Asian Journal of Information Technology 15 (8): 1322-1333, 2016
ISSN: 1682-3915
© Medwell Journals, 2016

Adaptive Video Streaming over Wireless Internet with Improved QoE in Client Server Architecture

[1]L. Arun Raj, [2]Dhananjay Kumarand and [2]A. Srinivasan
[1]Department of Computer Science and Engineering, B.S. Abdur Rahman University,
[2]Department of Information Technology, Anna University, MIT Campus, Chennai, India

Abstract: The current streaming systems involve making the client end adaptive or sending data with precautions to increase the efficiency and Quality of Experience (QoE) for the end user. Further, the efficiency of streaming reduces to a larger extent when the media content is carried over wireless networks. In this study, a novel system architecture has been proposed in which the QoE is increased by changing the specific video parameters at runtime, i.e., dynamically modifying the streaming parameters based on the client's link capability. The proposed Adaptive Client Server (ACS) algorithm is based on local maxima and local minima, using which the client identifies its incoming bit-rate of streaming video. The client samples and estimates the pattern of incoming bit-rate and sends the status report to the server in a predefined message format. Upon receiving the message the server decides and modifies the necessary video parameter based on the adaptive algorithm. The monitoring by client and listening for messages by server are carried out continuously in a feedback loop, thereby making the streaming system adaptive in a real time. Performance analysis shows that the proposed algorithm achieves an approximate increase of 25% in PSNR and 9% in SSIM than the traditional approach.

Key words: Client-server, video streaming, QoE, maxima, minima

INTRODUCTION

The demand for video streaming over a wireless network is rapidly increasing day by day. As per prediction of Cisco Visual Networking Index, mobile video will increase eleven times during 2015 and 2020 which will account for 75 % of the total mobile data traffic. Video codec inherently generates a large Variable Bit Rate (VBR) traffic special in the case of Hollywood movies and many live sports events. Further, there is a need to serve increased user satisfaction in terms of QoE than QoS (Quality of Service). This motivates to develop a suitable mechanism in the end devices working on TCP/IP networks.

Due to limited bandwidth of wireless channels and time-varying nature of radio link the video streaming employing internet protocol is challenging. Adaptive video streaming could be the solution to deliver content with improved QoE by adapting video parameters over time to time based on the prevailing link conditions (Ramamurthi and Oyman, 2014; Oyman and Singh, 2012). HTTP Adaptive Streaming (HAS) for video is a popular technique because of its inherent advantages over traditional methods such as Real Time Streaming Protocol (RTSP) (Akhshabi *et al*., 2011). Proprietary HAS based solutions for example Apple HTTP Live Streaming, Microsoft Smooth Streaming and Adobe HTTP Dynamic Streaming have become popular. HAS allows a client driven paradigm for adaptation which differs from traditional server based approaches. This provides a wide scope to dynamically adapt to varying available network resources. The Dynamic Adaptive Streaming over HTTP (DASH) by MPEG and 3GPP is a standardised approach to support video streaming (Rodriguez *et al*., 2014).

Many Adaptive Bit-rate Streaming (ABS) have been developed to facilitate and enhance the quality of service (QoS) of video delivery (Thang *et al*., 2012; Miller *et al*., 2012; Zhu *et al*., 2014; Liu *et al*., 2011). These schemes are implemented at clients and estimate the present network capacity by estimating the average throughput during past packets/segments of the video session attached with the client.

However, the link bottleneck of mobile networks is mainly due to the limitation of the radio access networks. Clearly, the throughput estimation based on past packets/segments may not provide true picture of the problem domain. Furthermore, most existing approach on link adaptation focuses on the traditional method of QoS measurements which may not be enough to estimate

Corresponding Author: L. Arun Raj, Department of Computer Science and Engineering, B.S. Abdur Rahman University, Chennai, India

