Rutgers, The State University of New Jersey

Department of Computer Science

CS553: Design of Internet Services

"MovieSync"

Prof. Srinivas Narayana

Team Members: -

Desmond Lobo (dl1053)

Mansi Rajesh Borole (mb2208)

Index

Introduction....2

Existing Works....2

Our Contribution....2

Design Overview....3

Implementation Details....4

Experimental Results and Performance Metrics....6

Conclusion....10

Comments and Future works....10

References....10

Introduction

This project introduces a way to watch videos in a synchronous way. We have created software that allows two or more peers to synchronize their video playback experience. Video streaming is also optimized by using the DASH (Dynamic Adaptive Streaming over HTTP) multimedia format to ensure that all viewers in the watch party have a smooth and uninterrupted viewing experience.

Project GitHub Link - https://github.com/desmondlb/MovieSync.git

Existing Works

There are several video synchronization systems that exist today. Netflix Teleparty, formerly known as Netflix Party, is a widely used implementation. It allows users to watch movies and TV shows together on Netflix with friends and family remotely. However, it only works on specific streaming platforms like Netflix, Disney Plus, and Hulu. Local-party and WatchParty are two open-source alternatives to Teleparty. However, they both have limitations, such as needing every client to have the same video files.

Our Contribution

We have developed MovieSync, a software that allows peers to synchronize their video playback experience with custom video files from a server where the mp4 file exists. This means that users can watch any video file they want, and not just content from specific streaming platforms. Additionally, we have done a comparative analysis of different implementations such as DASH vs Non DASH, CDN vs Non CDN.

DASH is a video streaming technology that adjusts the quality of a video stream according to the user's available bandwidth. Non DASH, on the other hand, is a traditional way of streaming where the video file is downloaded entirely before playback. Our analysis will help determine which method is better for synchronizing video playback.

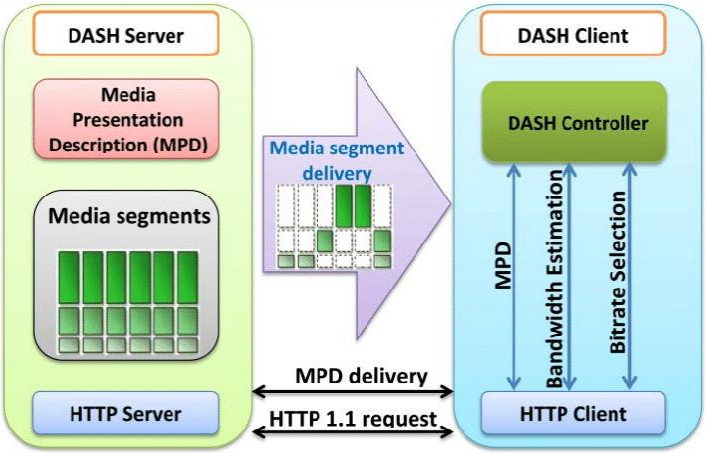


Figure 1: The overview of Dynamic Adaptive Streaming over HTTP technique

CDN (Content Delivery Network) is a system of distributed servers that deliver web content to users based on their geographic location. CDN is used to improve website performance, reduce latency, and provide a better user experience. Our analysis will determine if using CDN provides better synchronization compared to not using it.

