International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 5, May - 2014

A Framework for Dual Protocol Adaptive Video Streaming using Cloud

Chandrashekhara KT
Assistant Professor; BMSIT, ISE
Bangalore, India

Prashanth P Bharadwaj
Student; BMSIT, ISE
Bangalore, India

Srivatsa M Hegde
Student; BMSIT, ISE
Bangalore, India

***Abstract*—Mobile video viewing has been increasing day by day and is becoming very popular. However due to the lack of proper technology in this area of mobile video streaming, the user experiences frequent buffering and interruptions during playback. This paper proposes a scheme to overcome these problems using a video streaming framework that supports streaming using both HTTP and RTSP protocols and leveraging cloud computing simultaneously. This dual video delivery method enables our framework to support a wide variety of devices on the client side and heterogeneous hardware on the sever side as well. Further the framework provides adaptivity to the video to make the best use of the bandwidth. This is especially significant in case of a mobile link.**

Keywords—video streaming; HTTP; RTSP; adaptive video streaming; cloud

I. INTRODUCTION

With each passing day, the number of people using their mobile device to view videos has been constantly increasing. Nowadays everyone uses videos as a means of communicating topics of interest. With this increase in mobile video viewing, the wireless mobile link capacity is unable to support the increasing number of users. This causes unsatisfied video viewing experience with a lot of buffering and interruptions. Such an experience, especially when it comes to video viewing is extremely frustrating and entirely defeats the concept of mobile video viewing.

The main concept involved in ensuring an interruption free video viewing experience to the mobile user is adaptivity of the video stream. The video stream should be capable of adapting to the changing network conditions and adjust itself in such a way that the video is delivered to the user without any sort of buffering or interruption. Adaptivity can again be of two types: 1. On the server side 2. On the client side.

There have been many techniques proposed for adaptivity, both on the server side as well as the client side. However mere incorporation of adaptivity into a streaming service solves only a small part of the mobile video delivery problem. This is due to the fact that the scenario being considered involves a mobile device and particularly a fluctuating mobile link. Mobile links can never be expected to assure a particular value of bandwidth as the link capacity constantly keeps varying due to the movement of the mobile device and the user and also due to various signal related physical phenomenon like multi-path propagation which causes destructive interference and thus reduces the signal strength and the bandwidth if the mobile link subsequently.

Hence, we propose a streaming framework optimized for the mobile device model, which specifically takes note of the fact that mobile links tend to have extremely varying bandwidth and combines this approach with adaptivity to provide the best possible video viewing experience to mobile users. We make use of the HTTP and RTSP protocols to provide dual protocol video streaming to enable the support of a wide variety of mobile devices taking into account their processing powers and battery withstanding capabilities. The dual protocol approach enables this, also keeping complexity to a minimum in the process. The framework has also been developed keeping in mind it's extensibility to include new features, if any required in the near future. Also the framework is able to interact with third party software such as Wowza and Abobe Flash Streaming Server with minimal programming effort.

We have designed the HTTP part of the framework to use Apple's HLS (HTTP Live Streaming) and the RTSP part of the framework to use Adobe's RTMP streaming. Hence, in the process we have not bounded the framework to any one particular technology nor have we tied the framework to any propriety technologies as both the technologies employed have open source implementations. Also as part of the future work we have successfully implemented and tested the extension of our framework to incorporate live video streaming from one mobile device to another.

Another major component of the framework involves the use of cloud computing as the advantages of cloud computing are well known[2]. To elucidate a few may involve the discussion of topics like scalability, elasticity and a pay as you go model. Hence cloud computing has been leveraged to support a large number of mobile devices and offer immense scalability to the framework. The framework has been tested with Amazon Elastic Cloud Compute service.

The rest of the paper is organized as follows. We introduce related work in section II, section III covers the dual protocol video streaming framework, section IV elucidates the extension of the framework that has been implemented and lastly section V talks about the performance analysis and conclusion along with the scope for future enhancement.

II. RELATED WORK

A. *Adaptive Bitrate Streaming*

Adaptive Bitrate Streaming abbreviated as ABR is an emerging area of research in the field of video streaming. Adaptive video streaming, as the name indicates refers to the streaming of video in a manner in which the video 'adjusts' itself to the varying bandwidth conditions. By 'adjusting' we mean that the video stream is reduced in size to accommodate to the bandwidth of the link at that particular instant. The reduction in size may be performed by reducing the quality of the stream, reducing the frame rate of the stream etc. In essence, we attempt to reduce the bit rate of the video stream in adaptive streaming.