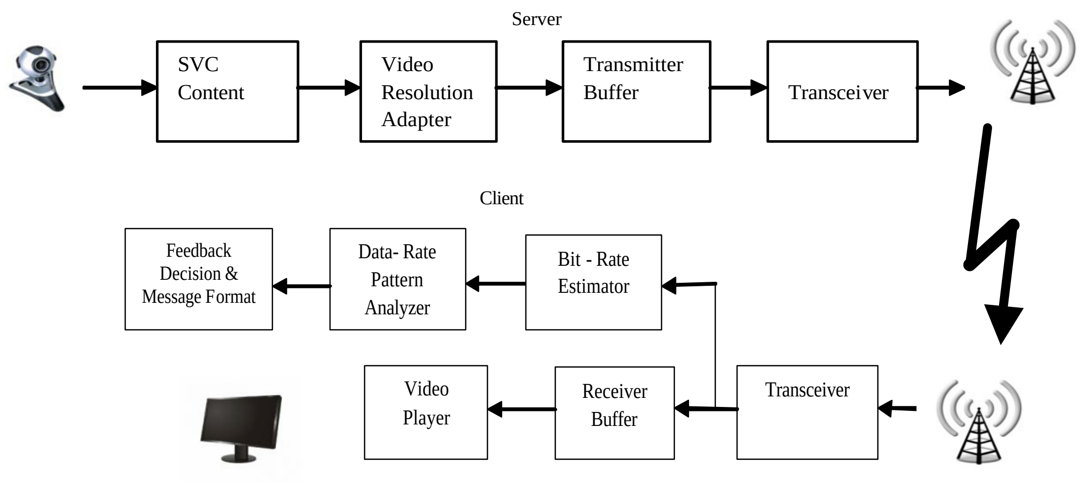

Fig. 1: Adaptive client server architecture

perceived video quality at client. This dictates to develop a mechanism to support only QoS but also QoE at the end users device (Dong *et al*., 2015).

In Device to Device (D2D) communication a QoE aware resource management scheme may be desirable. In particular, the aim of radio resource allocation methods could be to maximize the time-averaged quality of video streams transmitted (Wang *et al*., 2003). But, there is need to further improve the performance of D2D streaming with the QoS/QoE-aware solutions. This motivated the develop a On-The-Top (OTP) mechanism while leaving behind the constraints of underlying physical layer resource management issues and their limitations. The paramount interest here is to develop system which acts in feedback loop to continuously improve the quality of the video contents at client.

In this study, we propose an adaptive video streaming solution which is based on proper feedback from client systems. The system can be divided into different phases including video data streaming, rate measurement, bit rate pattern analysis, message definition, message transmission, parameter modification at the source side and updated streaming phases. The streaming parameters are modified based on how the client receives the data and estimate link capability. This approach takes up the problem of network vulnerabilities and tries to solve it by estimating link capacity based on current conditions prevailing at the client end due to the unreliability of the intermediate network.

Video streaming system architecture: The proposed video streaming systems here involves a client and a media server, at the basic level. It visualizes peer-to-peer connectivity by using two systems among which one acts as streaming server. The system incorporates the standard streaming architecture as well as model of peer-to-peer systems. The peer which acts as data server uses a web camera as a data source for live streams and storage mechanism for recorded contents Fig. 1. The server continuously listens to the client feedback messages and adjusts parameters of the outgoing video stream.

The client system or the second peer knows that the source of video and gets connected to the server using its URL (Uniform Resource Locator). An HTTP session is set up which maintains the connection between the former and the latter peers. Once, the video has been received for decoding, at the client side, several parameters relating to the received video can be retrieved. Some of the parameters include the input bit-rate, de-muxed bit-rate, lost and played pictures, number of read bytes and so on. Out of all these parameters, the input bit-rate is of primary concern to us. All these data are measured continuously till the end of the stream and stored as a separate data set. The monitoring process is carried out concurrently when the client decodes and plays the stream. Also for every N frames of the streaming video, analysis is performed for determining the link capability and content quality at the client. The bit rate pattern analyzer at client computes and maps the data rate into one of the pre-defined pattern.

MATERIALS AND METHODS

Adaptive Client Server algorithm: The proposed Adaptive Client Server (ACS) algorithm deals in identifying the network condition under which the client is getting the streams from the data source. This algorithm applies periodic evaluation technique instead of continuous evaluation method. Because if analysis is carried out for every single frame of video and changes are made based on the results of such analysis, the server may have to change the parameters of the stream too frequently. This may increase the server load and sometimes lead to incapacitation of the server. Also, there may be situations when the conditions may not be tolerable for only one frame and the server changes the configuration. But if the conditions regain normalcy for the next frame, the changes made by the server can cause degradation.

