International Journal of Soft Computing 10 (4): 261-273, 2015
ISSN: 1816-9503

Dynamic Adaptive Streaming Over HTTP (DASH)-Comprehensive Study and Rate Adaptation Performance Analysis

Selvaraj Kesavan and J. Jayakumar
Department of Electrical and Electronics Engineering, Karunya University,
Coimbatore, Tamil Nadu, India

Abstract: Streaming enables the users to view real time multimedia content on-the-go at anywhere, any time. With the revolution of internet and increasing bandwidth support, streaming usage grows more and more. Conventional Streaming Delivery Method widely uses Real-Time Transport Protocol (RTP) over User Datagram Protocol (UDP) and Hypertext Transport Protocol (HTTP), Real time Messaging Protocol (RTMP) over Transport Control Protocol (TCP). The quality of service is a major concern in the increasing network traffic and high user demand. An adaptive HTTP streaming is the emerging technique provides advancement in terms of quality and user experience compare to the traditional streaming techniques. It dynamically selects the media bit rate based on the network variation and processing capability to ensure the best quality of service to the consumer. In this study, researchers focus on end to end study of MPEG Dynamic Adaptive Streaming over HTTP (DASH) Method with gstreamer streaming client. The rate adaptation behavior of the DASH Model experimentally evaluated with internet, test bed environment under different network conditions and the results are analyzed.

Key words: Adaptive, conventional streaming, rate adaptation, quality of service, real time

INTRODUCTION

Video occupies substantial space of internet traffic and going forward streaming video will occupy more bandwidth to send high resolution content to the consumers. Media streaming is the most proficient method to pull the video directly from server to the client media player. In heterogeneous network architecture, achievingthe best Quality of Service (QoS) is really a challenge and it requires more processing power, massive storage and high speed transport.

Conventional RTP based streaming uses UDP protocol for real time delivery with simplified client and server architecture. However, it suffers with issues like server infrastructure, Network Address Translation (NAT) and port blocked in network firewalls. Alternate to UDP, TCP can be used for streaming delivery and overcomes network proxy issues faced by the traditional streaming. HTTP progressive downloads over TCP used as a hybrid solution to download and stream the content. However, promised packet delivery and packet retransmission in TCP consumes extra bandwidth and time which restricts the real time end user experience.

The HTTP adaptive streaming is the replacement to traditional models and overcomes all the issues faced by customary streaming techniques. Adaptive HTTP streaming has multiple bit rate download approach and client driven rate control technique to overcome the network bandwidth variation and to ensure the smooth playback. Numerous Adaptive Streaming Methods developed by different vendors like Microsoft, Apple and Abode. Each Adaptive Streaming Method uses different file formats and transportation standards. Dynamic Adaptive Streaming over HTTP (DASH) common standard in adaptive streaming and used worldwide. Hence in this study, researchers discuss DASH streaming architecture characteristics and its performance over the network. The main contributions of this study include the following two aspects:

- The end to end system architecture of DASH Method is discussed in detail
- The DASH rate adaptation behavior under different network conditions arecaptured, analyzed the findings with the help of an experiment and suggests the improvements based on the existing rate adaptation algorithms

LITERATURE REVIEW

RTP is Real-time Transport Protocol (Schulzrinne *et al*., 2003) is an application layer protocol,

Corresponding Author: Selvaraj Kesavan, Department of Electrical and Electronics Engineering, Karunya University, Coimbatore, Tamil Nadu, India

uses lower level protocols such as UDP and TCP for transport across the network. RTP provides a flexible framework for delivery of real-time media such as audio and video, over IP networks. Server controls the flow and rate of data transfer. Typically the server sends the data at the bitrate of the media stream. RTCP is the Real Time Control Protocol (Huitema, 2003) designed to work alongside the Real-time Transport Protocol (RTP) to provide feedback for flow control, manages several aspects of the delivery of real-time content. Real Time Streaming Protocol (RTSP) (Schulzrinne *et al*., 1998) is stateless protocol and responsible for session initiation and control the entire streaming session. Progressive Download (PD) uses HTTP protocol for content streaming where media playback is typically started before receiving the entire stream. HTTP protocol runs on top of TCP/IP, data rate adaptation is controlled at the client side. TCP has its own congestion and flow control mechanism. HTTP server sends data to the client at the available bandwidth. Real network and Microsoft supports HTTP progressive download streaming. Real-Time Messaging Protocol (RTMP) is proprietary Adobe streaming protocol used for streaming over TCP. It manages audio, video and data for live and on-demand streaming between a flash player and a flash media server.

Adaptive streaming is a Hybrid Streaming Delivery Method using HTTP. Since, HTTP is widely used proven web protocol, media delivery over HTTP can be easily optimized to deliver real time media and adapted to the network variations. The input media divided into number of small chunks for appropriate duration, encoded in different bit rates, stored and delivered to the client sequentially using HTTP download. The client can opt for different bit rate media chunks from server. Numerous adaptive streaming methods developed by different vendors like Apple HTTP Live Streaming (HLS), Abode HTTP Dynamic streaming (HDS) and Microsoft smooth streaming (Microsoft Smooth Streaming, 2012). Each method uses different file formats and transportation standards defined by the vendors. Dynamic Adaptive Streaming over HTTP (DASH) (Stockhammer, 2011) is widely used and accepted as adaptive streaming standard by most of the industry leaders.

There are many sender and receiver based rate adaptation researches in streaming to avoid buffer overflow/underflow and smooth playback. Continuous monitoring of network variation required to transmit best possible bit rate to the client. Network metrics calculated using various parameters such as packet loss, network jitter, delay and round trip time along with client buffer usage to make the proper stream/bit rate selection either in client, server or both. UDP based Streaming Methods does not provide any rate control mechanism by the transport protocol level and the rate control is completely depends on client and server application stack. The client application sends QoS metrics in feedback messages in constant interval to server and it can be used in server to model and decide the rate switch. Rate adaptive media streaming over HTTP and TCP presents unique challenges and obstacles. TCP provides adequate congestion and flow information. The server sends at maximum possible rate and playback buffer used in client to smooth the playback control mechanism.