Many technologies have been developed that implement adaptive video streaming with varying degrees[1],[3]-[5]. Some of the noteworthy researches carried out in this area include Microsoft's Smooth Streaming requiring their Silverlight plugin, Apple's HTTP Live Streaming, Adobe's RTMP flash streaming, Quavlive Smooth Streaming et al. Each one of these technologies has its own advantages and disadvantages, also requiring different hardware in case of each technology. There are also open source implementations that have been developed for some of the above mentioned technologies. The Apple HLS and Adobe RTMP streaming are discussed in separate sections given below.

The video formats that are primarily used for adaptive video streaming include H.264 AVC, H.264 SVC, FLV etc. H.264 AVC and H.264 SVC have also been discussed separately further. In most cases these formats are converted into raw formats of RGB and YUV for the purpose of applying adaptivity. Also adaptive streaming also requires additional processing power and storage space also on the server side. Hence it is important to strike a balance between the need of resources and the obtained advantage of the technology as over consumption of resources in case of any technology irrespective of how advantageous it may be will be null and void without any significant use.

Adaptation can take place at two places: 1. On the client side 2. On the server side. Client side adaptation techniques basically work by monitoring the client side bandwidth and then sending a request to the streaming server to stream the most appropriate video that has a bit rate that matches the client side bandwidth. Server side adaptation techniques basically work by selecting a particular video to stream, splitting the video into a number of sub streams of different quality and then streaming the most appropriate sub stream that matches the client bandwidth. Both the method have its own share of good and bad and hence it requires great prudence to select the most appropriate method depending on the application and scenario where it has to be applied.

Client side adaptation is usually coupled with reduced complexity on the server side thus allowing us to use even commodity hardware to stream videos. However it comes with the catch that the device to which the video is streamed should have a high processing capability and hence expects a certain degree of work to be performed by the mobile device also. Although this is not much of an issue with smartphones and current generation of mobile devices, yet there is the problem of battery drain to be considered.

Server side adaptation solves the problem of high processing power expectation from the mobile devices but it has the disadvantage of limited scalability. This is due to the fact that as the number of mobile users streaming form the server increase, the processing required for ensuring adaptive streaming also increases and this may overburden the server leading to freeze of server, suspending of the operations and hence interruptions in the video streaming service. Although this problem can be solved by upgrading server hardware, this approach is not economically viable. Hence server side adaptation should also be implemented after careful consideration and with a future vision in perspective.

B. *Apple HTTP Live Streaming*

HTTP Live Streaming can be abbreviated as HLS. It is a technology developed by Apple Inc in an attempt to solve the buffering problem in streaming video to mobile devices. HLS works by creating multiple streaming files of the video to be streamed with different bitrates and quality. Also the entire video is segmented into a number of small chunks, typically of ten second duration. In the beginning, a stream of a particular moderate quality is streamed to the mobile device. The client then uses its own heuristics to calculate the bandwidth on its side and requests the appropriate stream fir that bandwidth for the next time interval. Each stream has a manifest file in .m3u8 format which indicates the bit rate of the stream and the maximum and minimum bandwidth which can be used to comfortably accommodate the stream. Once the client has determined the available bandwidth on its side, it parses the .m3u8 file and determines the most appropriate video stream and requests the streaming server for it. In this way adaptation of the video takes place in HLS streaming. Hence, the video that needs to be played always has a URL pointing to the .m3u8 manifest file for that particular video. The biggest advantage of HLS streaming is that it allows the use of an ordinary HTTP server to serve the streaming video without the need of special media servers which have the problem of not being able to stream through firewalls.