To avoid such problems, our algorithm works on a continuous set of data and not on just a single frame of data. The primary parameter for our algorithm is the input bit rate. Based on the variations of this parameter, the status of the network link at client is identified. We have designated four different statuses for the client and each status is given a code to identify itself. This helps the server understand the client's status once it receives the status code in the message from the client.

The proposed ACS algorithm has two parts: one works in the client part and the other part works in the server side. The work in the server part is minimal thus, it prevents the server from getting overloaded. On the other hand, the client side of the algorithm does extensive work to clearly identify the state of the network that feeds the streams to it. Both desirable and unfavorable conditions are identified and given status codes.

Client part: The client part of ACS algorithm has two sub tasks namely sampling the input bit rate, finding all the local maxima and local minima from the sampled data, analyzing the local maxima and minima and their variation. The client side of the algorithm is given below: The IB is Input Bit rate storage for N frames. I is Index for N frames.

Algorithm:

```
    Spanall frames in IB
    Find Local Maxima (IB)/*create the list of local maxima $l_{max}$, start, mid end-Parameters to find the local maxima next-next element in IB*/
    Repeat
    find the next three distinct values (start, mid, end)
    if start < mid > end then
    add mid to $l_{max}$
    until the list IB is traversed fully
    Find Local Minima (IB)/*Create the list of local minima $l_{min}$start, mid, end-Parameters to find the local minima next– Next element in IB*/
    repeat
    find the next three distinct values (start, mid, end)
    if start > mid < end then
    add mid to $l_{min}$
    until the list IB is traversed fully
    $m_a$<-analyse Max Curve ($l_{max}$)
    find start, end and median//median of the sample data
    if (start, median, end) produce increasing monotonicity then
    return positive value//increasing case
    else
    return negative value//decreasing case
    $m_i$<-analyseMinCurve($l_{min}$)
    find start, end and median
    if (start ,median, end) produce increasing monotonicity then
    return positive value//increasing case
    else
    return negative value//decreasing case
    if $m_a$ is positive then
    if $m_i$ is positive then
    set status as Progressive
    else
    set status as Stabilized
    else if $m_a$ is negative then
    if $m_i$ is positive then
    set status as Fluctuated
    else
    set status as Degraded
    Set status code
    Send status to source side
```

Identifying local maxima and minima: The set of input bit rate values for every N frames are collected. This set of values is acted upon to get the local maxima and local minima values. Local maximum is the value whose previous and next neighbor values are lesser than itself. Local minimum is the value whose previous and next neighbor values are greater than itself. Let I be the set of all input bit rate values fetched as part of the media statistics, then the set L_{max} contains all the local maxima from the given set I and L_{min} contains all the local minima from the given set I:

$$I = \{i_1, i_2, \ldots, i_n\} \tag{1}$$

Where: n = the number of frames observed, X_i = the set of all triplets in I such that, they are all distinct:

$$X_i = \{\text{Sets}: s(a,b,c) \text{ such that } a,b,c \varepsilon I \text{ and } a != b != c\} \tag{2}$$

Where: X_{i+} is the set of all elements after the mid element of X_i and all elements before the mid element of X_{i+1}. This is used for finding the slope of the input bit rate curve to find whether the point at X_i =A local maximum or a local minimum:

$$X_{(i+)} = \left\{ \begin{array}{l} s : s(x,y) \text{ such that } x \varepsilon (b,c) \\ \text{in } X_i \text{ and } y \varepsilon (a,b) \text{ in } X_{(i+1)} \end{array} \right\} \tag{3}$$

X_{i-} is the set of all elements after the mid element of X_{i-1} and all elements before the mid element of X_i. This is used for finding the slope of the input bit rate curve to find whether the point at X_i is a local maximum or a local minimum:

$$X_{(i-)} = \left\{ \begin{array}{l} s : s(x,y) \text{ such that } x \varepsilon (b,c) \\ \text{in } X_{i-1} \text{ and } y \varepsilon (a,b) \text{ in } X_{(i)} \end{array} \right\} \tag{4}$$

