A Project Report

on

Lively: Low Latency Live Streaming

carried out as part of the course (CS1532) Submitted by

Prerit Pathak

169105131

5th Semester, Btech, Computer Science

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

MANIPAL UNIVERSITY
JAIPUR

Department of Computer Science & Engineering,
School of Computing and IT,
Manipal University Jaipur,
November, 2018

DECLARATION

I hereby declare that the project entitled “**Lively: Low Latency Live Streaming**” submitted as part of the partial course requirements for the course ___, for the award of the degree of Bachelor of Technology in Computer & Communication Engineering at Manipal University Jaipur during the _*5th*_ semester, has been carried out by me. I declare that the project has not formed the basis for the award of any degree, associate ship, fellowship or any other similar titles elsewhere.

Further, I declare that I will not share, re-submit or publish the code, idea, framework and/or any publication that may arise out of this work for academic or profit purposes without obtaining the prior written consent of the Course Faculty Mentor and Course Instructor.

Signature of the Student:

Place:

Date:

Abstract

In a world where interactions are primarily focused on virtualization and life has inclined towards the audio and video streaming there is hardly anyone who has not yet entered this phase. Be it anywhere in the world, people are connected to each other throughout their day with their loved ones. Unlike other video on demand streaming services or like video broadcasts, real time interaction has become crucial for both streamers and their audiences. Today popular livestream captures thousands of users and covers everything like press conferences, political protests stage concerts, TV shows and sporting events. To guarantee a no lag video streaming one must make sure that the latency is low and transmission rate is least possible. This is a significant challenge in today’s world for streamers.

The complete project is low latency live streaming project with client latency of less than 7 seconds and at the same time media server is also capable of publishing the same video to multiple accounts on Facebook and YouTube. Live streaming is an interesting way to engage audience’s worldwide in an event which is happening at a certain place, it’s an online streaming media which simultaneously records and broadcasts in real time to the viewer. In streaming video and audio, the traveling information is a stream of data from a server. The decoder is a stand-alone player or a plugin that works as a part of the Web browser. The server, information stream and decoder work together to let people watch live or prerecorded broadcasts. The program playing the streaming file discards the data as we watch. A full copy of the file never exists on the user’s computer so that he could save it for later. This is different from progressive downloads which download part of afile to our computer then allows us to view the rest as the download finishes.

As it looks so much like streaming media, progressive downloading is known as Pseudo Streaming. From a client- side perspective, it is web-based and mobile-friendly. It is intended to be useful for both research and practical purposes.

Media server is capable of transcoding and we only need to push the highest bit rate stream and the media server is capable to transcode it in multiple bitrates on the fly. Cue points (markers which tells the video to take some action) are also a part of this live stream project which helps the user to monetize the video. We are using cloud to store the m3u8 and ts files which was transferred from our media server to the cloud by s3 SDK or AWS client (in our case) as we are using S3 bucket.

TABLE OF CONTENTS

1. Introduction

1.1 Product Scenarios

1.2 Project Scope

2. Requirement Analysis (SRS)

2.1 Functional Requirements (Including test cases using white box/Black box)

2.2 Non-Functional Requirements

2.3 Use-Case Scenarios

2.4 Other Software Engineering Methodologies

3. System Design

3.1 Design Goals

3.2 System Architecture

3.3 Detailed Design Methodologies (as applicable)

4. Work Done

4.1 Development Environment

4.2 Results and Discussion

5. Conclusion

6. References

Introduction

Scope of the Work

This section describes the business environment in which the product will be used.

Current Situation

In today's socially active world, where people want to share the important moments of their lives as they happen, live streaming plays a really important role. There are various apps and websites available in the market that allow their users to use their services to stream live content.

Motivation

Even though a lot of companies provide live streaming service, the catch here is that one can only livestream to only one service at a time. This can be really stressful because everyone has a different audience to cater to on different services. After carefully studying this problem and understanding the shortcomings of the current scenario of social live streaming, the idea for Lively was conceptualized.

