OPEN - SOURCE REAL - TIME VIDEO PLAYER

Team Roles

PROJECT LEADER
BRIAN HERNANDEZ

CUSTOMER LIAISON & QA
MIRASOL DAVILA

DOCUMENTATION LEAD
WENDY JOYA

SOFTWARE DEVELOPER
ASHLEY JETTY

SOFTWARE DEVELOPER
ISRAEL LOPEZ

SOFTWARE DEVELOPER
DAVID MELENDEZ

SOFTWARE DEVELOPER
TIM ELLIS

SOFTWARE DEVELOPER
RAFAEL MENDOZA

SOFTWARE DEVELOPER
JEFF LUU

Faculty

FACULTY ADVISOR

DR. NEGIN FOROUZESH

Liaisons

AT&T LIAISON

MANUEL BRIAND

AT&T LIAISON

PESHALA PAHALAWATTA

AT&T LIAISON

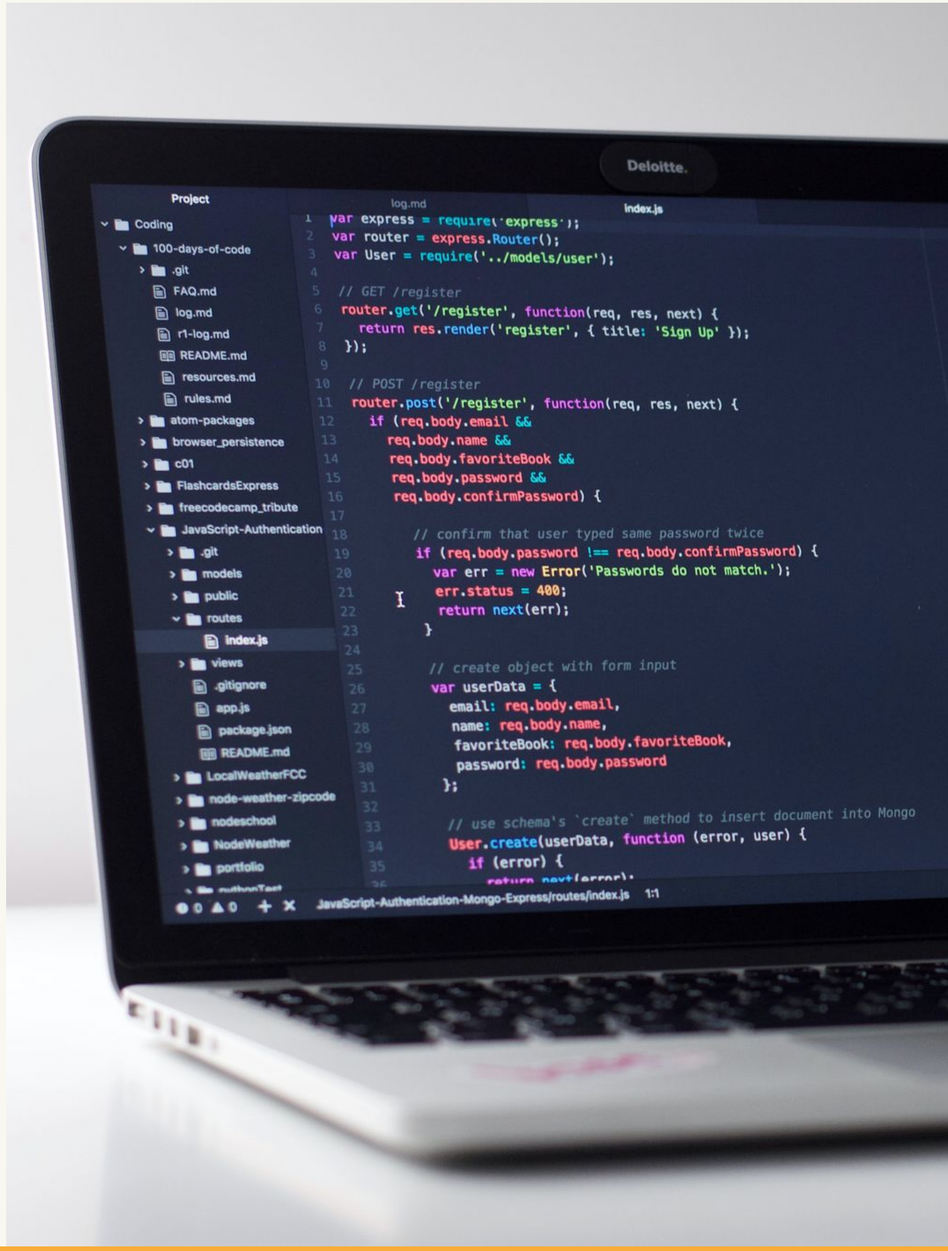
ROSS CASTILLO

AT&T LIAISON

SHAMIK MAITRA

Overview

- ABOUT THE PROJECT
- PROJECT GOALS (PROPOSAL)
- OPEN-SOURCE VIDEO PLAYERS
- ADAPTIVE BITRATE ALGORITHM
- WHAT IS HLS AND DASH?
- QUANTITATIVE METRICS
- HLS.JS VIDEO PLAYER DEMO
- ACCOMPLISHED & FUTURE GOALS



About Our Project:

Open - Source

Real - Time

Video Player

POPULAR OPEN-SOURCE VIDEO PLAYERS:

- ExoPlayer
- Shaka
- HLS.js
- Video.js

OPEN-SOURCE

- We are selecting an open source video player and extending it to meet AT&T's proposed capabilities

WHAT IS A REAL-TIME VIDEO PLAYER?

- Allows you to stream videos and gives you real-time information about the video playback

PROJECT GOAL

- Extend at least one open-source video player
- Add more features, such as new metrics, new graphs, option to select a network profile, option export data

Project Proposal

Mirasol Davila

Project Goals (Proposal)

Extending HLS.js open-source video player

- Display Video Startup Time (VST in sec)
- Display Rebuffering Ratio (%)
- Display VMAF (and/or SSIM) score of the selected video track
- Enable support for both HLS and DASH
- Simulate real-time throttling of the network conditions