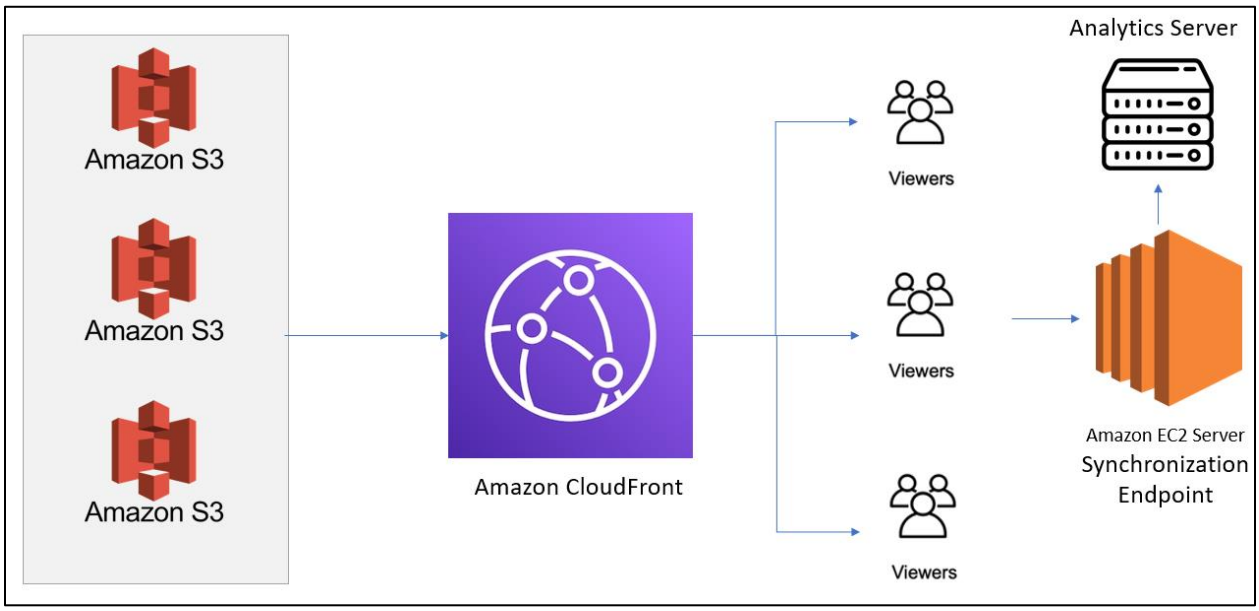


Figure 2 : MovieSync Architecture

MovieSync aims to provide a flexible and robust video synchronization system that can work with any video file and provide a seamless playback experience for users located anywhere in the world.

Design Overview

MovieSync is a video synchronization system that allows two or more peers to watch the same video file simultaneously, regardless of their location. The system consists of a React client, a server, and an analysis server. The React client is responsible for setting up the room, providing the room name, room code, and username. The server takes care of broadcasting **play** and **pause** actions to all other clients. The analysis server logs all actions performed by users and timestamps in UTC to analyze latency, BitRate and BufferRate. All communication for synchronization happens through sockets, whereas there are separate APIs for creating and deleting a room. The room information is stored on MongoDB. Clients and the server are hosted on Amazon EC2, and the video files are stored on S3.

Architecture:

React Client:

The React client creates a room and provides a synchronized video player for users. Users enter the room name, code, and username to access the player. The player uses React, Socket.IO, and HTML5 video to synchronize playback.

Server:

The server is responsible for broadcasting play and pause actions to all other clients in the room. When a user clicks the play or pause button, the server receives the action and broadcasts it to all other clients in the room. The server uses Node.js, Socket.IO, and MongoDB to synchronize video playback.

Analysis Server:

The analysis server logs all actions performed by users and timestamps in UTC to analyze latency. When a user performs an action, the server logs the action and the timestamp. This data can then be analyzed to determine if there is any latency in the video synchronization.

Communication:

All communication for synchronization happens through sockets. When a user performs an action, the action is sent to the server through a socket. The server then broadcasts the action to all other clients in the room through sockets. Socket.IO is used to handle real-time communication between clients and the server.

APIs:

There are separate APIs for creating and deleting a room. When a user creates a room, the room information is stored on MongoDB. When a user deletes a room, the room information is removed from MongoDB. Express.js is used to handle API requests.

Storage:

Clients and the server are hosted on Amazon EC2. The video files are stored on AWS S3. It is highly scalable and durable and is designed to store and retrieve any amount of data from anywhere on the web.

Implementation Details

Dynamic Adaptive Streaming over HTTP (DASH) is a streaming protocol used to deliver video content over the internet. It is an adaptive bitrate streaming technique that dynamically adjusts the video quality based on the viewer's internet speed, device capabilities, and other factors.