In adaptive HTTP streaming rate, adaptation calculated before start request every segment download. It require dynamic, Aggressive Rate Adaptation algorithm to make decisions at the client. Rate adaptation in adaptive HTTP streaming is quite new and many research is going on in this area. By Akhshabi *et al*. (2011), an experiment evaluation of internal rate adaptation algorithms was performed using the Microsoft smooth streaming, Netflix player and Adobe OSMF player to performance. Rate adaptation decision made by monitoring playback buffer level and usage was discussed by Huang *et al*. (2013). Zhou *et al*. (2013) propose the buffer based rate adaptation using parameter driven Proportional Control Model which helps choosing accurate buffering size and rate adaptation in MPEG DASH. Actual throughput based rate adaptation models are presented by Liu *et al*. (2011a, b, 2012). The methods basically decides when to fetch the next segment to keep the playback intact. Liu *et al*. (2012) proposed the adaptation metric and algorithms for Dynamic Adaptive Streaming over HTTP (DASH) with serial and parallel segment download.

ADAPTIVE STREAMING

Providing high quality audio, video in diverse platform environment is real challenge for broadcasters. Increasing user demands for state-of-art, real time, high definition video delivery without compromising content security requires capable hardware and software platforms. Set top box and digital TV platforms increasingly use web protocols for movie downloads and streaming. Conventional streaming models such as RTP, RTMP and HTTP progressive download methods poses many issues and challenges to meet these requirements. Alternatively adaptive streaming provides good quality streaming adapting to varying network circumstances and understanding the end device capabilities with reuse of existing web resources. The basic concept of adaptive HTTP streaming is to divide audio, video into number of small chunks for appropriate duration, encoded in different bit rates, stored and delivered to the client using HTTP download. Fragmented MP4 (f-MP4) and MPEG-2 Transport Format (M2TS) containers are commonly used as transportation file formats.

The most important part of adaptive HTTP streaming is the client driven rate adaptation technique. The receiving network module has the ability that determines when and how to switch different bit rates representation.

The best quality, uninterrupted playback depends on the efficiency of the adaptation logicat the receiver. Even though Smooth Streaming, HTTP Live Streaming (HLS), HTTP Dynamic Streaming (HDS) standards are created specifically by companies, the specifications are open.

To avoid many different adaptive streaming standards in market, the MPEG-LA and ISO organizations create open specification for single cross-industry standard to unify adaptive streaming over HTTP protocol called MPEG-DASH (Dynamic Adaptive Streaming over HTTP) and it is widely adopted by many vendors, solution providers and users. Hence in this study, the discussion mainly focuses on MPEG DASH standard, implementation and its merits.

Microsoft smooth streaming: Smooth streaming server stores client and server manifest files for each media file along with the media content. Client initiates the connection request with URL and server sends manifest file which comprise metadata required by the client to start the session.

Smooth streaming supports basic and advanced playback controls such as play, pause, stop, DVR support for live smooth streaming, trick play, slow motion/fast-forward/rewind, kultiple audio language support Ad playback integration, live ad insertion and DRM content protection. It provisions play ready Digital Rights Management (DRM) and play ready can be enabled with Software Development Kit (SDK) licensing or via play ready service from third party DRM provider. The files are encrypted as a single file container in Protected Interoperable File Format (PIFF) format. PIFF encryption generated using standard 128 bit AES Encryption Method. The track data should use Advanced Encryption Standard counter mode (AES-CTR) or cipher block chaining mode (AES-CBC) to encrypt the content.

HTTP Live Streaming (HLS): HTTP Live Streaming (HLS) is most popular and widely used method in Apple devices and gaining traction in IP based television environment. The basic principle of HLS is to divide the overall streaming into number small segmented MPEG2-TS files of HTTP download. The HLS implementation makes minimal addition and uses most of the existing web infrastructure. HLS client media player can play and switch the streams logically based on the network throughput and device capabilities. On the fly adaptation ensures glitch free video streaming and high quality user experience in any network. HLS supports media protection technique allows secure transmission between server and client.

HTTP Dynamic Streaming (HDS): HTTP Dynamic Streaming (HDS) is open standard streaming solution developed by adobe to support for adaptive bit rate HTTP dynamic live and on-demand streaming using HTTP caching servers and using a Fragmented MP4 container format. Like other Adaptive Streaming Methods, it also uses the existing web infrastructure and delivery the high quality content by adjusting network variations and device processing capabilities.

DYNAMIC ADAPTIVE STREAMING OVER HTTP (DASH)

The first DASH draft specification was published in February 2011. DASH supports on-demand, live and time shift applications with various content profiles, deployed using standard web servers and work with present internet infrastructures. A XML-based manifest file describes the media presentation details and playlists. The media segments with various bit representation can be requested based on the information extracted from the manifest file. With the aid of Fig. 1, we can understand the basic adaptation logic. It clearly illustrates that the bit rate changes up and down based on the network variation.

Container fileformat: MPEG Part 12 ISO Base Media File Format specification design allows two ways to organize data in MP4 file format. One way is to store entire Meta data first and then store media data. Another way is to store media data in a fragmented way. This is called fragmented MP4 (f-MP4) approach where it stores short Meta data followed by media data. It helps in the adaptive streaming use cases where the duration of media data changes very often. Also, in live streaming, the data dynamically encoded and transmitted so Meta data should be place prior to each fragment. The basic design of fragmented MP4 (f-MP4) enables stateless video delivery over standard HTTP/TCP protocol.

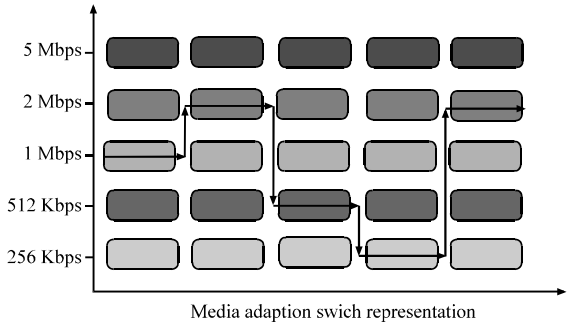

Fig. 1: DASH media adaptation logic

In f-MP4 file format audio/video are stored in individual tracks with separated Meta data. Fragments arranged in such a way that each f-MP4 starts with key frame for both on-demand and live streaming. The fragments can be decoded and rendered independently without any need of previous or future fragments. HTTP download request from client for the certain bit representation require only that specific audio, video fragments to be downloaded. When client request for particular bit representation, web server dynamically seeks to appropriate fragment in the MP4 file and retrieve the fragment from file and sends it to the client. With use of f-MP4, HTTP cache hit ratio has improved compare to MPEG-2 transport stream format. The bit overlap switching latency is negligible and uses only short video fragments while swapping.