- Enable loading a pre-defined network profile



Metrics Implemented

General Metrics

- Volume Percentage

Counters

- Buffer Count
- Frames per Second
- Seek Counter
- Error Count
- Full Screen Count
- Play Count

Timers

- Session Timer
- Buffer Duration

Extras

- Buffer Size
- Buffer Percentage
- Bandwidth
- Current Time
- Duration
- Watched Percentage
- Bitrate Change Level
- Bitrate Mean
- Bitrate History
- Bitrate Graph







Open-Source Video Players

Ashley Jetty







Open-Source Players

EXOPLAYER

- a media library used for Android
- made by Google
- used in apps like Youtube
- plays hls and dash videos
- Kotlin/Java



SHAKA

- a media library used for browsers
- made by Google
- made for streaming adaptive bitrate video
- plays hls and dash videos
- Javascript



HLS.JS

- a media library that live streams to the browser using a server
- made by Apple
- can natively run in HTML5
- made for streaming adaptive bitrate video
- plays hls videos
- Javascript



VIDEO.JS

- a media library used for browsers
- sponsored by Brightcove
- HTML5 video player
- hls video supported
- dash supported except in iOS Safari
- Javascript





Clappr-Stats

CLAPPR

- a media library used for browsers and android
- HTML5 video player
- hls video supported
- javascript

WHAT IS CLAPPR-STATS?

A Clappr video player plugin that reports playback statuses from the videos played.

WHAT DOES IT DO?

Gives real time stats on key performance indicators of the metrics being recorded and is able to be recorded. Allowed the team to study how metrics are evaluated.

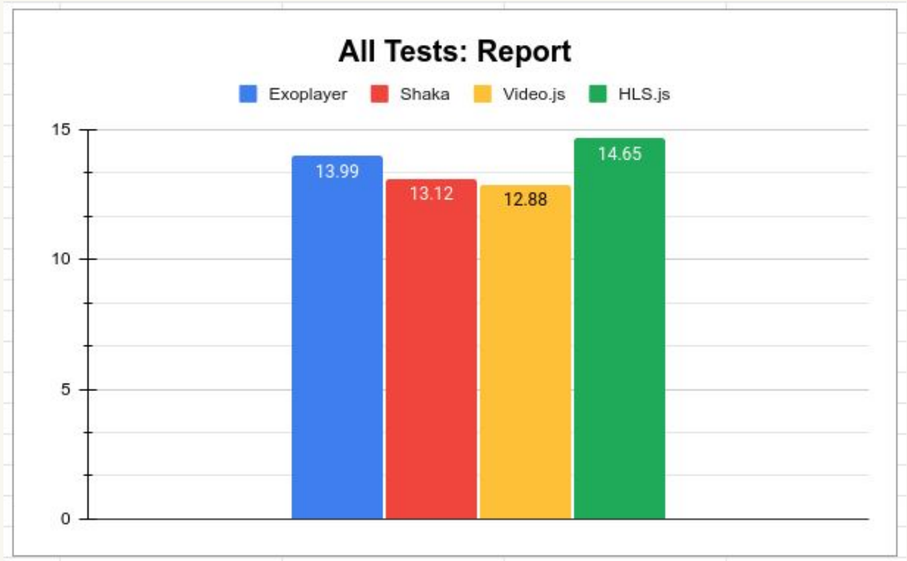
WHAT WE IMPLEMENTED FROM IT?