In DASH, the video file is broken down into small segments and each segment is encoded at multiple bitrates. The video player then selects the appropriate bitrate and quality for each segment based on the viewer's internet speed and device capabilities. This ensures that the viewer always gets the best possible quality for their current conditions.

DASH uses HTTP to stream video segments to the viewer's device. This makes it compatible with most web browsers and devices. It also allows for easy content delivery over content delivery networks (CDNs), which can improve video playback performance by delivering video segments from the nearest server to the viewer.

One of the main advantages of DASH is its adaptability to varying network conditions. It can dynamically adjust video quality in real-time based on available bandwidth and other factors, which

reduces buffering and ensures a smooth playback experience. DASH also supports multiple audio and subtitle tracks, which allows for more flexibility in content delivery.

However, one disadvantage of DASH is that it requires more processing power and network bandwidth than other streaming protocols like HTTP Live Streaming (HLS) or MPEG-DASH. This can result in higher server costs and lower video quality for viewers with slower internet connections.

DASH is a powerful streaming protocol that provides a high-quality, adaptive video playback experience for viewers. Its flexibility and compatibility make it a popular choice for video content providers.

The Media Presentation Description (MPD) is an XML file that contains metadata required by a DASH Client to construct appropriate HTTP-URLs to access Segments and to provide the streaming service to the user.

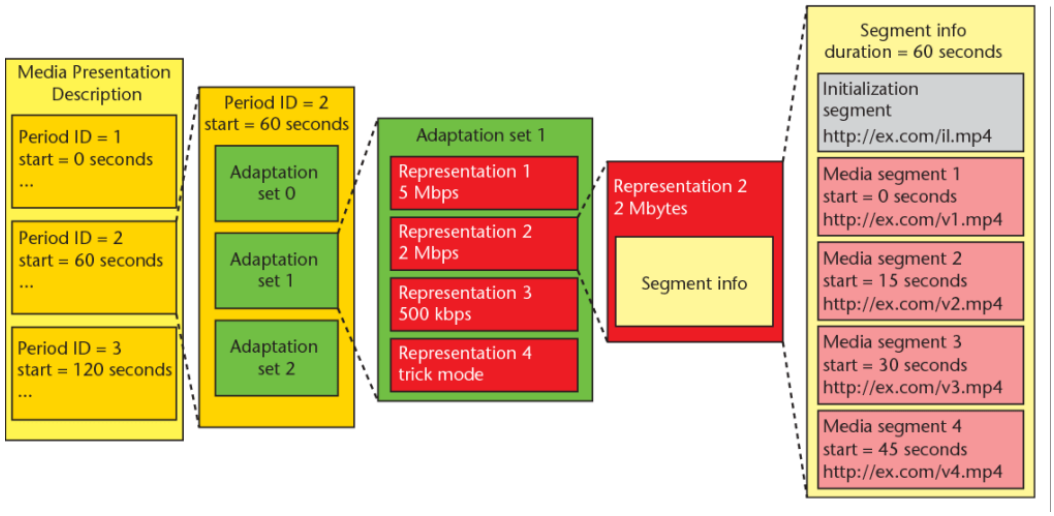


Figure 3 : Structure of MPD Manifest file

The MPD file structure consists of:

1. **Periods**: describe a part of the content with a start time and duration,
2. **Adaptation Sets:** There are used to group video and audio segments with similar characteristics (such as bitrate and resolution).
3. **Representation Sets:** Each representation element that specifies the segment information, such as the segment duration, bitrate, and URL. They also contain multiple versions of a stream, each encoded differently along with SubRepresentations which are subsets of representation information about a particular stream,
4. **Media Segments** which are media files that are played by the client, and
5. **Index Segments** which contain information about media segment durations and stream access point types.