MP4 file structure: The basic building blocks of MP4 format are called atoms. It basically describes the file type, movie information, various mediatracks, media data and sample time, etc. In f-MP4, the four key atoms are called 'moov', 'moof', 'mdat' and 'mfra'. Combination of 'mdat' and 'moof' called media fragments that givesmajor information required for the end client. The movie fragments compose of encoded audio/video/text data that can be decoded independently.

DASH profiles: MPEG DASH profiles are defined aspart of ISO/IEC 23009. It imposes set of boundaries in presentation description and media transport segments properties such as media mime types, media formats, codec types, protection formats, bit-rates, fragment durations and sizes. As shown in Fig. 2, DASH specification uses six different profiles for efficient media transport. Three profiles based on ISO base media file format. ISO base media file on-demand and live profiles are subset of ISO base media file main profile. These profiles uses MP4 container as a transport format.

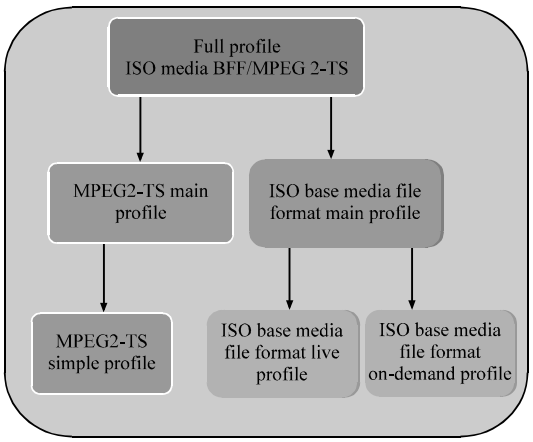

Fig. 2: DASH container profile representation

Two other profiles namely MPEG-2 TS simple profile and MPEG2-TS main profile are defined based on MPEG-2-TS media segment formats. MPEG-2 TS main profile imposes little constraint on the media segment format for MPEG-2 transport stream content. Simple profile is subset of main profiles. It defines procedure on media content encoding and transport via multiplexing. This helps clients to seamless, fast adaptation for network variations.

All the five profiles are subset of full profile. The full profile includes all features and fragment types defined in specification. It comply with description and media presentation based on ISO base media file format and MPEG2-TS.

Architecture: DASH media flow starts with sharing media presentation description file from server to client. Media file deliveryusing either ISO Common file Format (f-mp4) or MPEG2-TS format based on the profilesignaled in the Media Presentation Description (MPD) file. Figure 3 shows the graphical representation of the client server communication mechanism.

Session establishment: The media content encoded in different bit rates and the fragments, index files are stored in server. The DASH client requests the server with URL to get the Media Presentation Description (MPD) file. The server delivers the appropriate MPD file. Client parses the MPD file and extracts the adequate details to start the media playback session. The MPD file is a presentation XML file provides sufficient information to the client for selecting and switching between various streams such as profile information, presentation duration, DRM information, audio, video mime type, dynamic audio stream and video stream switching capabilities. The MPD can be divided into multiple compact sizes and the elements can be referenced externally.

Media transport: Media transport to the client starts with transferring initialization segment before transmitting actual media fragments. It is essential to increase the performance and fast channel switching. Initialization segment is either in MPEG2-TS or MP4 format carries program specific information. So, the media segment is not required to carry meta data information further. The DASH client streaming framework equipped with receiving HTTP container, parse, separate media and description

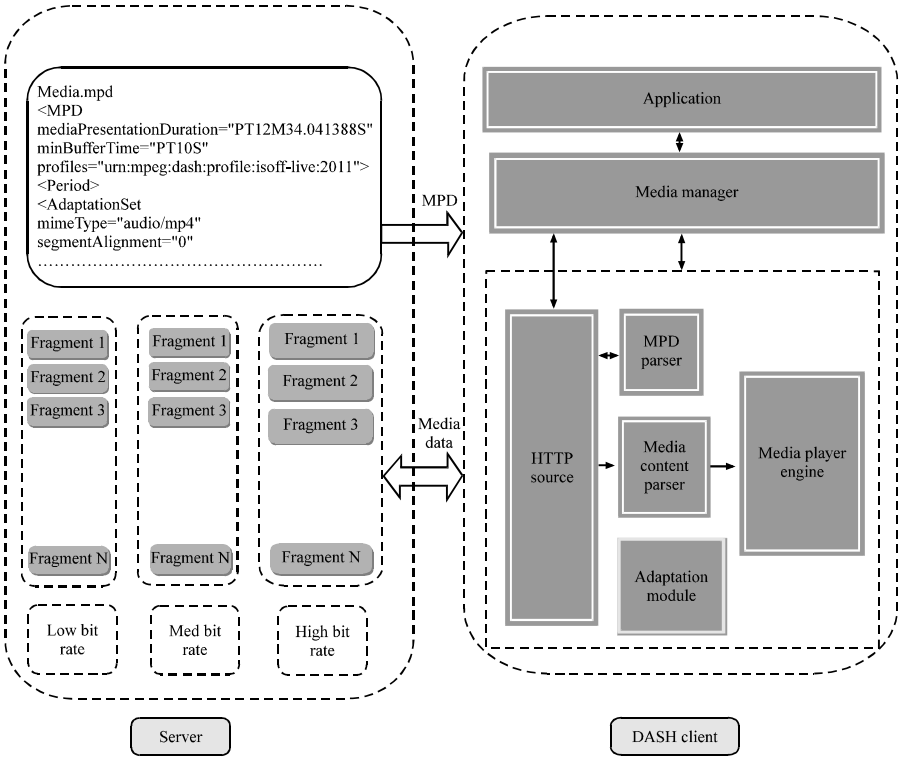

Fig. 3: DSH client server representation

information. The main part of the client is to store the media data in adaptive buffer and calculate the network variations time to time before requesting the media segments from server.