For each metric that we wished to study and record, we adapted how Clappr-stats implements its own report to work with the chosen video player, hls.js.

Continued work with HLS.js

WHY HLS.JS

- Exoplayer is for Android mobile apps only.
- AT&T is familiar with Shaka and HLS.js.
- HLS had the best performance in testing between all of the video players.
- What development that is done on HLS can be easily adapted to Shaka.
- Future work with Shaka will be done next semester.



VIDEOS 1- 4 REPORT COMBINED (Total Video Tests: 72)

ALL 72 TESTS				
Video Metric/KPI	**Exoplayer**	**Shaka**	**Video.js**	**HLS.js**
Video Quality	2.99	2.9	2.4	3
Video Start Time	2.22	2.18	2.57	2.82
Time-To-Best-Quality	2.78	2.04	2.09	2.83
Rebuffering Count	3	3	2.92	3
Rebuffering Duration	3	3	2.9	3
Score	**13.99**	**13.12**	**12.88**	**14.65**





DASH AND HLS

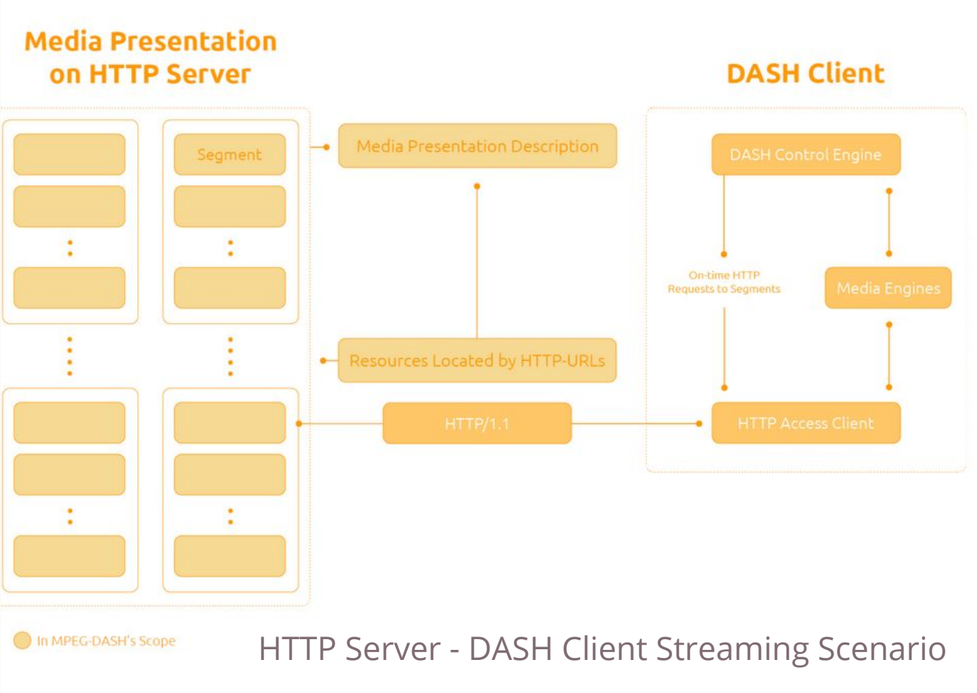
Tim Ellis





MPEG-DASH

(Dynamic Adaptive Streaming over HTTP)