C. *Adobe RTMP Streaming*

RTMP stands for Real Time Messaging Protocol. It is a propriety protocol developed by Adobe in order to support live streaming using the flash video format. It is based on the Real Time Streaming Protocol (RTSP). As in case of RTSP, RTMP requires a dedicated media server to deliver the streaming video to the clients. Adobe uses the Adobe Flash Media Server on the server side to deliver RTMP streamed video. The URL for streaming the video contains the RTMP protocol specifier in the beginning. H.264 videos can also be streamed using the RTMP protocol along with flash videos. RTMP supports adaptive bit rate streaming by taking a video, decoding it into raw format, splitting the raw video into a number of sub streams with one base sub stream and the rest enhancement layer sub streams, and selecting the appropriate number of enhancement layer subs streams to be sent along with the base stream, according to the bandwidth of the link. After the total number of sub streams have been selected, the sub streams are

again encoded and the resulting video stream is sent via NAL (Network Abstraction Layer) Packets, using an RSTP server. The client can make use of a suitable video player that supports RTMP streaming to play the videos.

D. H.264 Advanced Video Coding

The H.264 Advanced Video Coding (AVC) is a video compression standard, also known as H.264/MPEG-4 Part 10. It is the most popular video compression standard used today for the recording, compression and distributing of video content. It was developed by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC JTC1 Moving Picture Experts Group (MPEG). It is a block-oriented motion-compensation-based video compression standard. This standard is used to encode the video for Blu-Ray discs. H.264 is a lossy compression technique, however the compression algorithms are extremely efficient and hence the loss becomes imperceptible. The standard also mentions a number of profiles, which is the different ways of encoding the video and in different resolutions and qualities.

E. H.264 Scalable Video Coding

The H.264 Scalable Video Coding (SVC) is an extension of the H.264 AVC standard, the Annex G extension of it. SVC was developed to standardize a video stream that contains a number of bit streams. The sub stream can represent a lower spatial resolution (smaller screen), lower temporal resolution (lower frame rate), or lower quality video stream. H.264 AVC was developed jointly by ITU-T and ISO/IEC JTC 1. Unlike the H.264 AVC which supports only a single video stream, SVC can support multiple bit streams and hence it can be used for adaptive bit rate streaming. Among the bit streams, one particular stream becomes the base quality stream, a certain minimum quality and the rest of the bit streams are additional qualities which may be added to the base stream to further enhance the quality, provided the bandwidth capacity allows for it.

III. DUAL PROTOCOL VIDEO STREAMING FRAMEWORK

We have programmed the framework in Java. Any object oriented language of choice can be used for programming the same. A brief description of the different modules implemented in the framework is given below:

A. Basic HTTP Server

This module performs the functions that a server is expected to do. When a client requests for a service (here a request for a video to be played that is specified by the URL) the server has to handle this request. Hence this Basic HTTP Server module has request handlers that can be specified by the programmers depending on the request type and the context in which the server is being used. It also includes functions to start, stop and run the server.

B. RTSP Server

The function of the RTSP server is similar to the HTTP server. This module implements a minimal RTSP server that handles all the video requests and uses the RTMP protocol for streaming. Hence these two modules are core to the supporting of the dual protocols for video streaming. Similar to the HTTP server module, this module also has the start, stop and run functions.

C. MP4 Config

This module helps in getting the SPS (Sequence Parameter Set) and PPS (Picture Parameter Set) parameters from the video to be played. The two entities in the H.264 bit stream: the Sequence Parameter Set (SPS) and the Picture Parameter Set (PPS). Both entities contain information that a H.264 decoder needs to decode the video data, for example the resolution and frame rate of the video. The H.264 bit stream contains a sequence of Network Abstraction Layer (NAL) units. The SPS and PPS are both types of NAL units. The SPS NAL unit contains parameters that apply to a series of consecutive coded video pictures, referred to as a "coded video sequence" in the H.264 standard. The PPS NAL unit contains parameters that apply to the decoding of one or more individual pictures inside a coded video sequence.

D. MP4 Parser

As the name suggests this module parses the contents of an MP4 file. The MP4 file has a tree structure where each node has a name and a size. It also has functions to find SPS and PPS parameters that are them sent to MP4Config. It also has the stsdBox which contains video compression related information regarding the decoding format used. Also it contains multiple tracks for each media type it contains.

E. H.264 Stream

This module is essentially related to the video streaming portion. The H.264 is a standard for video encoding and decoding and provides compression at a better quality than previous technologies. The H.264 Stream is derived from Video Stream module which in turn is derived from Media Stream module. Its function is to start stop or prepare the stream of data (video) that is being transmitted to client. But before transmission encoding or decoding has to be done which is done by the H.264 packetizer

F. AAC Stream

This is the audio version of H.264 Stream, in the sense it helps in streaming audio supporting compression and decompression by calling the AAC ADTS Packetizer module. AAC Stream essentially has functions that help to start, stop and prepare the stream (audio) before transmission.