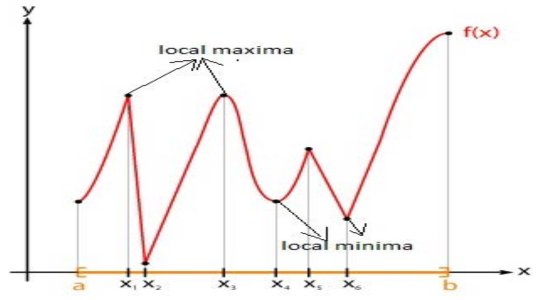

Fig. 2: Local maxima and local minima

L_{max} contains a list of elements such that each element l_{ma} in the set, l_{ma} = mid (X_i) such that:

$$\frac{d(x_{i-})}{dx} > 0; \frac{d(x_i)}{dx} = 0; \frac{d(x_{i+})}{dx} < 0 \quad (5)$$

L_{min} contains a list of elements such that each element l_{mi} in the set, l_{mi} = mid (X_i) such that:

$$\frac{d(x_{i-})}{dx} < 0; \frac{d(x_i)}{dx} = 0; \frac{d(x_{i+})}{dx} > 0 \quad (6)$$

In other words, local maximum represent a crest value and local minimum represent a trough value. In case of consecutive identical values, the next and previous different values are considered for finding the local maximum and local minimum. This is repeated for all the frames in the fetched sampled data set (Fig. 2).

Analyzing local maxima and minima: Four different cases have been identified in the variation of the local maxima and local minima. After all the local maxima and local minima are identified, they are stored in array. Now, both local maxima and minima curves can have either increasing or decreasing monotonicity. This leads to four combinations in total. Under given circumstances if both the local maxima and local minima curve have increasing monotonicity, then we designate this state as 'Progressive'. If the local maxima curve is increasing but the local minima curve is decreasing then we designate that state as 'Fluctuated'.

If the local maxima curve is decreasing but the local minima curve is increasing then we identify this state as 'Stabilized'. On the other hand if both the curves are decreasing, we identify the state as 'Degraded'. All the statuses are given codes for identity. These statuses are sent to the server along with the current frame width, height, frames per second, audio and video bit rate and checksum. The process is repeated for every set of N (N = 300) frames. The number 300 is used because, 30 frames per sec being the most prevalent temporal resolution, this gives 10 sec interval to monitor the variation of input bit rate.

Thus, the monitoring process is neither too long nor too short. The four states of the client and their respective graphs are shown in Fig. 3. The states progressive and degraded are opposite to each other. Similarly, the states stablilized and fluctuated are exactly opposite to each other.

In the server, the system receives the message sent by the client and gets the status code from the message. Using this status code, the server can identify the status of the client to be one of progressive, fluctuated, stabilized and degraded.

Server algorithm:

```
Si-Spatial resolution, Tj-Temporal resolution
S-{SQCIF, QCIF, CIF, QVGA} T-{15, 25, 30}
i, j-Indices to traverse through the collections of Spatial and Temporal Resolution respectively.
    Initial Configuration set to ($S_i$, $T_j$)-default specification set to QCIF and 30 fps
    repeat
    Receive status from client
    if Status–>Progressive//local maxima and local minima are increasing
    continue with same spatial and temporal resolution ($S_i$, $T_j$)
    else if Status–>Stabilized//local maxima decreasing and local minima increasing
    same spatial resolution $S_i$
    proceed with increased temporal resolution $T_{j+1}$
    set configuration: ($S_i$, $T_{j+1}$)
    else if Status–>Fluctuated//local maxima increasing and local minima decreasing
    same spatial resolution $S_i$
    reduce the temporal resolution to $T_{j-1}$
    set configuration: ($S_i$, $T_{j-1}$)
    else if Status–>Degraded//local maxima decreasing and local minima decreasing
    reduce the spatial resolution to $S_{i-1}$
    same temporal resolution $T_j$
    set configuration: ($S_{i-1}$, $T_j$)
    until connection is terminated
```

Based on the status of the client, the server side modifies the spatial resolution and/or the temporal resolution i.e., frame rate. Spatial resolution refers to the width and height of each frame. Common spatial resolutions for video streaming include CIF, QCIF, SQCIF and QVGA. There are other high definition formats but we have restricted our system to these four. It is designated in pixels. Temporal resolution refers to the number of frames sent per second and common temporal resolutions include 30, 25 and 15 frames per sec.