HTTP Server - DASH Client Streaming Scenario

Source: https://www.encoding.com/mpeg-dash/

HISTORY

- 2010: Moving Picture Expert Group (MPEG), started working on DASH
- 2011: DASH becomes international standard
- 2012: First live public trial during London Olympics

HOW IT WORKS

- Manifest containing media segments, metadata, and URLs
- Client requests a segment
- DASH tests network conditions and provides appropriate segment
- Most codecs supported

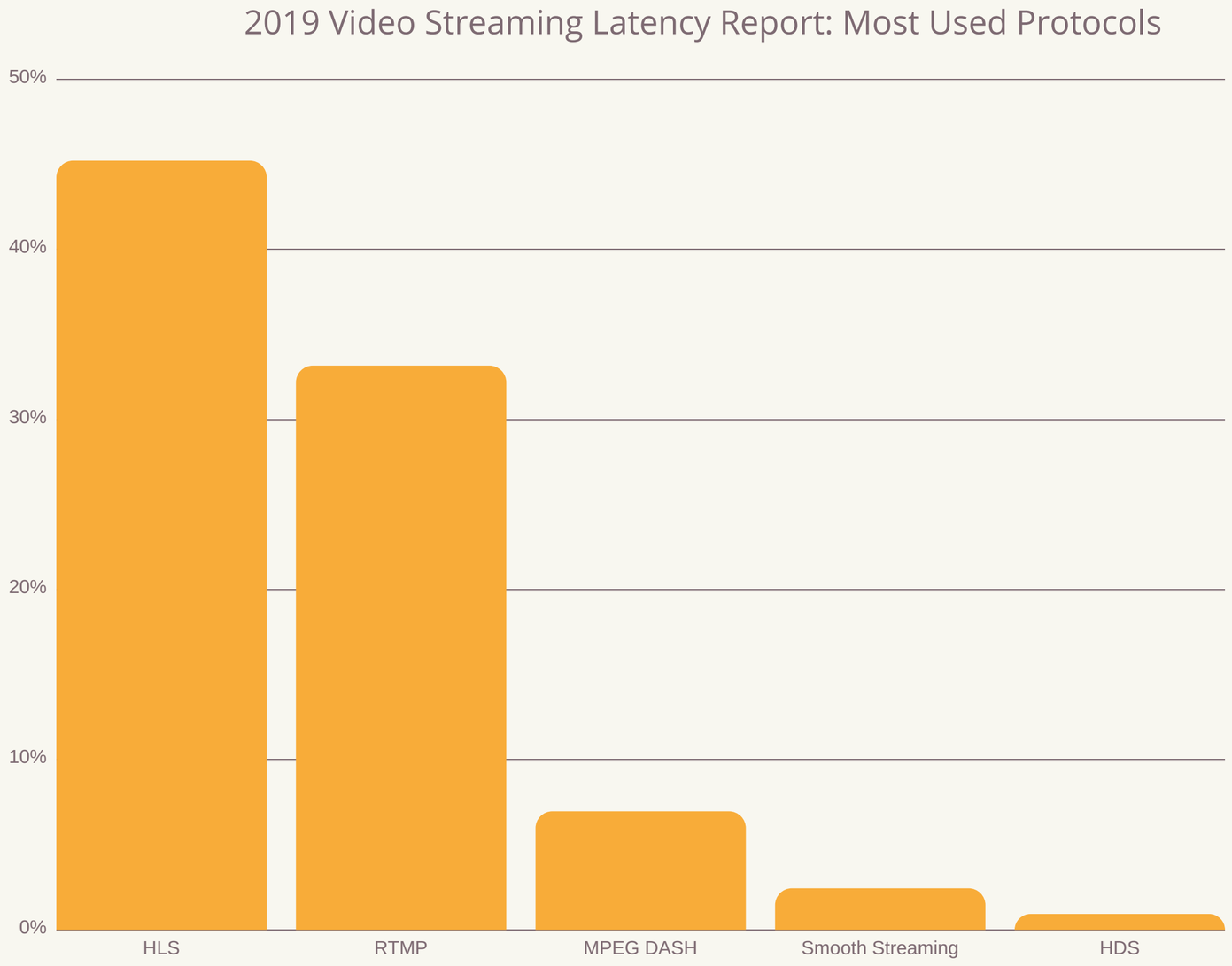
HLS (HTTP Live Streaming)

HISTORY

- 2009: HLS released
- 2011: Android 3.0 supports HLS
- 2015: Windows 10 has native HLS support

HOW IT WORKS

- Manifest containing media segments, metadata, and URLs
- Client requests a segment
- HLS tests network conditions and provides appropriate segment
- Must be encoded with H.264 or H.265 codecs



Source: https://www.wowza.com/blog/2019-video-streaming-latency-report





Adaptive Bitrate Algorithm

David Melendez







What is Adaptive Bitrate?