Product Scenarios

Webinars:

Chris is a world famous MBA coach. He tutors interested people online, in fields of finance, sales and marketing. He generally uses the Twitch service to conduct these live webinars. He records this webinar and posts it later on his Facebook group. The people in his Facebook group who don’t use Twitch miss the opportunity to converse with him live. Chris can use Lively to stream his content live to Facebook, Twitch, Youtube and other services at once to save a lot of time, cost and effort.

Music Events:

Jai is a college going guy. He enjoys going to musical concerts. As a normal kid, he is active on Facebook, Youtube etc. But his audience on both apps are different. Now he faces a dilemma: which service should he use to stream the concert live? He can use Lively to stream the concert live at many services with just a click of a button!

Requirement Specification

Introduction

Purpose

The purpose of this document is to present a detailed description of Lively. It will explain the purpose and features of the software, the interfaces of the software, what the software will do and the constraints under which it must operate. This document is intended for users of the software and also for potential developers.

Document Conventions

This document was created based on the IEEE template for System Requirement Specification documents.

Intended Audience and Reading Suggestions

This document is intended for both, the developers and the end users.

Project Scope

Lively is a web based tool which helps users to stream content to multiple services at the same time. The crux of this project is the implementation of Adaptive Bitrate Streaming (HLS) and a comparison & contrast between TCP (Transfer Control Protocol) and UDP (User Datagram Protocol) protocols for live streaming as well as video-on-demand (VOD). Invasion of someone’s privacy is not promoted in any way.

System Design

Design Goals

Since Lively is majorly a service which relies on the backend, liberty has been taken while designing the website. The website has been designed using plain Vanilla Js, with minimal CSS styling and a basic HTML form to take RTMP link inputs from the user.

System Design and Architecture

Architectural Styles

Lively is based on a Service Oriented Architecture (SOA) model. Which is a style of software design where services are provided to the other components by application components, through a communication portal over a network.

Streaming Protocols

Choosing a streaming protocol is a difficult task that depends on the type of information to be shared. Communication must be made using a protocol formed by a group of rules defining how data are transmitted over the network and divided into different parts, such as headers, data, authentication and error handling. Thus, a streaming protocol may be viewed as a communication protocol where the transmitted data are media data.

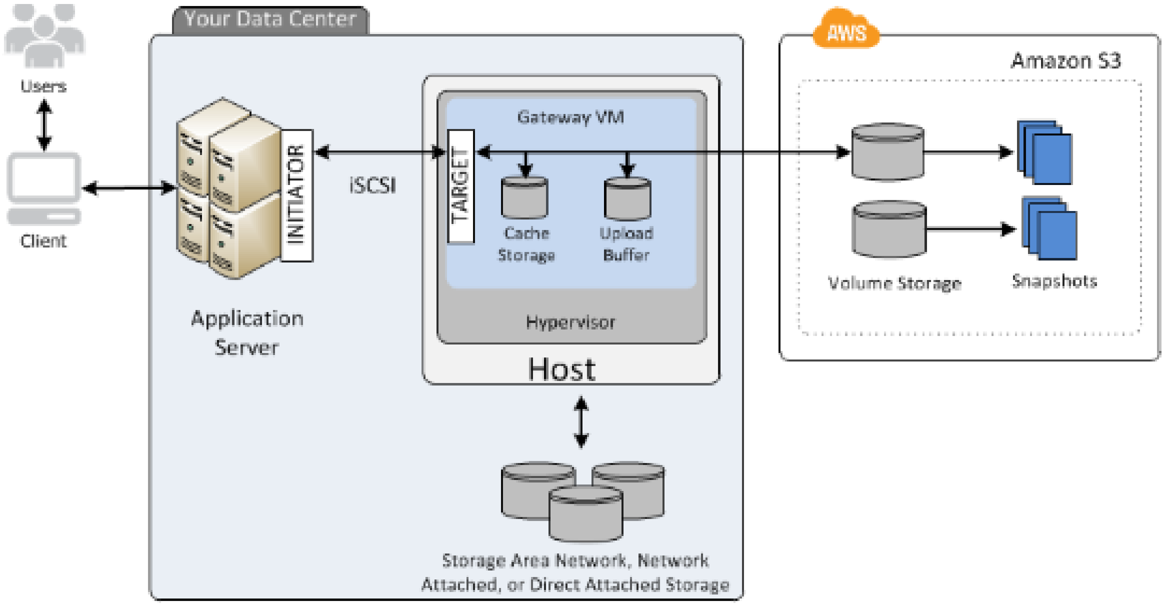
In this study, the main objective is to share audio and video media. For this reason, the most important point is the guarantee of a low latency and efficient transmissions with occasional packet losses. A media streaming protocol is defined, considering the structure of the packets and the algorithms used to send real-time media on a network. Different media streaming protocols are available today, which differ in a few implementation details. Traditionally, these communication categories have been divided into push-based and pull-based protocols. Push-based protocols consist of established communication between the client and the server, where the client is responsible for establishing the connection, and the server sends a packet stream until the client stops or interrupts the communication. In this type of protocol, the server, in addition to sending media packets, maintains a session to listen for commands from the client. These protocols usually work over the User Datagram Protocol (UDP) but could work over the Transmission Control Protocol (TCP) if necessary. They normally use the Real-time Transport Protocol (RTP)