Experimental Results and Performance Metrics

We performed various experiments to measure the performance of our system. This included the measurement of metrics such as the video synchronization latency, Bit-Rate and Buffer-Rate. We have orchestrated the tests using **Selenium** to mimic user behaviors of joining rooms and client actions such as playing, pausing etc. However due to the limitations of our screens and laptops we were only able to create rooms with max ~20 clients.

Metric1: Video Synchronization latency Summary

Observations:

From the below tables we can clearly see that when we use CDN for streaming, along with the MPEG-DASH protocol performs is the best. This is because CDN caches the content to be delivered near the clients. Therefore, the clients do not have to wait long for the video to start playing and can signal the playback action to the server quickly.

We Also observe a trend that as the number of clients connected to a Room increases, the latency increases. A large number of clients increase the load on the server, as the server must maintain many sockets and manage client-client communication effectively.

Number of clients	Average Latency
2	20.191
21	172.88

Table 1: Streaming MP4 from AWS S3

Number of clients	Average Latency
2	21.48
6	46.7
11	95.99
15	199.876

Table 2: Streaming MP4 from S3 connected to CDN (AWS CloudFront

Number of clients	Average Latency
2	20.115
7	43.4
10	37.907
19	109.6944

Table 3: Streaming DASH from S3 connected to CDN (AWS CloudFront)

Metric 2: MP4 BitRate

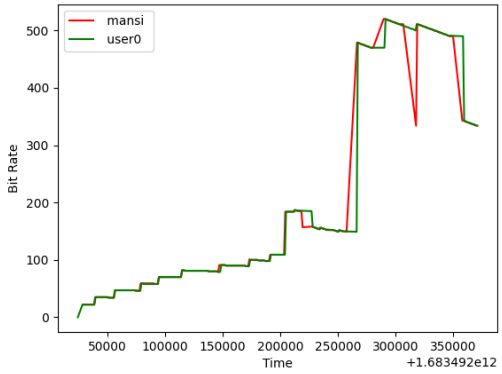
The instantaneous bitrate is the amount of data transmitted per unit of time at any given moment. It can be calculated by measuring the amount of data transmitted during a certain time interval, such as one second, and then dividing it by the duration of that interval.

We observe that the Bitrate increases when we perform experiments like play, pause and most importantly seeking to a particular time stamp either backwards or forwards.

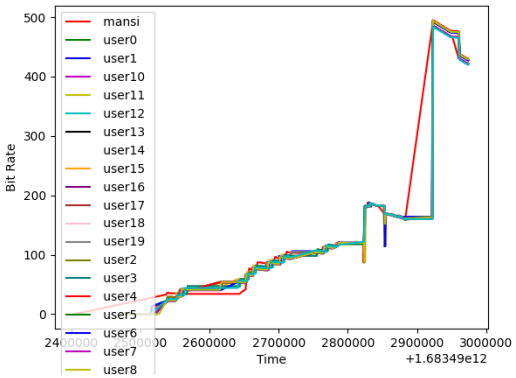
Especially, when we seek backwards in the video, the player may need to download additional data to fill the buffer and maintain smooth playback. This can result in a temporary increase in the bitrate.

Sudden spike in Bitrate represents events - when we seek either forward of backward in the video!

MP4 Streaming from S3:

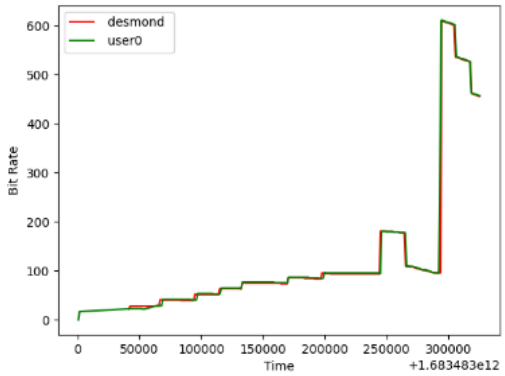


MP4 BitRate (2 Clients)