ADAPTIVE BITRATE

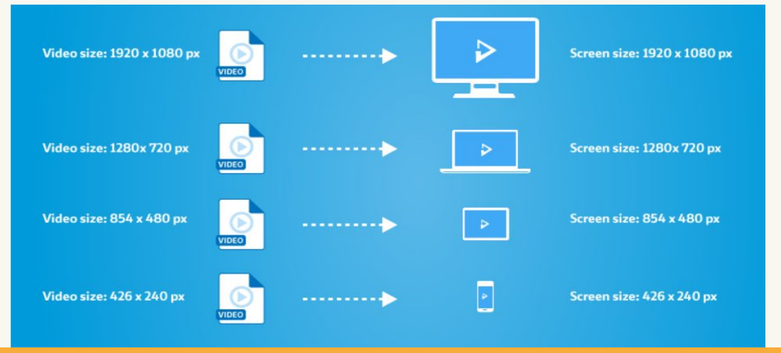
- Bitrate: amount of data transferred in a certain period between the client and the server
- High Bitrate = High Resolution
- Adaptive Bitrate (Adaptive Streaming): player adjusts video quality based on the bitrate of the user
- Used in most video players such as Youtube, Netflix, etc.



Adaptive Bitrate Algorithm

HOW IT WORKS

- Video file is converted into Adaptive Format, video becomes segmented
- Each segment 4 secs long(length can vary)
- Video is played at quality that matches device being streamed to, if internet speed drops video switches over to lower quality to avoid buffering pauses.
- Solves issue of Buffering and Quality
- “One Size Fits All”



Bitrate (kbps)	Resolution
235	320x240
375	384x288
560	512x384
750	512x384
1050	640x480
1750	720x480
2350	1280x720
3000	1280x720
4300	1920x1080





Quantitative Metrics

Jeffrey Luu







Quantitative Metrics

Video Start Time

- A measure of when the user tries to play a video til when it starts to play

Rebuffering

- Count - How many times rebuffering occurs
- Frequency - How often does rebuffering occur
- Duration - How long does a rebuffer take
- Ratio - A percentage of how long the player rebuffered versus the overall playtime



Video Start Time

- Shorter is better

- According to mux.com:
  - Users are more likely to abandon a video if it takes longer than 2 seconds for a player to start playing
  - Each additional second increases the chance a viewer may leave
  - Mobile platform users are more forgiving of longer startup times

Rebuffering

- Users want a seamless experience
- Common causes include:
  - Insufficient Bandwidth
  - Network Errors
  - Player Errors
- Much less common today due to adaptive bitrate streaming









Network Throttling

Rafael Mendoza







Network Throttling

WHAT IS NETWORK THROTTLING?

- Slow down internet connection

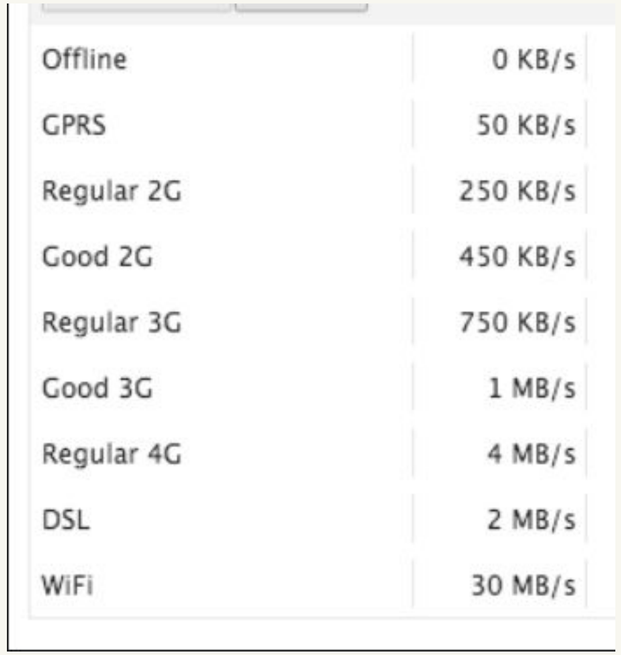
WHAT IS THE PURPOSE?