If the status of the client is found to be progressive, the server understands that the streamed content is received well at client. It then continues with the same spatial and temporal resolution. If the client's status is found to be stabilized, then the server experiments the system by having the same spatial resolution but higher

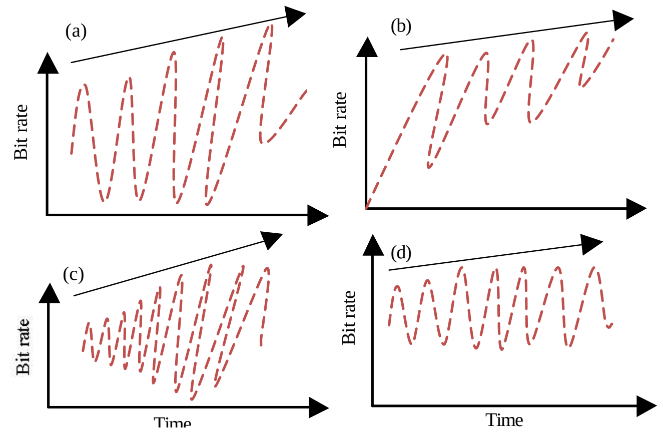

Fig. 3: Patterns of input bit rate

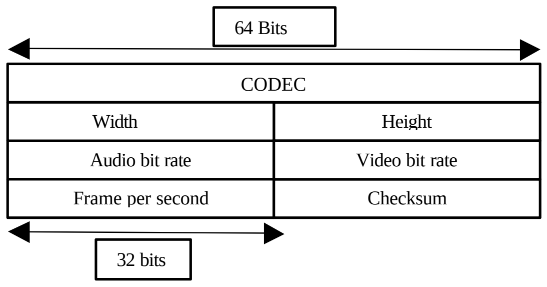

Fig. 4: Message packet format

temporal resolution. If the client's status is fluctuated, then the server reduces the temporal resolution alone. If the client's status is degraded, then the system needs utmost care and so, the server reduces the spatial resolution i.e. it sets the next lower spatial resolution.

Message format definition: Several parameters form the complete metric set of video transmission. In our system, the primary ones are the client status code, video bit rate, audio bit rate, frames per second, resolution, i.e. the width and height of frame. Each of these parameters has its own effect on the efficiency and effectiveness of video transmission. So the desired message format should encapsulate all these parameters (Fig. 4). This format must be easier for the other end to comprehend and extract all individual fields. The size of each field is set to be 32 bits for an integer type of value. The status code is the code allotted to the input bit rate status of the client. All the other parameters represent the current condition of the streaming system.

Implementation of client-server system: Here, one-to-one server-client streaming is performed using two systems. The server streams the live or stored data to client on its request. The client specifies the URL of the server to get the streams and play it. So, the whole system can be broadly categorized into two main components-source/server components and the client component.

Tasks performed by different components: The various tasks performed by the source component include:

- Streaming video data to the client
- Listening to messages from any client
- Analyzing the message to identify the state of the client
- Modifying the temporal and spatial resolution based on the state of the client

The various tasks performed by the client component include:

- Decoding and playing the stream sent by the source side
- Concurrently fetching the media statistics parameters like input bit rate, total bytes read, demux bit rate, played and lost pictures, etc
- Implement the local maxima and local minima based algorithm on specific set of consecutive frames and identify the status of the client and set a code for the status
- Encapsulate the status code, the current video configuration such as audio and video bit rate, frames per second, width and height of each frame along with the checksum of all the above data into a single message and send it to the server side

Source side workflow: The source component consists of two primary modules which run concurrently and both are dependent on each other. The first module performs the task of providing the video data stream to the client. The server needs to stream either stored contents or live