G. Session

This module is used to create a session between the client and the server, before streaming starts between them. Stream

is referred to as track here. Two types of tracks namely audio and video are to be accommodated. A session essentially is a socket creation to which the client attaches himself and periodically if any client is requesting service and if yes it services the client.

H. H.264 Packetizer

Here the packetization of stream data is done. data in the form of NAL units are received from the H264Stream and these are then packetized into units which have data not more than 1400 bytes. Sometimes the NAL units could be larger than expected, in such cases it is split into appropriate number of units and sent.

I. AAC ADTS Packetizer

This is the packetization module in case of audio. The principles remain the same as far as packetizing audio and video are concerned. The change however is seen in the packet size which is 100 bytes. Again any packet that is larger than average size is split into appropriate sized packets.

Hence the above mentioned modules form the proposed framework and based on the above elucidation of each of the modules it is easy to see how the framework can be employed to stream videos. The main advantage of the proposed framework is the fact that, it can be used to support a wide variety of devices and is not restricted to devices manufactured by a particular set of manufacturers. This is due to the fact that it support two protocols for video streaming and hence can cover a whole gamut of devices. The flow of control among the modules is shown in Fig 1.

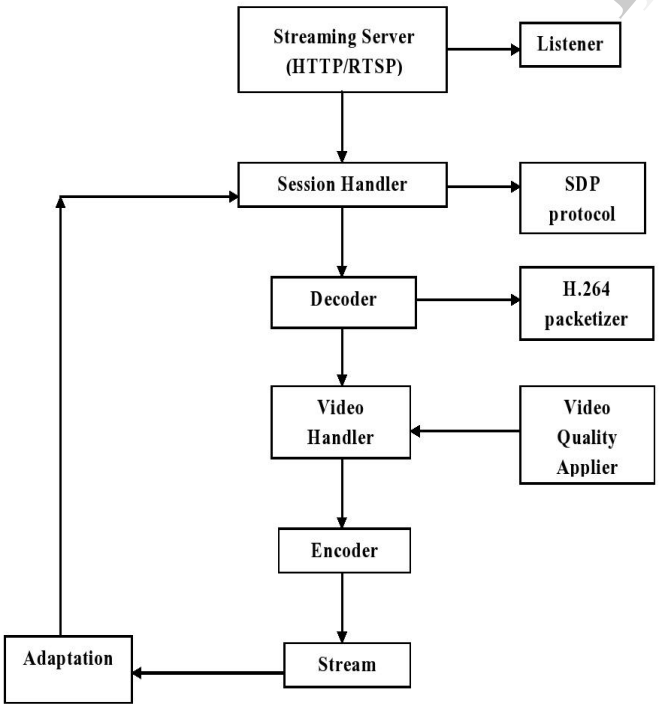

Fig 1. Flow of control between modules

requests for service. The server on the other hand carries on with the job of listening on the port, checking

On the server side too, the framework can provide much needed flexibility as a normal HTTP server can be used to store and serve the videos which is economically very beneficial and also helps in tackling security issues such as the streaming of videos through firewalls. Also the framework can be used with a dedicated media server which may be required in certain situations where the number of users using the streaming service is large and hence to ensure the supporting of a large number of users a normal HTTP server would prove to be inefficient. The overall architecture of the framework is shown in Fig 2. Fig 3 gives the sequence diagram of a user interacting with the video streaming framework to play a video.

IV. EXTENSION OF FRAMEWORK

As part of extending the framework to support useful functionality, we have successfully implemented live streaming from a mobile device to the cloud and this stream is made accessible to any device capable of playing RTMP videos. This extension takes the stream from the camera of the mobile device and stores the video on the cloud. Then application of adaptivity to it takes place in the same manner a VOD (Video On Demand) video is adapted and streamed. Thus it enables mobile devices to easily use their camera to perform a live video broadcast with minimal resources needed.