- To experience the site like a mobile user



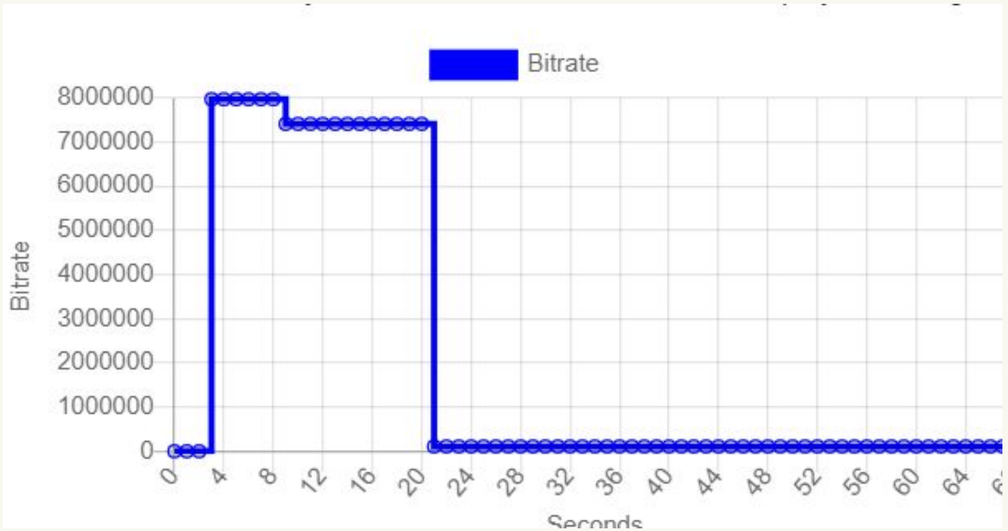
Network Profiles

- Network configurations that represent different mobile data.
- Each network preset assigned with its own download speed.