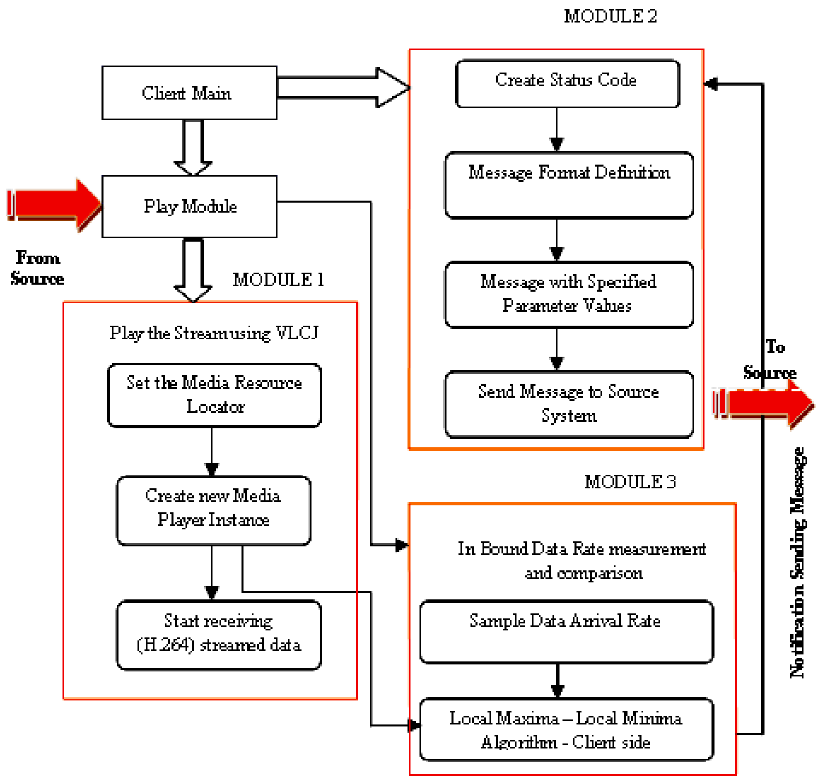
MODULE 2
Client Main
Create Status Code
Message Format Definition
Play Module
From Source
MODULE 1
Message with Specified Parameter Values
Play the Stream using VLCJ
Set the Media Resource Locator
Send Message to Source System
To Source
Create new Media Player Instance
MODULE 3
In Bound Data Rate measurement and comparison
Start receiving (H.264) streamed data
Sample Data Arrival Rate
Local Maxima – Local Minima Algorithm - Client side
Notification Sending Message

Fig. 5: Source side workflow

events. In case of a stored stream, the media resource locator is set to the location of the video file in the disk. In case of live stream, the media resource locator is set to the capture device. Once the media resource locator is set, the necessary transcoding of the video is performed. To improve efficiency and flexibility of implementation, H.264 encoding is used. While transcoding the desired number of frames per second, audio and video bit rate, resolution are set to a pre-defined value then the address to which streaming has to be carried out is specified along with the port number. The work flow of streaming process at server side is depicted in Fig. 5.

The second module opens up a separate connection in a specific port and starts listening for messages from clients. Once it receives a message, it identifies the client which has sent the message, then it identifies the status of the client from the status code with reference to the server side implementation of the local maxima local minima algorithm. This is carried out after the server separates the message from client every byte-by-byte. The server then finds whether it has to change the temporal resolution or spatial resolution or both or nothing.

Once the necessary changes are determined, the server passes the control to the send process and requests it to restart the stream with the modified parameters. Both tasks run in parallel so that both are active all the time and there is no deadlock involved in this process.

Client side work flow: The client side consists of three modules. The first module reads the stream and start playing through media player. The second module which records the media statistics implements our proposed algorithm and identifies the status of the client. The third module populates the message to be sent to the server and sends it. The play module knows the source from which it has to stream the data i.e. the IP address and the port number. It sets the above information in the media resource locator and then it starts streaming data. This headless media player is created with the help of packages provided by the open source VLC media framework. The client side workflow is represented in Fig. 6.

The second module holds the responsibility of implementing the local maxima and local minima based algorithm. The input bit-rate for N (e.g., N = 300) frames is stored in a simple data structure like a list or an array. From this fetched data set, all the local maxima and local minima points are identified and stored separately. Then the monotonicity of the measurement of both the local maxima and local minima are observed and based on this the status of the client is set. The third module reads the