Pull-based protocols are based on the HTTP protocol and consist of a communication between client and server where the client is responsible for sending a request to the server, and the server starts a communication where the client downloads the video streaming. In these protocols, the streaming speed depends on the bandwidth of the client network. The most commonly used pull-based protocol is a progressive download, in which a client sends a request to the server and starts pulling the media content from it as quickly as possible

CLOUD S3 STORAGE

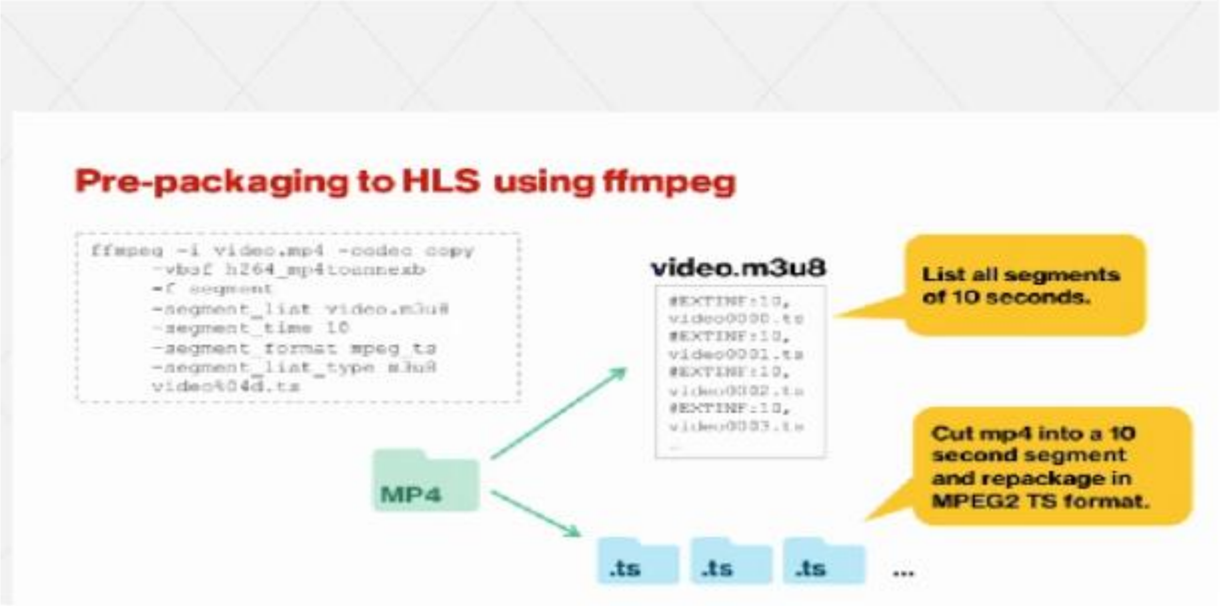
Amazon Simple Storage Service (Amazon S3) is a scalable, high-speed, low-cost, web-based cloud storage service designed for online backup and archiving of data and application programs. S3 was designed with a minimal feature set and created to make web-scale computing easier for developers. The S3 cloud storage service gives a subscriber access to the same systems that Amazon uses to run its own websites. S3 enables a customer to upload, store and download practically any file or object. Amazon S3 is an object storage service, which differs from block and file cloud storage. Each object is stored as a file with its metadata included and given an ID number. Applications use this ID number to access an object.