DASH client framework: The main performance difference among various adaptive streaming derived mainly from the client implementation and adaptation policy. The end user devices support DASH streaming using appropriate media player. Open source media framework gstreamer provides adequate additions in its plug-ins to enable DASH streaming. The pictorial representation of the Gstreamer DASH client is given in Fig. 4.

The MPD file parser which is part of dash demux component parse the received MPD file and stores details. Download task responsible for media data fetching, pass media fragments to the shared queue, interacts with adaptation module. The adaptation module monitors the network variation, makes decision to get optimal media representations and initiates the new representation download with server. The server keep sending the media file based on the client request till end of the session. Stream task de-queue the buffer and push down the pipeline. Whenever representation switch happens, it is

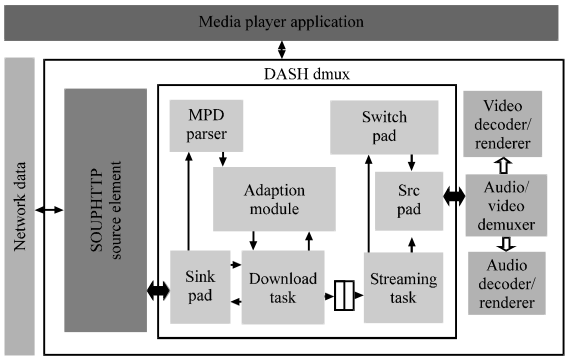

Fig. 4: DASH gstreamer client framework

necessary to negotiate the new capabilities with downstream elements (decoder and renderer) of the pipeline. The cap negotiator receives the new media capabilities and negotiates with downstream element pads.

The content can be encrypted by server and delivered to client in any one of the DRM schemes including Open Mobile Alliance (OMA), Microsoft Play Ready, Google Wide vine and Ultraviolet (UV). The supported DRM schemes can be signaled in MPD file.

Advantages: DASH poses numerous advantages over other adaptive streaming methods and some are:

- DASH specification designed to address current industry standards and practices
- It supports interoperable DRM which allows different vendor devices to communicate and stream across
- Designed to re-use most of the existing infrastructure and tools

Limitations: Even though DASH Model is very famous among the consumer, not all the vendors provide adequate support and implement full MPEG-DASH specifications in their client streaming player.

ADAPTIVE STREAMING RATE ADAPTATION MODEL

Network congestion and device capabilities plays important role in end user video experience. It is important to control network congestion and adjust the video playback to get good quality of video delivery at the end client. Even though congestion control procedures are implemented in different levels of communication network stack, the media QoS is mainly depends on server and client rate control.

Receiver driven rate adaptation: In receiver driven rate adaptation, the receiver collects all required data chooses appropriate sending rate and communicate the decision to sender to adapt the bit rate. An example of client-driven rate adaptation used in HTTP progressive streaming and described in Fig. 5.

Adaptive streaming adaptation model is new and evolving. Based on the literature sturdy, the existing adaptation logic and implementation poses many complications and challenges such as client fails to choosebest likely bitrates, interrupted playback, buffer over flow, underflow, etc. There is no specific research analyze DASH rate adaptation behavior under different network conditions.

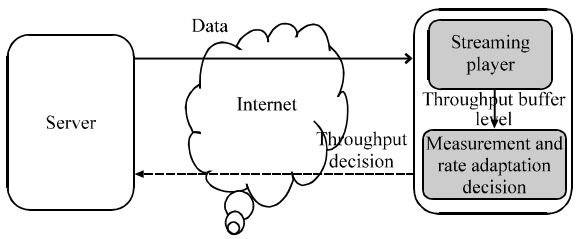

Fig. 5: Receiver Driven Rate Adaptation Model

Problem definition: The rate adaptation problem can be articulated such a way that the receiving throughput almost closely matches to the sending rate of the server so that there will not be degradation of the video stream quality and efficient use of the bandwidth. Overflow arises when the receiving rate exceeds the upper threshold and underflow occurs when receiving rate goes below lower threshold. Data loss/drop occurs when receiving rate go beyond playback rate.

To model the rate adaptation and evaluate the performance, we use the following key notations listed in Table 1. The final goal is to calculate target receiving rate R(t) over the interval, compare with the target bit rate and measure the system performance using the metrics captured. Switching to new target bit rate depends on the actual bit rate received in the client, network delay and client buffer data size. This metrics can be measured with Rate Adaptation Model and described in the study.

R(t) is the function of playback, actual received rate at client and data remaining at the client buffer. It is defined as:

$$R(t) = Fn(p(t), A(t), B(t)) \quad (1)$$

p(t) is playback rate depends on decoding rate d(t) and pre-buffered data availability. It is define as:

$$p(t) = Fn(m(t), k) \quad (2)$$

For constant bit rate encoded streaming, the media decoding rate is same for the entire duration and variable bit rate the encoding rate keep changing. Rate adaptation efficiency can be measured when the error rate between target bit rate and arrival bit rate falls within the lower and upper threshold. It can be expressed as:

$$l < E(t) < \mu \quad (3)$$

$$E(t) = R(t) - A(t) \quad (4)$$

Table 1: Rate adaptation model notations

Notation	Definition
A(t)	Actual arrival rate in client at time t>0
p(t)	Playback rate in the client at time t>0
B(t)	Data remaining at buffer at time t>0
a	Pre-buffered duration
I	Lower threshold
u	Upper threshold
E(t)	Error rate at time t>0
R(t)	target network sending rate at time t>0
d(t)	Decoding rate at time t>0
S_i	Packet sending time at server i = 0, 1, ..., N
R_i	Packet receiving time at client i = 0, 1, ..., N
D_i	Packet delay i = 0, 1, ..., N
J_i	Packet jitter i = 0, 1, ..., N

Error rate E(t) depends on many server, network and client parameters such as encoding delay, serialization delay, networ kjitter, network buffer queuing, network congestion and de-packatization delay.

Adjusting to new bit rate is required when E(t) crosses the upper or lower threshold. If E(t) is below lower threshold, R(t) should be reduced. If E(t) is above the upper threshold, R(t) should be increased. The value of l and u are important because it determines the efficiency of a rate adaptation algorithm.

