*International Journal for Modern Trends in Science and Technology*, 7(12): 92-95, 2021

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST0712016
Available online at: http://www.ijmtst.com/vol7issue12.html

# Real-Time Streaming Application

**Pushkar Dureja | Naman Samra | Nipun Gupta| Dr. Farzil Kidwai**

Department of Computer Science and Engineering, Maharaja Agrasen Institute of Technology, Rohini, Delhi

**To Cite this Article**
Pushkar Dureja, Naman Samra, Nipun Gupta and Dr. Farzil Kidwai. Real-Time Streaming Application. *International Journal for Modern Trends in Science and Technology* 2021, *7* pp. 92-95. https://doi.org/10.46501/IJMTST0712016

**Article Info**
Received: 28 October 2021; Accepted: 03 December 2021; Published: 05 December 2021

## ABSTRACT

This research paper focuses on analysis of different streaming protocol majorly including HTTP Live Streaming (HLS), Dynamic Adaptive Streaming over HTTP (DASH). This research will explain broadly about these two protocols and compare them on the basis of various factors: latency time and compatibility. As a case study, Modern gaming is an activity that requires multimedia communication including video and audio, then this requirement can be solved with the development of media streaming based software using the RTMP or HLS protocols. In the end it can be concluded about the feasibility of using HLS protocol to develop the streaming application for this case

**KEYWORDS:**Video Streaming, HTTP Live Streaming, Real Time Messaging Protocol, JavaScript, Node.js.

## INTRODUCTION

Streaming is the new way to watch video content. It offers a high-quality streaming experience and lets you access your favourite shows whenever and wherever you want, with minimal buffering and low costs. While there are a lot of streaming protocols, they all have one thing in common: they help stream media between different devices such as phones, tablets or computers to share it with others.

Streaming protocols are the main components for streaming media. There are two types of protocols: HTTP and RTSP. HTTP is a stateless protocol, which means there is no session or keep alive mechanism in it. This also means that all responses get returned to the client as a whole packet which may not be optimal if you have many clients connected to your server at once. The major advantage of RTSP is its ability to return only partial or individual responses (elements).

Streaming protocols are generally used in the media industry to transmit audio and video over a network. Today, they have become an important part of the Internet and internet protocol (IP) networks. Streaming protocols perform efficient compression of multimedia data using lossy or lossless compression methods.

Http Live Streaming is the protocol for transmitting information, such as audio and video, over HTTP. It's designed to allow streaming content to be delivered not just reliably but also efficiently using server-side prediction of varying bandwidth constraints while maintaining continuous playback.

## OBJECTIVES

Themainobjectivesofourproject"Real-Time Streaming Application"are:

Exploring the potential gains on live streaming by developing a platform independent software that that can reach devices running different operating system and browser.

- This software is capable of live streaming both high

resolution video (HD) and still images on to popular desktop platforms including Windows, Mac OS X, Android, Linux. This will allow the accessibility and reach of live streaming rights to unserved populations in the world using mobile phones.

- A platform that can stream both video and audio data is used if it offers equal quality on all supported platforms with no pre-installation instructions; this removes fundamental roadblocks to scaling up system architecture development.
- To allow the seamless delivery of multiple media types on one server without requiring any client side software or plugins.This also allows itto scale across multiple servers, platforms, and environments.

## TECHNOLOGY USED

- Nginx: Nginx RTMP module, which is a wrapper for the popular and widely used Real Time Messaging Protocol (RTMP). The module provides support for RTMP streaming on a number of platforms and has been designed to simplify development.

- JavaScript: JS is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js

- NodeJS: NodeJS is a JavaScript runtime built on Chrome's V8 engine. It provides event-driven I/O to the web by abstracting away the complexity of network communications and storage. It also makes it easy to write servers that are not only fast, but also highly scalable and energy efficient. In short, Nodejs lets you build server-side applications with JavaScript, making them lightweight and highly performant

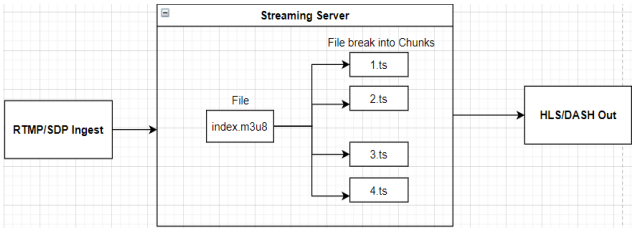
## IMPLEMENTATION

Implementation of the project has been done in two following

- Developing a user interface using Reacts, Redux, Firebase and NodeJS to analyze the performance of different streaming protocols.
- Delivering the video streams through various protocols including HTTP Live Streaming, Dynamic Adaptive Streaming over HTTP using *node-media-server* and Custom *nginx-rtmp-server* etc.
- The basic principle of live streaming with HLS is: take a scene (for example, an advert), then split it in multiple chunks that are delivered to users sequentially.
- The goal is to reduce the total bit rate as much as possible while maintaining a good quality and good video appearance on all devices both on mobile phones and desktop computers.
- In cases where you want your viewers to enjoy smooth playback with minimal buffering, you will frequently see footage shown at half resolution or quarter resolution. While doing so saves bandwidth, this technique can mean a loss of quality.
- Minimizing the active bit rate produces a better result for each typical viewer. Depending on the average viewing size, this could mean that you'll cover your usual audience with less than half as much data transmitted from live point to end-user location; or it may not be noticeable at all – because in many cases no encoding is needed at all to display smooth video and return an acceptable audio signal.
- The quality and speed results of selected HLS streams differ on different devices, depending on the capabilities. On large screen TVs for example a 4K HDR broadcast can be delivered over 10Mbps with 1000-2000 interlaced frames per second in HEVC 8bit + AAC audio at 44100Hz [@ 100kbps; resulting live bit rate: 50816 kbps]. And yet this file produced by a broadcaster at 50kbps would play on an older 2K TV with a different resolution, and the same audio in 32bit @ 44100Hz [@ 100kbps; resulting live bit rate: 12250].
- This is because HLS streams can be decoded by most devices even if they're not HD capable. So as long as your material has high requirements of average quality normally seen while watching shows that are produced at Broad casters' bit rate, you are sure to produce a higher quality HLS stream than the same broadcast being served at MP4 or WebM format.
- Depending on your content and end user devices you may need larger file sizes than what we normally deliver with streaming on desktop sized TV sets (1920×1080 pixels display resolution).