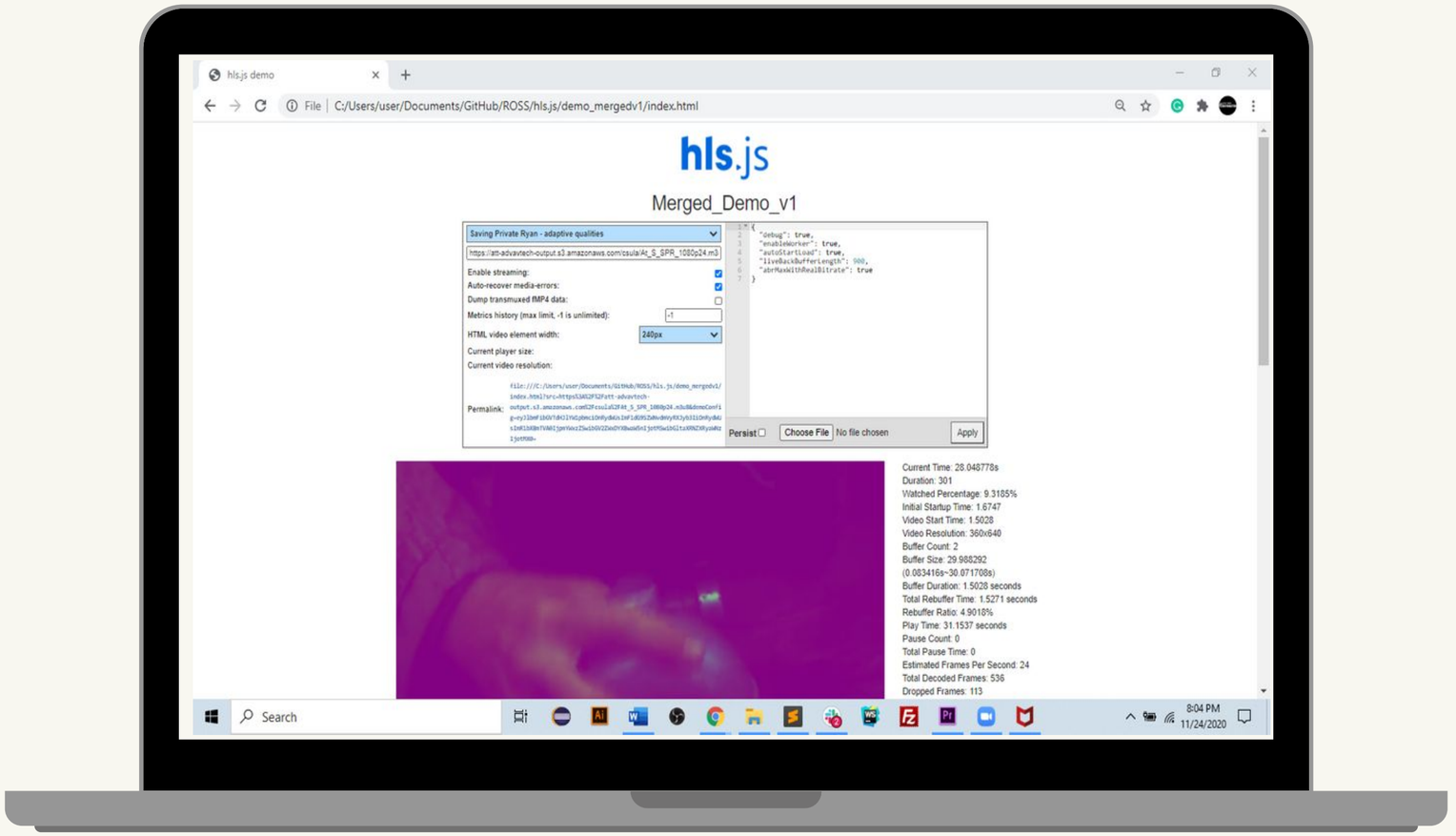


Offline	0 KB/s
GPRS	50 KB/s
Regular 2G	250 KB/s
Good 2G	450 KB/s
Regular 3G	750 KB/s
Good 3G	1 MB/s
Regular 4G	4 MB/s
DSL	2 MB/s
WiFi	30 MB/s

Network Throttling Simulation



- Will not be changing the actual speed of the internet
- Will manipulate the bandwidth value of the video source to simulate network throttling
- Assign the download speed of the selected network preset to the video source bandwidth
- The bitrate of the video src will change
- The bitrate value determines the rate data is transferred



Demo

Israel Lopez-Diaz

Demo:

- How do you select a video?
- Where are the Real Time Metrics?
- How do you begin the throttling simulation?
- How do you record and download the recorded data?
- What is inside the Downloaded XLSX file?




hls.js





Future Goals

Wendy Joya





Accomplished



RESEARCH

- Video players
- Metrics
- Different Libraries
- Network Throttling



ANALYSIS

- Video Player performance
- Metrics
- Network Throttling



IMPLEMENTATION

- New Metrics
- Created a HLS.js video player with Network Throttling

Future Goals:

TASKS:

- Obtain Real-Time Values
  - KPI metrics
  - Network Throttling
- Graphs illustrating Real-Time values
- Video Quality

COMPLETE:

- Finalize HLS.js player
- Complete our Shaka player; including network throttling real-time.
- Front-end and back-end interface improvements

HOSTING:

- Hosting of both our HLS.js and Shaka websites.



Project Timeline



JANUARY 2021

- Completion of our HLS.js player.
- Start working on Shaka
- HLS.js Performance Analysis

MARCH 2021

- Completion of Shaka,js
- Host our Websites
- Shaka Performance Analysis
- Comparison (HLS.js vs. Shaka) Performance Analysis

APRIL 2021

- Project Poster
- Project Presentation
- Documentation (Final Version)

Questions

CITATIONS - VIDEO PLAYERS & LIBRARIES

ExoPlayer: https://github.com/rc728m/ExoPlayer.git

Exoplayer Demo: https://exoplayer.dev/demo-application.html

Shaka: https://github.com/rc728m/shaka-player.git

HLS.js: https://github.com/video-dev/hls.js/tree/master

HLS Demo: https://hls-js-dev.netlify.app/demo

Video.js: https://github.com/rc728m/video.js.git

Clappr Stats: https://github.com/clappr/clappr-stats

Mux.js: https://github.com/videojs/mux.js#readme

Chart.js: chartjs.org

MyExcel: https://github.com/jsegarra1971/MyExcel