Lower threshold calculated as sufficient amount of data required to continue with current playback rate, data arrival rate and pre-buffered content without underrun till next rate adaptation time or next fragment download. Upper threshold calculated as sufficient amount buffer free space available to hold the date with current playback rate, arrival rate content without overrun till next adaptation time or next segment download.

Calculation of network parameters

Throughput: Multimedia applications are bandwidth greedy. Constant network throughput is critical to carry continuous multimedia for real time delivery of steaming multimedia contents. Actual incoming average throughput measurement for the interval T can be done at time t using Eq. 5:

$$\overline{A(t)} = \frac{1}{T\int_{t=0}^{T} A(t)} \tag{5}$$

Play out buffer management: In streaming session, playback buffer is filled certain amount before starting the actual playback. Playback buffer bring the balance between network jitter and playback. It also removes the effects of jitter from the stream by buffering each arriving packet for a short interval before playing it out. Client controls the buffer management functionality.

High network jitter causes buffer underflow and re-buffering creates playback pause until the re-buffering is complete. The duration of the re-buffering time varies based on the network condition. Playback of prefilled data ensures continue playback even though high network jitter occurs sometimes. To reduce network jitter, media packets are to be captured as early as possible. Play out buffer management is required to removes jitter delay and regulates the receiving data flow. One possible solution could be dynamically adjusting the receiver buffer size so that the streaming performance can be improved. The client buffer data availability b(t) at time t can be calculated using the expression:

$$\overline{P(t)} = \begin{cases} \frac{1}{T-\alpha}\int_{t=\alpha}^{T} P(t) & t > \alpha \\ 0 & t > \alpha \end{cases} \tag{6}$$

$$B(t2) = B(t1) + \int_{t=t1}^{t2} A(t) - \int_{t=t1}^{t2} P(t) \tag{7}$$

Inter arrival jitter: Receiving throughput fluctuates mainly due to network jitter. It is the deviation of the difference in packet spacing at the receiver compared to the sender for two consecutive packets. The jitter may vary from packet to packet due to variations in the network conditions. Low latency network, the delay is acceptable to delivering real time content. Network jitter can be calculated using Eq. 8. Packet delay between ith and (i-1)th packet:

$$D(i, i-1) = (Ri-Ri-1)-(Si-Si-1) \tag{8}$$

Packet Jitter at ith packet:

$$J(i) = J(i-1) + \frac{D(i, i-1)}{16} \tag{9}$$

Client device capabilities: Even though proper network resources available, selection of capable device plays important role in quality of streaming service. The device capability metrics such as size, display, hardware acceleration, processing power and memory needs to be considered while choosing the device for multimedia applications.

EXPERIMENTAL SETUP

The experimental implementation was performed in two stages. The first stage consists of internet experiment which was used to analyze the rate adaptation behavior of the system with global networking resources. The second testing environment consists of test-bed laboratory experiment which was used to investigate the system performances under local networking capabilities.

The Gstreamer player running in Ubuntu Linux 12.04 Desktop Machine having Kernel Version 3.8.8 used as streaming client. The DASH streaming evaluation has been carried out network simulation with different internet and test bed bandwidth configuration using dummy net tool. Dummy net is a network emulation tool which serves to access and limit link bottleneck. Internet and test bed experiments use the same segmented video files for testing. The bandwidth is configured from 512 Kbps to 8 Mbps in order to study the system behavior under low, medium and high network availability. The experiment is aimed to measure and compare the following metrics useful to analyze the performance of the rate adaptation.

Average throughput: The average received output of the streaming media is calculated for the duration of 300 sec.

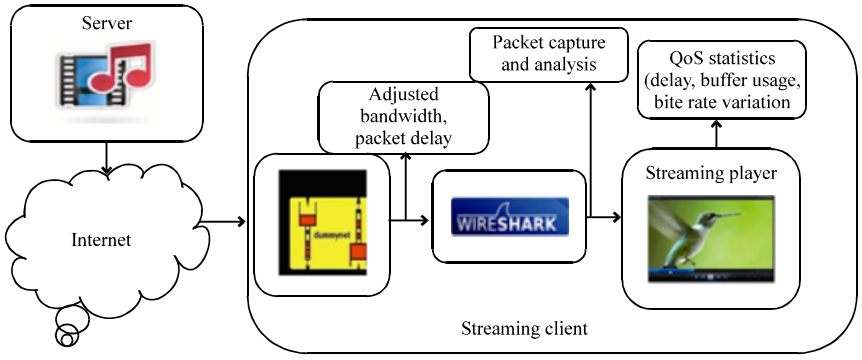

Fig. 6: Experimental setup

Average throughput is one of the metricto measure the effectiveness of the stream switching algorithms for the available bandwidth.

Average PSNR: Average value of peak signal to noise ratio of the received video. PSNR gives the efficiency of the received throughput versus the target bit rate and it defines the quality of the stream.

Average delta delay: It is the average delay between each received consecutive packet.

Average bandwidth utilization: The bandwidth utilization measures the actual use of available link capacity. It is important performance metric of the client rate adaptation.

Packet loss: The number of packets arrives later than play out latency due to network/retransmission delay or retransmitted due to loss or completely lost in transport.

Startup delay: First fragment rendering delay from the point user initiates the session.

Bit rate level change: The number of times client triggers the bit rate level change to keep the quality intact based on the session quality metrics.

Internet experiment: The internet experimental test setupis shown in Fig. 6. The representation of dataset used for testing is given in Table 2. The video is encoded with bit rates from 100-6000 Kbps and quality level ranging from 240 p up to 1080 p resolution. To analyze the system behavior for short and long duration media segments, adaptive video of 4 and 10 sec segments with 16 different levels are chosenas sample for testing. To maintain the testing consistency, each time the video streamed from public server and played over the duration of 300 sec. There are enough debug messages added in receiving streaming player DASH demux component to capture the session logs along with wire sharkpacket analysis.

Table 2: Dataset representation

Video levels	Bit rate (Kbps)	Resolution (p)	Audio (Kbps)
0	100	240	64
1	150	240	96
2	200	360	128
3	250	360	165
4	300	360	
5	400	360	
6	500	480	
7	700	480	
8	900	480	
9	1200	480	
10	1500	720	
11	2000	720	
12	2500	720	
13	3000	1080	
14	4000	1080	
15	5000	1080	
16	6000	1080	

Test bed experiment: The test-bed experimental test setup consists of local apache server and it is connected to desktop Gstreamer streaming client using Wi-Fi network. The server is configured to keep persistent connection on and the video file is encoded into different bit rates, quality levels with two different segment sizes of 4 and 10 sec. The DASH encoder also generates the manifest files required for client and server communication. The resulting video segments, manifest files are placed into the server to make the video is suitable for adaptive streaming. Dummynet was used to configure the bandwidth settings in the client.