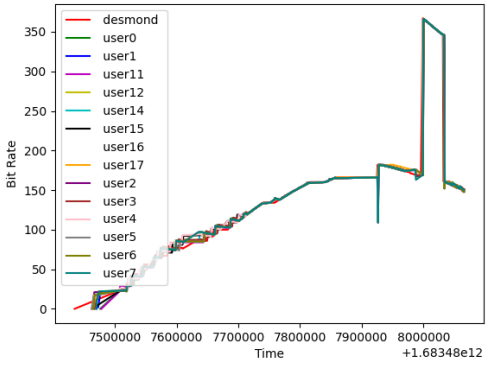


MP4 BitRate (21 clients)

MP4 Streaming from CDN:

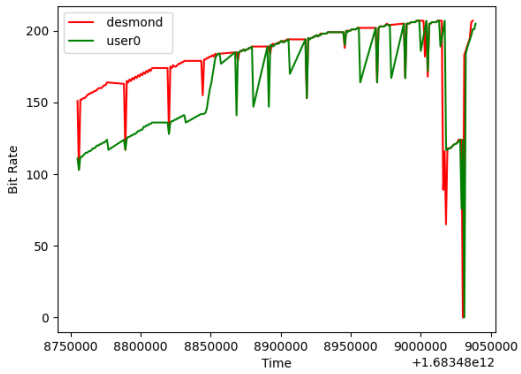


CDN BitRate (2 clients)

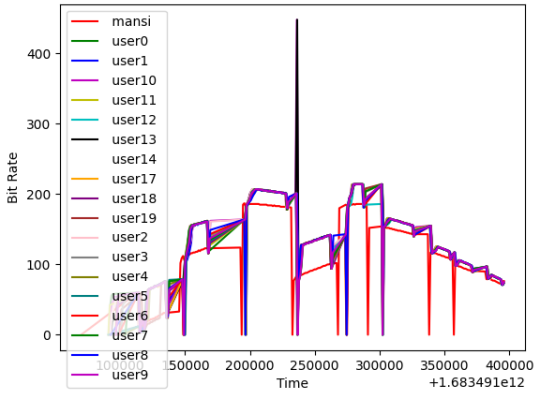


CDN BitRate (19 clients)

DASH Streaming using CDN :



DASH-CDN BitRate (2 clients)



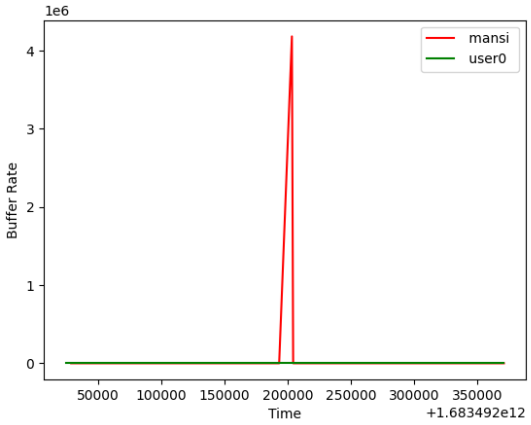
DASH-CDN BitRate (19 clients)

Metric 3: Buffering Rate

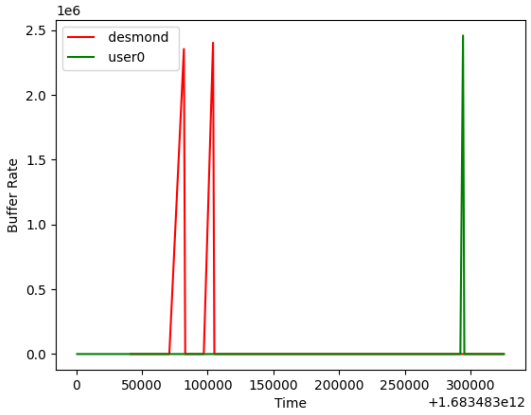
Buffering rate in video streaming is the speed at which the video player is able to download and buffer video data from the streaming server to the user's device. A high buffering rate means that the video player is able to download video data quickly and maintain a consistent playback without any buffering or interruption. On the other hand, a low buffering rate means that the video player is not able to download video data quickly enough to maintain smooth playback, resulting in buffering or interruption during playback.