- Our first phase involved creating an instance of node-media-server that consumes raw RTMP streams and generates output streams in different formats, including HLS, DASH, WebRTC and FLV.
- The encoding process includes multiple compression levels to achieve a high-quality stream with low bandwidth requirements. This process also enables improved visual quality, because it does not introduce errors into the video stream, like lower quality methods can do.

  This method of encoding results in more efficient use of network resources, due to less data being transferred and processed over the network during playback, which lowers latency and increases efficiency
- In our second phase, we hosted a virtual machine on AWS and created a NGINX server with RTMP module which intake raw RTMP stream and generates HLS stream. Here FFMPEG was used to transcode RTMP stream to HLS segments
- By tweaking the configuration of NGINX configuration file, we were able to generate 720p stream with around 15-18 seconds' delay.
- A stream segmenter segments the media data, which is received with respect to a fixed time interval, creates a segmented file, and then produces a playlist file of metadata after the media is encoded.
- Segmented media data can instead be produced based on a predetermined duration of a media segment which is then stored. When using live streaming, data must be stored in real-time, so segmenting media data through a segmenter is not necessary.
- Segmented media data can instead be produced based on a predetermined duration of a media segment which is then stored. When using live streaming, data must be stored in real-time, so segmenting media data through a segmenter is not necessary.
- To enable continuous live streaming, each audio/video file must be formatted as MPEG-2 TS. The playlists must be updated accordingly with each MPEG-2 TS file and its URIs.




## FUTURE SCOPE

- Improved Latency: HLS requires that three segments are queued before it will allow playback, and each segment is broken down by keyframes in the video. HLS can only create a stream with super low latency (less than one second) if you encode the video going out with a keyframe every 250 ms. Each segment in the playlist would last a quarter of a second. There would be a lot of HTTP calls happening (at least four per second) and the server would be overloaded.
- Scaling using CDN: In order to handle multiple concurrent users, we can use a Content Delivery Network (CDN) which can serve static files like images, videos, and other content which can help reduce the amount of traffic needed to serve these files. CDNs increase performance and stop the choppy, slow videos.

## CONCLUSION

We can stream high-quality HLS and DASH streams with our project. Having a decent user interface and being able to select various operations easily is key. The project has great potential to be used both for personal and commercial purposes. One of the biggest advantages of using HLS is that it supports adaptive bit rate streaming, so you can stream HD video on your mobile when you're at home on your WIFI. You can watch the same HD video in a lower resolution if you're commuting on a bus or metro with limited data connection. HLS automatically handles this.

The HLS application runs on Android, iOS, Windows, Mac, Linux, Chrome OS, and other modern devices. Aside from smart TVs, set-top boxes, and gaming consoles, HLS is compatible with smart devices. The Safari browser, however, does not support MPEG-DASH. Since most iPhone, iPad, Apple TV, and macOS users use Safari, this is a major disadvantage.

The majority of browsers and devices are supported by HLS, and MPEG-DASH is codec agnostic so that you can play video streams in any format you choose

## ACKNOWLEDGEMENT

Foremost,we would like to express our sincere gratitude to our mentor Dr.Farzil Kidwai for the continuous support during the project and research paper, for their

patience, motivation, enthusiasm, andimmense knowledge. Their guidance helped in all the time of research

## REFERENCES

[1] Apple. "Understanding the HTTP Live Streaming Architecture." Understanding the HTTP Live Streaming Architecture, https://developer.apple.com/documentation/http_live_streaming/understanding_the_http_live_streaming_architecture.

[2] Syaifudin, Yan Watequlis. "Study of Performance of Real Time Streaming Protocol (RTSP) in Learning Systems." Study of Performance of Real Time Streaming Protocol (RTSP) in Learning Systems, International Journal of Engineering & Technology, 2021,https://www.researchgate.net/publication/331162020_Study_of_Performance_of_Real_Time_Streaming_Protocol_RTSP_in_Learning_Systems.

[3] SYNOPI. "What Is HLS (HTTP Live Streaming)? How HLS Works?" What Is HLS (HTTP Live Streaming)? How HLS Works?, https://www.synopi.com/hls-http-live-streaming/.

[4] Waheed, Muhammad Hamza Bin. "A New Efficient Architecture for Adaptive Bit-Rate Video Streaming." 2021, https://www.mdpi.com/2071-1050/13/8/4541/htm. Accessed 7 03 21.

[5] Wang, Jiushuang. "A study of live video streaming system for mobile devices." A study of live video streaming system for mobile devices, IEEE, 2016, https://ieeexplore.ieee.org/document/7778898.