RESULT AND ANALYSIS

Table 3 and 4 lists the outcome of the internet and test bed experiment with different bandwidth settings. The observation of each bandwidth configuration is given below.

Table 3: Internet experiment

Bandwidth	Segment size (sec)	Avg. throughput (kb sec^{-1})	Avg. PSNR (dB)	Avg. delta delay (msec)	Avg. BW utilization (%)
8 Mbit sec^{-1}	4	641.810	18.72	3.72	62.67
	10	624.420	17.64	3.98	61.02
4 Mbit sec^{-1}	4	328.120	18.06	4.08	64.08
	10	322.870	17.98	4.31	63.10
2 Mbit sec^{-1}	4	201.220	20.17	3.97	78.60
	10	198.150	20.87	4.02	77.40
1 Mbit sec^{-1}	4	100.370	20.03	3.84	78.41
	10	102.528	21.19	3.91	80.10
512 kbit sec^{-1}	4	47.040	17.19	3.96	73.50
	10	49.510	19.89	4.18	77.33

Table 4: Test-bed experiment

Bandwidth restriction	Segment size (sec)	Avg. throughput (kb sec^{-1})	Avg. PSNR (dB)	Avg. delta delay (msec)	Avg. BW utilization (%)
8 Mbit sec^{-1}	4	702.75	19.84	2.39	68.62
	10	662.08	19.14	3.02	64.65
4 Mbit sec^{-1}	4	389.75	20.18	3.08	76.12
	10	357.25	19.78	3.31	69.77
2 Mbit sec^{-1}	4	227.01	23.49	3.19	88.67
	10	217.96	22.45	3.02	85.14
1 Mbit sec^{-1}	4	107.23	21.81	3.39	83.62
	10	101.48	21.02	3.11	79.28
512 kbit sec^{-1}	4	51.08	20.24	3.91	79.81
	10	50.75	19.09	3.45	79.30

Case 1 (8 Mbps): In internet and test bed experiment, when available bandwidth is higher than the maximum bit rate of the streaming video, streaming client able to stream video at the maximum rate but not always. The frequent bit rates switch makes the system was not able to maintain consistency in highest bit rate and gives less average bandwidth utilization. The streaming client able to handle 1080 p resolution and it streams the 1080 p video. Under local network conditions the client receives average throughput of around 702 and 662 kb sec^{-1} for 4 and 10 sec segments, respectively. With public network provisions, the captured average throughput values are around 641 and 624 kb sec^{-1}. The adaptive video with 4 sec segment receives more average throughput compare to the 10 sec segments but short segments streaming suffers with more bit rate level change. The average packet arrival delay varies from 3-4 msec. Short duration segment packets are received with little less delay compare to the long duration segments. The PSNR value variesbetween 8-19 dB.

Case 2 (4 Mbps): Under the 4 Mbit sec^{-1} bandwidth configurations, the average throughput, PSNR, bandwidth utilization and average packet arrival delay of short and long duration segment are significantly improved in test bed environment and utilize the bandwidth much better than the public network environment. Sometime the requested bit rate is higher than the available network bandwidth which makes the client buffer to drain fast and video playback to freeze.

Case 3 (2 Mbps): When bandwidth reduced to 2 Mbit sec^{-1}, client streams the medium quality data with average throughput rate of around 200 kb sec^{-1} and average packet delay of 3 msec. We observe that in internet experiment, around 210 sec of the playback, few frame drops occur while streaming long segment adaptive videos. It happens due to sudden changes in the received data. Overall, it appears that the smooth playback, player attempts to keep the audio and video stream download processes as much in sync as possible.

Case 4 (1 Mbps): When bandwidth reduced to 1 Mbit sec^{-1}, the bandwidth utilization is improved with significant PSNR values. The test outcome listed in the table indicates that the rate adaptation is effective in 1 and 2 Mbps bandwidth scenario compare to the higher bandwidth conditions. In test bed experiment long duration segment gives the average bandwidth utilization is around 80% with average packet delay of 2-3 msec.

Case 5 (512 Kbps): Low constrain bandwidth settings less PSNR values. Even though the bit rate level reaches to the maximum available bandwidth within few steps, it does not stream always at maximum allowed bit rate. With the end device capability, player is able to receive and process the low quality media packets. During the session, few pause-play interrupt and glitches observed. It indicates that the data consumption from streaming player buffer is not properly matching with received data.

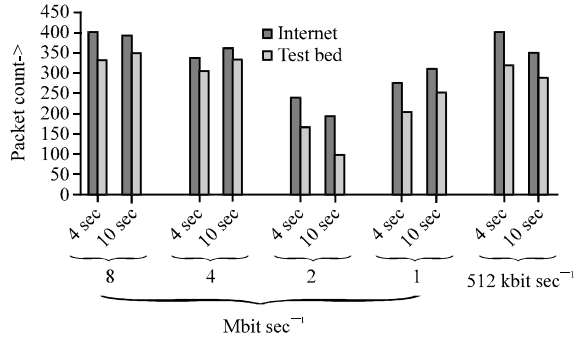

Fig. 7: Packet loss delay

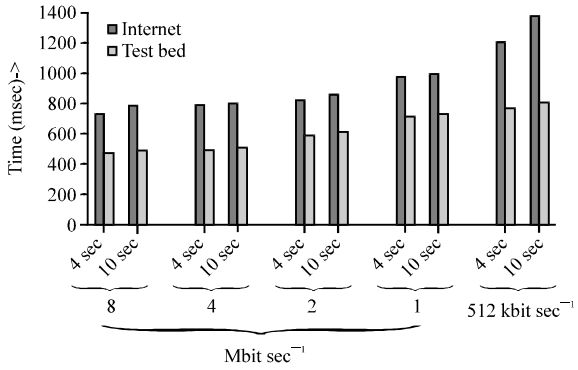

Fig. 8: Streaming startup delay

The packet loss scenario of internet and test bed experiment is depicted in Fig. 7. The packet loss is more in network lower and higher bandwidth settings andquite less range in medium bandwidth scenario. In medium bandwidth and bit rateof the stream, the system efficiently uses the resources to receive and render the data in a better way. The video rendering performance is customary with very few glitches occur due to packet loss. When bandwidth is high, video with high bit rate is streamed and sudden throughput change causes the buffer to fill fast and the client is not able handle the complete received data in time. Similarly when bandwidth is low, video with low bit rate is streamed. The client buffer drained slowly and the packets arrived later than the play out time is dropped. From the experiment we found that medium bandwidth settings work well in both test bed and internet environment.

