Tamper	✓ Redo Checksum	✓ Inbound	✓ Outbound	Chance(%):	10.0

Stopped. To begin again, edit criteria and click Start.

In order to control the available bandwidth per second, we add lag to all traffic from and to the server ip address. As a result, the bandwidth transported in a unit time is reduced. We use Chrome network inspect to observe the behave of different protocols.

1) Delag = 0ms

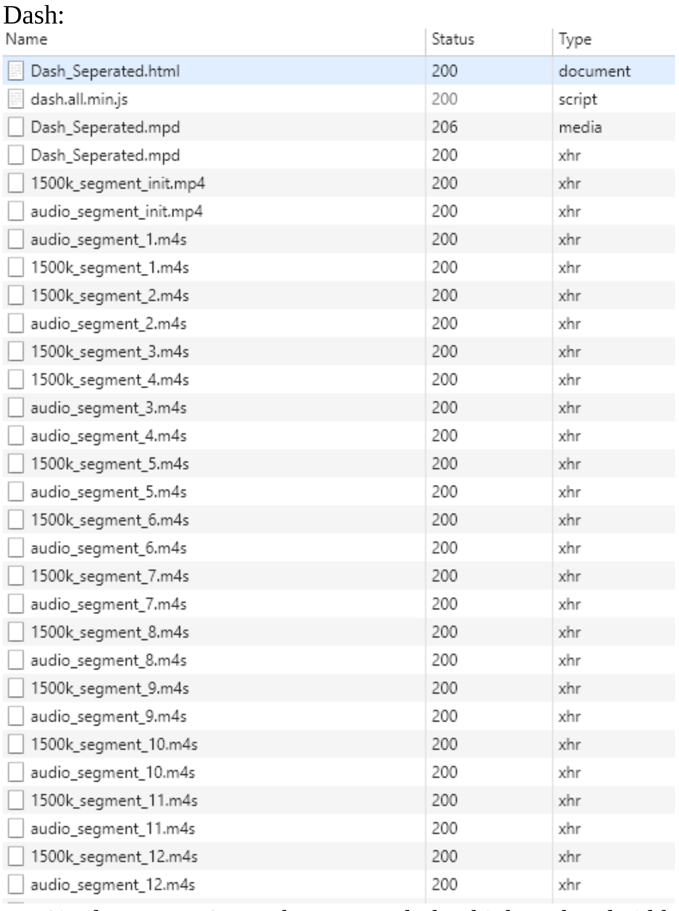
HLS:



Name	Status	Type
Hls.html	200	document
hls.js@latest	200	script
main.m3u8	200	xhr
1500k.m3u8	304	xhr
audio.m3u8	304	xhr
1500k_000.ts	304	xhr
752430cf-ee50-4b44-b925-3b319ef3ddca	200	text/javascript
audio_000.ts	304	xhr
b89181eb-d6e1-431e-a6de-047004f10d3b	200	text/javascript
1500k_001.ts	304	xhr
audio_001.ts	304	xhr
audio_002.ts	304	xhr
1500k_002.ts	304	xhr
audio_003.ts	304	xhr
1500k_003.ts	304	xhr
audio_004.ts	304	xhr
1500k_004.ts	304	xhr
audio_005.ts	304	xhr
1500k_005.ts	304	xhr

We can easily find that the best quality is requested and responded when these is no delay. Video and Audio track is downloaded under the same frame from their segments. Main.m3u8 is requested first, then 1500k.m3u8 and audio.m3u8.

Dash:

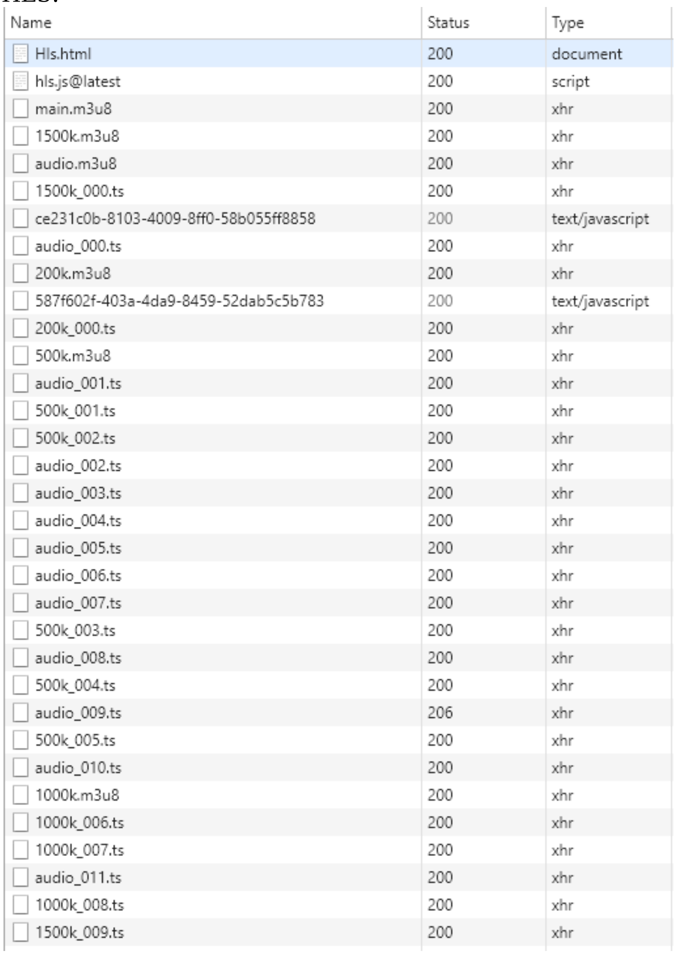


Name	Status	Type
Dash_Seperated.html	200	document
dash.all.min.js	200	script
Dash_Seperated.mpd	206	media
Dash_Seperated.mpd	200	xhr
1500k_segment_init.mp4	200	xhr
audio_segment_init.mp4	200	xhr
audio_segment_1.m4s	200	xhr
1500k_segment_1.m4s	200	xhr
1500k_segment_2.m4s	200	xhr
audio_segment_2.m4s	200	xhr
1500k_segment_3.m4s	200	xhr
1500k_segment_4.m4s	200	xhr
audio_segment_3.m4s	200	xhr
audio_segment_4.m4s	200	xhr
1500k_segment_5.m4s	200	xhr
audio_segment_5.m4s	200	xhr
1500k_segment_6.m4s	200	xhr
audio_segment_6.m4s	200	xhr
1500k_segment_7.m4s	200	xhr
audio_segment_7.m4s	200	xhr
1500k_segment_8.m4s	200	xhr
audio_segment_8.m4s	200	xhr
1500k_segment_9.m4s	200	xhr
audio_segment_9.m4s	200	xhr
1500k_segment_10.m4s	200	xhr
audio_segment_10.m4s	200	xhr
1500k_segment_11.m4s	200	xhr
audio_segment_11.m4s	200	xhr
1500k_segment_12.m4s	200	xhr
audio_segment_12.m4s	200	xhr

Similar to HLS, Dash requested the highest bandwidth when there's no lag. Segments of Video and Audio are requested. MPD file is requested at the beginning.

2) *Delag = 250ms*

HLS:

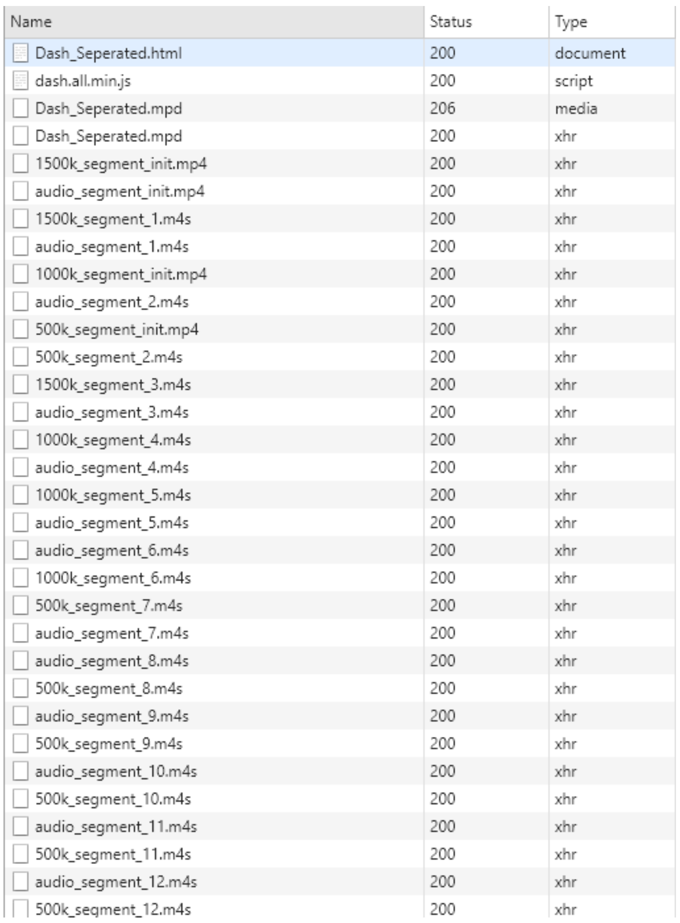


Name	Status	Type
Hls.html	200	document
hls.js@latest	200	script
main.m3u8	200	xhr
1500k.m3u8	200	xhr
audio.m3u8	200	xhr
1500k_000.ts	200	xhr
ce231c0b-8103-4009-8ff0-58b055ff8858	200	text/javascript
audio_000.ts	200	xhr
200k.m3u8	200	xhr
587f602f-403a-4da9-8459-52dab5c5b783	200	text/javascript
200k_000.ts	200	xhr
500k.m3u8	200	xhr
audio_001.ts	200	xhr
500k_001.ts	200	xhr
500k_002.ts	200	xhr
audio_002.ts	200	xhr
audio_003.ts	200	xhr
audio_004.ts	200	xhr
audio_005.ts	200	xhr
audio_006.ts	200	xhr
audio_007.ts	200	xhr
500k_003.ts	200	xhr
audio_008.ts	200	xhr
500k_004.ts	200	xhr
audio_009.ts	206	xhr
500k_005.ts	200	xhr
audio_010.ts	200	xhr
1000k.m3u8	200	xhr
1000k_006.ts	200	xhr
1000k_007.ts	200	xhr
audio_011.ts	200	xhr
1000k_008.ts	200	xhr
1500k_009.ts	200	xhr

We can find that in HLS. The highest quality is still requested at the very beginning. However, the next requested is the lowest bandwidth because of lag. We can also see the request of Video and Audio is individual from each other. When video is loaded ahead of watching, HLS still try to download better quality video. The speeding up is slow but continuous and try to find a balance.

Note that if I add lag suddenly during playing, it acts the same as lag at the beginning.

Dash:

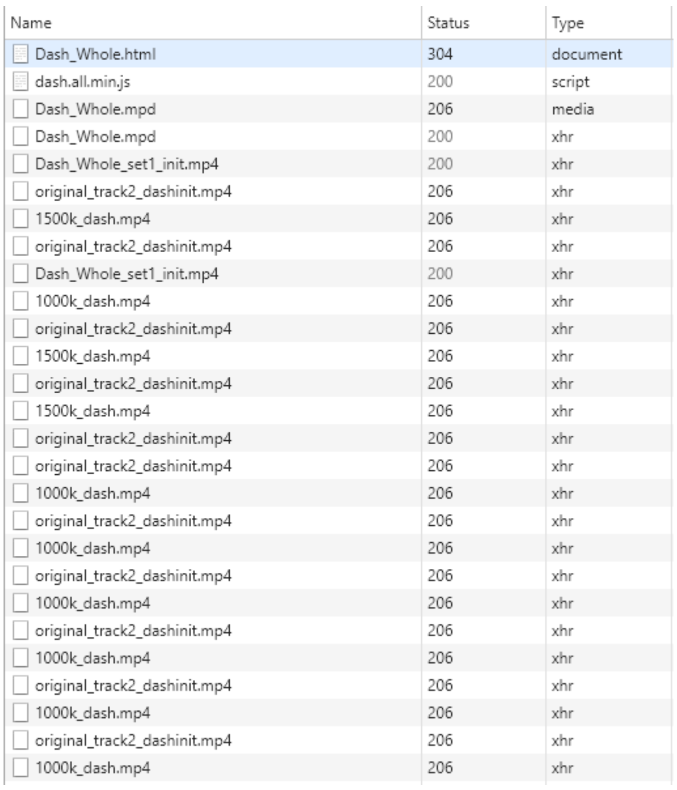


Name	Status	Type
Dash_Seperated.html	200	document
dash.all.min.js	200	script
Dash_Seperated.mpd	206	media
Dash_Seperated.mpd	200	xhr
1500k_segment_init.mp4	200	xhr
audio_segment_init.mp4	200	xhr
1500k_segment_1.m4s	200	xhr
audio_segment_1.m4s	200	xhr
1000k_segment_init.mp4	200	xhr
audio_segment_2.m4s	200	xhr
500k_segment_init.mp4	200	xhr
500k_segment_2.m4s	200	xhr
1500k_segment_3.m4s	200	xhr
audio_segment_3.m4s	200	xhr
1000k_segment_4.m4s	200	xhr
audio_segment_4.m4s	200	xhr
1000k_segment_5.m4s	200	xhr
audio_segment_5.m4s	200	xhr
audio_segment_6.m4s	200	xhr
1000k_segment_6.m4s	200	xhr
500k_segment_7.m4s	200	xhr
audio_segment_7.m4s	200	xhr
audio_segment_8.m4s	200	xhr
500k_segment_8.m4s	200	xhr
audio_segment_9.m4s	200	xhr
500k_segment_9.m4s	200	xhr
audio_segment_10.m4s	200	xhr
500k_segment_10.m4s	200	xhr
audio_segment_11.m4s	200	xhr
500k_segment_11.m4s	200	xhr
audio_segment_12.m4s	200	xhr
500k_segment_12.m4s	200	xhr

Dash also try to get the highest quality in the beginning. However, because of lag, it lower its request to 1000k bandwidth. If the bandwidth is still too high, it will switch to a even lower quality, until there's a balance.

IV. Other Comparation

Behavior of Dash (fragmented MP4)



Name	Status	Type
Dash_Whole.html	304	document
dash.all.min.js	200	script
Dash_Whole.mpd	206	media
Dash_Whole.mpd	200	xhr
Dash_Whole_set1_init.mp4	200	xhr
original_track2_dashinit.mp4	206	xhr
1500k_dash.mp4	206	xhr
original_track2_dashinit.mp4	206	xhr
Dash_Whole_set1_init.mp4	200	xhr
1000k_dash.mp4	206	xhr
original_track2_dashinit.mp4	206	xhr
1500k_dash.mp4	206	xhr
original_track2_dashinit.mp4	206	xhr
1500k_dash.mp4	206	xhr
original_track2_dashinit.mp4	206	xhr
original_track2_dashinit.mp4	206	xhr
1000k_dash.mp4	206	xhr
original_track2_dashinit.mp4	206	xhr
1000k_dash.mp4	206	xhr
original_track2_dashinit.mp4	206	xhr
1000k_dash.mp4	206	xhr
original_track2_dashinit.mp4	206	xhr
1000k_dash.mp4	206	xhr
original_track2_dashinit.mp4	206	xhr
1000k_dash.mp4	206	xhr
original_track2_dashinit.mp4	206	xhr
1000k_dash.mp4	206	xhr

a.

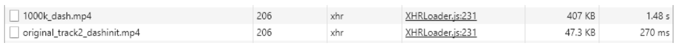
When segments are saved in a whole fragmented Mp4 file. Each http request if sent for the Mp4 file. However, inside the request, there's a range to make sure only wanted fragments are responded.

In fact, the network behaviors of two different implementations are the same in network meaning. But different in the storage structure. The range requested function must be available for the http server.

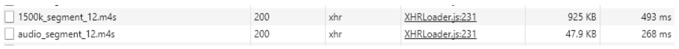
Also note that under same network condition, the quality is different under protocols and implementations.

There may be because of the different size of each downloaded package and the ability of the server to handle different request.

Evidence:



1000k_dash.mp4	206	xhr	XHRLoader.js:231	407 KB	1.48 s
original_track2_dashinit.mp4	206	xhr	XHRLoader.js:231	47.3 KB	270 ms



1500k_segment_12.m4s	200	xhr	XHRLoader.js:231	925 KB	493 ms
audio_segment_12.m4s	200	xhr	XHRLoader.js:231	47.9 KB	268 ms

Downloading of range request is slower, this cause the lower quality of video.

References

Useful websites:

1. https://blog.streamroot.io/encode-multi-bitrate-videos-mpeg-dash-mse-based-media-players/
2. https://developer.mozilla.org/en-US/Apps/Fundamentals/Audio_and_video_delive

ry/Setting_up_adaptive_streaming_media_sources#MPEG-DASH_Encoding
3. https://bitmovin.com/mp4box-dash-content-generation-x264/
4. http://www.tothenew.com/blog/adaptive-video-streaming-hls/
5. https://ffmpeg.org/ffmpeg-formats.html#hls-2
6. http://jagt.github.io/clumsy/index.html