The programming for the prototype was done using the Adobe Creative Cloud Flash SDK. The mobile device platform that was targeted was Android due to the huge majority of its number of users. The usage of the extension is as follows. Firstly the mobile device user installs the application (as an apk file) onto his mobile device. Since the Adobe Creative Cloud Flash SDK was used to build the application, it is necessary to install Adobe AIR (Adobe Integrated Runtime) in order to use the application. After the following steps, the user opens the application.

The user interface of the application is such that a text box is provided for the user to enter the URL of the cloud server to which he wishes to store the video. After connecting to the server, the user then specifies a name he wishes to give to that particular stream and then starts recording from his devices' camera by pressing the record button. This stream can be accessed from any supporting device with suitable software by specifying the URL of the cloud server followed by the name of the stream.

Hence, this prototype was mainly developed to test the ease with which the framework could be extended to support additional functionality and we have implemented the above functionality with minimum programming effort and at the same time maintaining the clean and simple slate keeping in mind which the framework was designed. Thus, the framework has a great scope for further extension and enhancement and can be considered as a playground for researchers and engineers alike. Also this prototype demonstrates the innumerable number of ways in which the framework can leverage cloud computing and make its best use to improve all types of video streaming, be it live or on demand.

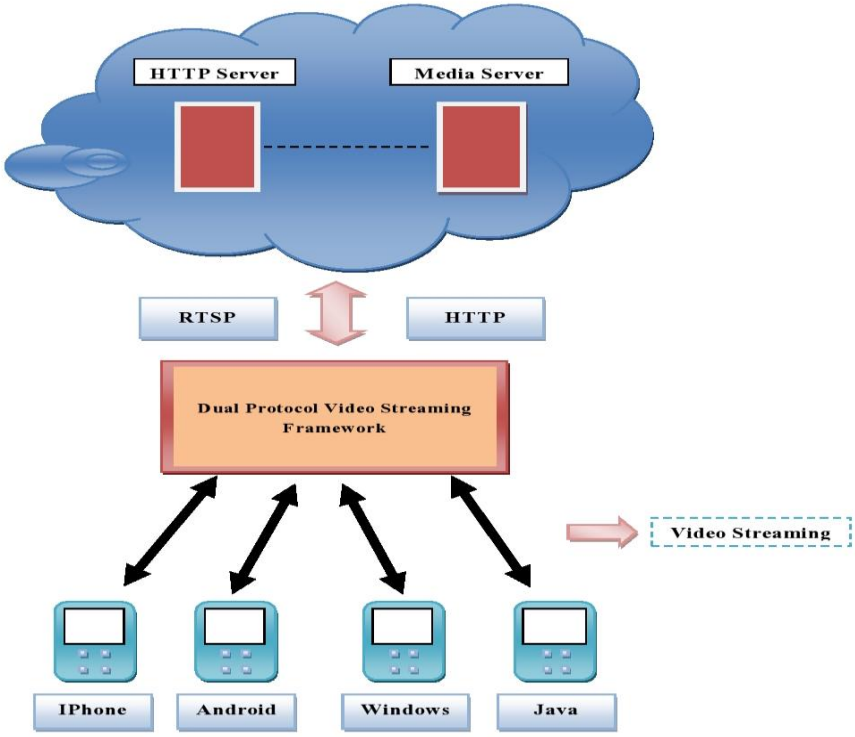

Fig 2. Architecture of the dual protocol video streaming framework

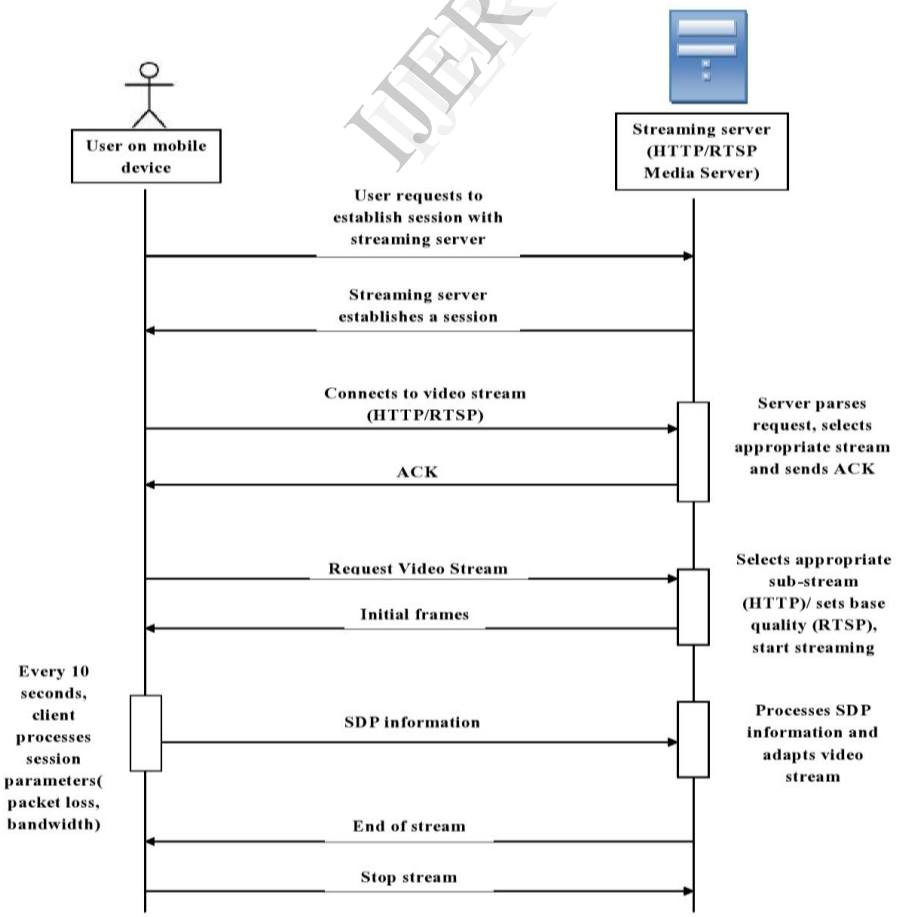

Fig 3. Sequence diagram of a user interacting with the framework

V. PERFORMANCE ANALYSIS AND CONCLUSION