The evaluation result in the Fig. 8 shows that the first fragment rendering delay from the point user starts the session which is called as initial fragment delay. The delay is significantly less in test bed experiment compare to the internet evaluation. In test bed research 8, 4 Mbps settings the delay is around 0.5 sec and video can be displayed almost immediately. In other bandwidth settings, the delay is 0.6-0.8 sec and buffering takes small time. In internet 8 and 4 Mbps configuration the delay is closer to 0.8 sec. Lower bandwidth configuration, the

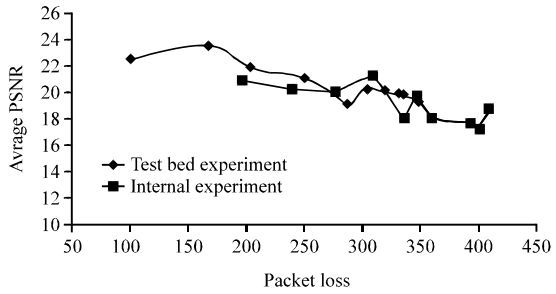

Fig. 9: Average PSNR vs. packet loss

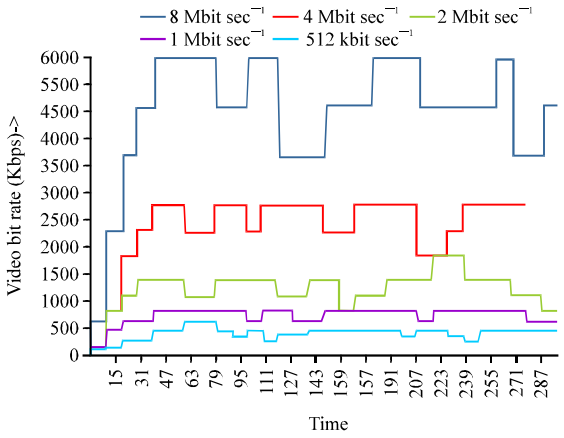

Fig. 10: Internet experiment bit rate level change -4 sec segment

delay is crosses more than a seconds and the delay is clearly visible to the end user. A good adaptive streaming player should be able to compensate the initial delay using its playback buffer and stream fast as much as possible without annoying to the user.

Figure 9 illustrates the average PSNR over 300 sec of the short and long segment streaming video a function of the packet loss rate. We could infer from the figure that as the packet loss rate increases, the overall quality of the bit stream degrades. In bandwidth settings of 2 and 1 Mbps comparatively packet rate loss rates are less and PSNR values are more. The packet loss is more and calculated average PSNR values are relatively less in high and low bandwidth scenarios. In high and low bandwidth, the system is not properly tuned to handle high and low bit rates.

The bit rate adaptation level changes of the streaming video of 4 and 10 sec segments to different bandwidth variations are described with the help of Fig. 10-13. The following are some interpretations:

8 Mbps:

- In 4 sec segment video streaming, player starts lowest to highest with 7 transitions levels within 24 sec to keep smooth ramp up

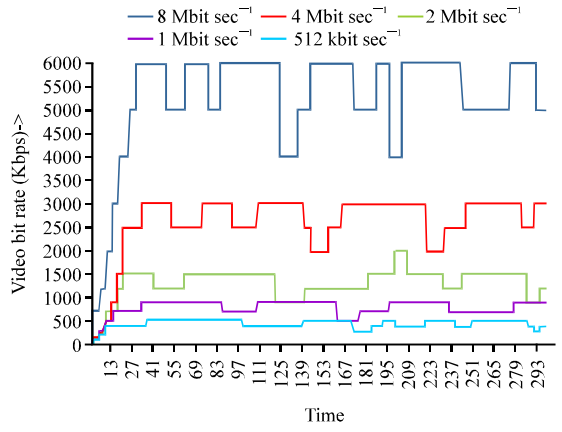

Fig. 11: Internet experiment bit rate level change -4 sec segment

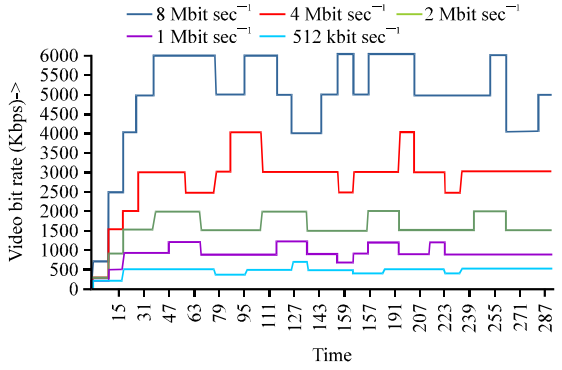

Fig. 12: Test bed experiment bit rate level change -10 sec segment

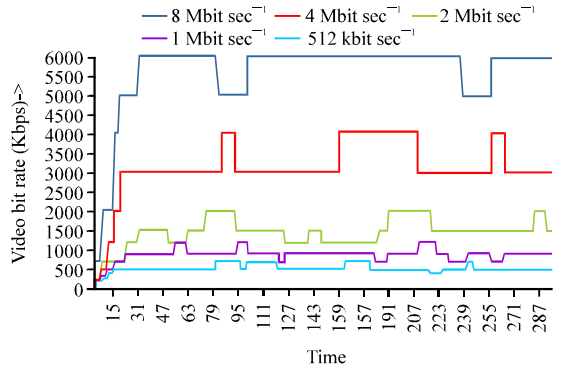

Fig. 13: Test bed experiment bit rate level change -4 sec segment

- In 10 sec segment video streaming, player takes 5 transitions, 40 sec to reach maximum bitrate. The rate adaptation takes less transition and more time than short segment streaming
- Throughput reaches the maximum bit rate of 6000 Kbps and never stream beyond the maximum available bandwidth range
- The streaming video maintains almost same bit rate consistently in both test bed and internet experiment