The sudden spike in the graphs below represents the events like seeking to a particular time stamp. Especially, if we seek-to a timestamp that is not in the buffer, we see a higher buffering rate. We see that DASH performs better as compared to mp4 **because DASH supports multiple bitrates and resolutions for a single video**, which allows for a smoother transition between different quality levels depending on the available bandwidth. This helps to ensure a more consistent and seamless playback experience. As seen below: DASH bfeer rates are in the rage (0.00 – 0.002)kbps while the mp4 buffering rate vary from 0 to 2.5 kbps

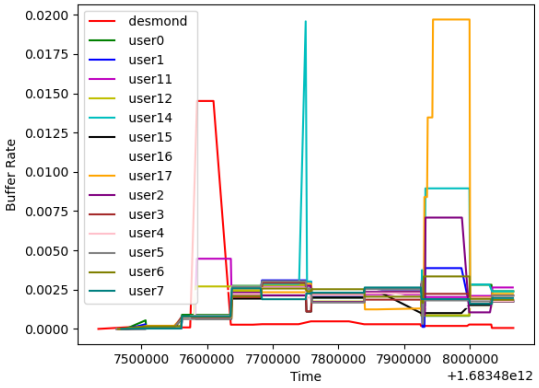
Streaming MP4 file from Amazon S3



MP4 Streaming from CDN:

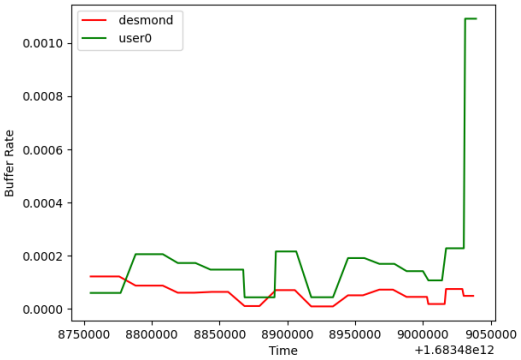


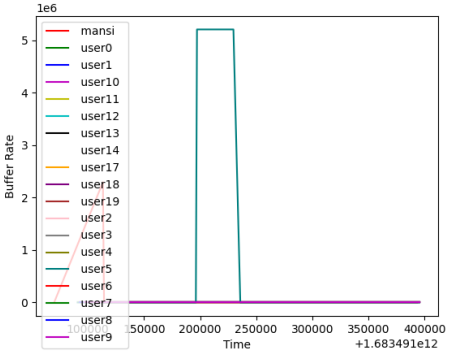
Buffer Rate (2 Clients)



CDN Buffer Rate (15 clients)

Streaming DASH file from CDN





DASH-CDN Buffer Rate (2 clients) DASH-CDN Buffer Rate (19 clients)

Conclusion

After evaluating our system on a combination of streaming methods we can conclusively say that the Movie Synchronization platform works best when it uses MPEG-DASH protocol to stream data using a CDN. The system also shows the lowest latency when the number of clients is restricted to around 2-7.

Comments and Future works

- We tried to test with as many clients as possible but we were limited by our compute resources.
- The AWS free tier services have a memory and compute limit which cannot handle a huge number of clients and rooms together.
- In the future we intend to implent an Upload feature and a utility that will convert mp4 file to MPD and its file fragments using FFMPEG tool and GPAC.

References

Figure 1: Huy, Vu & Mashal, Ibrahim & Chung, Tein yaw. (2017). A novel bandwidth estimation method based on MACD for DASH. 11. 1441-1461. 10.3837/tiis.2017.03.011.

Thomas Stockhammer(Qualcomm Incorporated) Dynamic Adaptive Streaming over HTTP – Design Principles and Standards Paper (w3.org)

What is MPEG-DASH Video Streaming Protocol? How Does MPEG-DASH Work?