As part of assessing the performance of the prototype, we tested the framework using Amazon EC2 cloud service (Elastic Cloud Compute) and used the framework to stream videos on all types of mobile devices supporting video playback. We used videos ranging from Standard Definition (SD) to High Definition (HD) for streaming. Practical tests showed that using 3G network, all the videos played on the mobile device smoothly, without interruptions and buffering, both using HLS streaming as well as RTMP streaming. Hence the performance was definitely improved compared to normal streaming which clearly buffers when streaming HD videos.

As part of future improvement in areas related to the framework it was determined that the seek performance when it comes to the videos is poor and has some latency. Also the framework can be furthered to support streaming of content using 2G networks. Also the framework can be made more compatible with a wide range of cloud service providers. Researchers can also explore the various possibilities in which the framework can be extended to include more useful functionality.

REFERENCES

[1] Xiafoei Wang, Min Chen, Ted Taekyoung Kwon, Laurence T Yang and Victor C M Leung, "AMES-Cloud: A Framework of Adaptive Mobile Video Streaming and Efficient Social Video Sharing in Clouds,IEEE Transactions On Multimedia, Vol. 15, No. 4, June 2013.

[2] Mahadev Satyanarayanan, Bahl, P, Caceres R and Davies N., "The Case for VM-Based Cloudlets in Mobile Computing," Pervasive Computing, IEEE , Volume:8 , Issue: 4, Oct-Dec 2009.

[3] Weiwen Zhang , Nanyang Technol, Yonggang Wen , Zhenzhong Chen and Khisti, A., "QoE-Driven Cache Management for HTTP Adaptive Bit Rate Streaming Over Wireless Networks," in Multimedia, IEEE Transactions, Volume:15 , Issue: 6, Oct 2013.

[4] Gopalakrishnan, V, Jana, R., Seungjoon Lee , Misra V, Ramakrishnan K.K. and Rubenstein D, "Joint-Family: Enabling adaptive bitrate streaming in peer-to-peer video-on-demand," IEEE Transactions, Oct 2013.

[5] Xiaoling Qiu, Davis, Haiping Liu , Deshi Li , Song Zhang , Ghosal, D and Mukherjee B., "Optimizing HTTP-based Adaptive Video Streaming for wireless access networks," IEEE 2010 Conference, 26-28 Ict 2013.