4 Mbps:

- In internet experiment, small and large segment video never crosses the bit rate of 3000 Kbps and it takes 5 transitions to reach 3000 Kbps. Most of the time, the bit rate varies between 2500-3000 Kbps
- In test bed experiment, video ramp up with fewer transitions. Few times the rate switch is changed to 4000 Kbps which beyond the maximum allowed 4 Mbps bandwidth. Most of the time the video bit rate level is maintained in 3000 Kbps and it never goes below 2500 Kbps after initial ramp up

2 Mbps:

- Internet evaluation, the video bit rate level switches between 1200-1500 Kbps and switched to 2000 Kbps for minimal time. For short time long segment level switched to 900 Kbps when the received throughput drops sharply
- Test bed evaluation bit rate level maintains significant time in 2000 Kbps
- The system renders the video and audio in the nominal quality without much interrupt

1 Mbps:

- In internet and test bed experiment, video bit rate level reaches to consistent level within 3 transitions
- In internet experiment, the bit rate varies between 700 and 900 Kbps after the initial ramp up where as in test bed setup it varies between 700, 900 and 1200 Kbps

512 Kbps:

- In internet environment, player starts with lowest 150 Kbps and reach 500 Kbps with 3 transitions. Though the ramp up is smooth it takes more time to buffer and start video display. The bit rate level varies between 300, 400 and 500 Kbps
- In test bed setup, most of the time the bit rate level maintained in 500 Kbps which is quite consistent than internet experiment
- Both cases, the bit rate changed to 700 Kbps for a while and it is above the allowed bandwidth limit

Based on the experiment outcome and result analysis following are the key take away points:

- Bit rate level change calculation is not completely using the available bandwidth. Sometimes requested bit rate exceeds the allowed limit and sometimes requested bit ratemuch lower than the available bandwidth
- Under lower network bandwidth, transition from lower to higher bit rate take more time and it causes start up delay. Short term and inefficient rate change trigger causes frame drop and quality degradation
- The PSNR values are low and packet loss is more in higher and lower network bandwidth settings
- Under local network settings, the rate switch is smoothening than the public network experiment
- Rate change decision is not aware of the complete end device processing constrain which causes buffer drain/overflow and play-pause interruption
- The system and rate adaptation works considerably well in bandwidth which is closer to the medium bit rate of the streaming video
- The client does not efficiently handle the packets received with very low or high delay
- The rate adaptation triggers the client reacts after event occurs in case of enormous packet loss or network problem

The adaptive streaming logic should be able to calculate the available resources and should workwell across all the bandwidth configurations. The Rate Adaptation algorithm plays important role in getting optimal performance. The rate adaptation algorithm should be robust and moredynamic. Most importantly it should consider all the factors listed as:

- Available network bandwidth and short time instability
- Measure playback buffer occupancy
- Time to download next segment
- End device processing constrain
- Video segment duration
- System should predict the network behavior in advance up to certain level and respond

CONCLUSION

In this study, we have discussed Dynamic Adaptive Streaming over HTTP (DASH) architecture with Client Server Communication Model. The problem of receiver driven rate adaptation logic is defined and the behavior of the DASH rate adaption is analyzed by streaming short and long segment duration video under various bandwidth configurations inlocal and public network environment. The performance metrics are measured for each case and the outcomes are compared. The experiment evaluation gives the behavior understanding of the HTTP adaptive streaming in the bottleneck environment and helps to understand issues and challenges of the current rate adaptation methods. It also provides more research possibilities in the area of improvement required thereby paving way for further research and development in streaming rate adaptation.

ACKNOWLEDGEMENT

Researchers would like to thank Karunya University for providing extensive guidance and support to carry out this research.

REFERENCES

Akhshabi, S., A.C. Begen and C. Dovrolis, 2011. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. Proceedings of the 2nd Annual ACM Conference on Multimedia Systems, February 23-25, 2011, San Jose, California, pp: 157-168.

Huang, T.Y., R. Johari and N. Mckeown, 2013. Downton abbey without the hiccups: Buffer-based rate adaptation for HTTP video streaming. Proceedings of the ACM SIGCOMM Workshop on Future Human-Centric Multimedia Networking, August 16, 2013, Hong Kong, China, pp: 9-14.

Huitema, C., 2003. Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP). Request for Comments: 3605, October 2003. http://tools.ietf.org/html/rfc3605.

Liu, C., I. Bouazizi and M. Gabbouj, 2011a. Segment duration for rate adaptation of adaptive HTTP streaming. Proceedings of the IEEE International Conference on Multimedia and Expo, July 11-15, 2011, Barcelona, pp: 1-4.

Liu, C., I. Bouazizi and M. Gabbouj, 2011b. Rate adaptation for adaptive HTTP streaming. Proceedings of the 2nd Annual ACM Conference on Multimedia Systems, February 23-25, 2011, San Jose, California, pp: 169-174.

Liu, C., I. Bouazizi, M.M. Hannuksela and M. Gabbouj, 2012. Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network. Signal Process.: Image Commun., 27: 288-311.

Schulzrinne, H., A. Rao and R. Lanphier, 1998. Real Time Streaming Protocol (RTSP). IETF, Request for Comments: 2326. http://www.ietf.org/rfc/rfc2326.txt.

Schulzrinne, H., S. Casner, R. Frederick and V. Jacobson, 2003. RTP: A transport protocol for real-time applications. Internet Engineering Task Force, Request for Comments: 3550. http://tools.ietf.org/pdf/rfc3550.pdf.

Stockhammer, T., 2011. Dynamic adaptive streaming over HTTP: Standards and design principles. Proceedings of the 2nd Annual ACM conference on Multimedia Systems, February 23-25, 2011, Santa Clara, CA, USA., pp: 133-144.

Zhou, C., C.W. Lin, X. Zhang and Z. Guo, 2013. Buffer-based smooth rate adaptation for dynamic HTTP streaming. Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, October 29-November 1, 2013, Kaohsiung, pp: 1-9.