Unlike file and block cloud storage, a developer can access an object via a rest API. Amazon S3 comes in two storage classes: S3 Standard and S3 Infrequent Access. S3 Standard is suitable for frequently accessed data that needs to be delivered with low latency and high throughput. S3 Standard targets applications, dynamic websites, content distribution and big data workloads. S3 Infrequent Access offers a lower storage price for backups and long-term data storage.



MPEG-DASH

The MPEG-DASH standard defines the media presentation description (MPD) as well as segment formats and deliberately excludes the specification of the client behavior, i.e., the implementation of the adaptation logic, which determines the scheduling of the segment requests, is left open for competition. In the past, various implementations of the adaptation logic have been proposed both within the research community and industry deployments/products.



Pre-packaging to HLS using ffmpeg

```
ffmpeg -i video.mp4 -codec copy
       -vbsf h264_mp4toannexb
       -f segment
       -segment_list video.m3u8
       -segment_time 10
       -segment_format mpeg_ts
       -segment_list_type m3u8
       video%04d.ts
```

video.m3u8

```
#EXTINF:10,
video0000.ts
#EXTINF:10,
video0001.ts
#EXTINF:10,
video0002.ts
#EXTINF:10,
video0003.ts
...
```

List all segments of 10 seconds.

Cut mp4 into a 10 second segment and repackage in MPEG2 TS format.

NABILIVE – Mobile App / OBS software for desktops

A video live streaming platform that allows its customers to broadcast live video content online, and viewers to play the content via an internet connected device. It is a live video streaming IOS application that helps the user to broadcast anything live and posts it on the Twitter, Facebook and YouTube, it uses the RTMP compatible streaming protocol that helps a person to be social in no time. It offers unlimited streaming lifetime access It also supports the HD stream quality up-to 1080p. One must get the stream name/key and put key into app then you can Go Live.

Live Streaming from Your Camera
Real Time Streaming to PC and Smartphone
Support to the Camera Rotation
Registration not Required
554
title.sdp
RTSP push to server disconnected
Server Address

CONCLUSION

After Implementing the project on AWS servers, we can get a client latency of 6-7 seconds including a 2-3 second client buffer. Best results show us latency less than 5 seconds. We also able to publish the live streams to Facebook and YouTube on our account and make the things work.

REFERENCES

[1] L. D. C. a. S. Mascolo, “An experimental investigation of the Akamai Adaptive Video Streaming”.

[2] A. F. Lippens, “A Review Of Http Live Streaming,” 2010.

[3] A. R.-G. ,. J. M.-G. a. P. C.-G. Iván Santos-González, “Implementation and Analysis of Real-Time Streaming Protocols”, 2017.

[4] X. Z. Z. a. B. Y. Bolun Wang, “Anatomy Of a Personalized Livestreaming System”.

[5] D. W. M. S. R. G. C. M. S. L. Christian Timmerer, “LIVE TRANSCODING AND STREAMING-AS-A-SERVICE WITH MPEG-DASH”.

[6] L. R.-G. J. GARCÍA-ZUBIA, “An Open and Scalable Web based interactive live streaming architecture”, 2017.