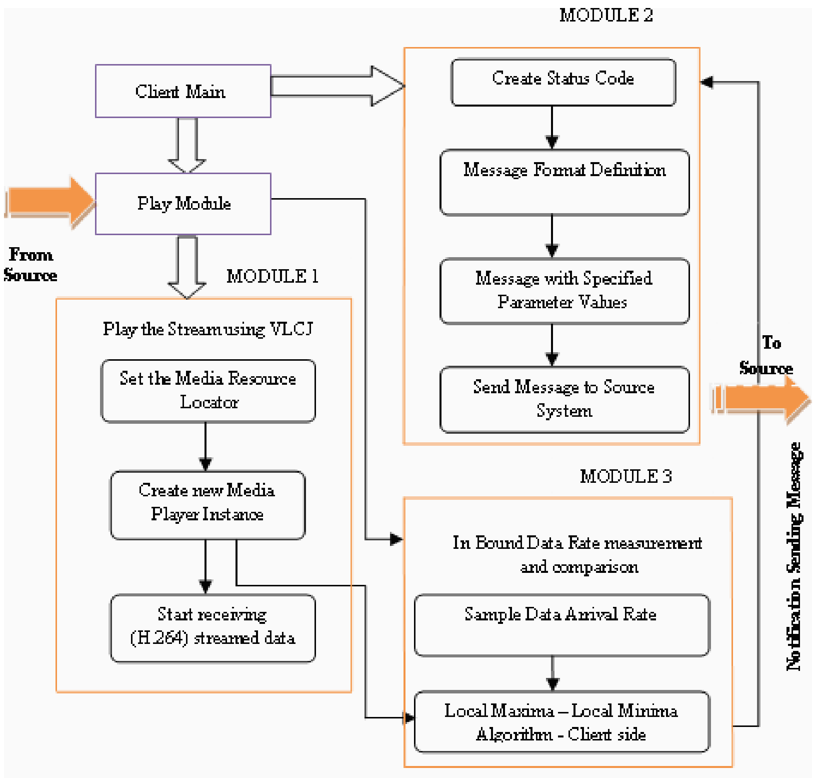
MODULE 2
Client Main
Create Status Code
Message Format Definition
Play Module
From Source
MODULE 1
Message with Specified Parameter Values
Play the Stream using VLCJ
Set the Media Resource Locator
Send Message to Source System
To Source
MODULE 3
Create new Media Player Instance
In Bound Data Rate measurement and comparison
Notification Sending Message
Start receiving (H.264) streamed data
Sample Data Arrival Rate
Local Maxima – Local Minima Algorithm - Client side

Fig. 6: Client side workflow

status of the data link analyzes it and then connects to the server which is waiting in a definite port. This module encapsulates the feedback of the client's status along with several other parameters into a message. These parameters include, the audio and video bit rate, number of frames per second and width and height of each frame. These are the values represented at the current timestamp or current condition. The checksum of all the above values is calculated and added with feedback messages; the server uses it to verify the integrity of the message over the network.

At source: Dell Inspiron N5010 desktop computer has been used as the test device. It is configured with 2.53 GHz i3 processor and 300 GB RAM. The broadband connection used in this system on part of source side is Reliance Netconnect+. It works on 3G WCDMA technology. The service provider had promised a maximum speed of 3.1 Mbps but as per the measurement by Speedtest (17), the average uplink speed is found to be 0.45 Mbps and the average downlink speed is 0.80 Mbps.

Client side: At the client a Dell Inspiron N5010 desktop computer has been used as the test device. It is configured with 2.53 GHz i3 processor and 300 GB RAM. The broadband connection used in this system on part of client side is Tata Photon+. It works on 3G CDMA technology. The service provider had promised a maximum speed of 3.1 Mbps but as per Speedtest, the average uplink speed is 0.40 Mbps and the average downlink speed is 0.75 Mbps. Successful extension of our system.

RESULTS AND DISCUSSION

The proposed system is analyzed both using objective and subjective parameters. Subjective parameters include the user perception, degree of user acceptance and Mean Opinion Score (MOS). Additionally, we have evaluated the full reference metrics objective parameters such as PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index).

(A) Peak Signal to Noise Ratio (PSNR): The Mean Square Error (MSE) is defined as:

$$\mathrm{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left[I(i,j)-k(i,j)\right]^2 \qquad (7)$$

Now, the PSNR is represented as:

Table 1: PSNR without adaptation

Frame	Value	Frame	Value
1	26.64293	14	29.71997
2	27.27006	15	29.76094
3	27.39611	16	29.88675
4	27.54983	17	30.04594
5	21.27956	18	30.04796
6	27.31895	19	30.09554
7	27.65851	20	30.28559
8	27.72627	21	30.40153
9	28.00803	22	30.46432
10	24.93596	23	30.41034
11	28.55474	24	30.37876
12	28.75276	25	30.38894
13	29.23838	26	30.25869
		Average	28.63374

Table 2: PSNR with adaptation

Frame	Value	Frame	Value
1	24.3014	14	36.23358
2	37.91204	15	37.69557
3	37.47955	16	36.12432
4	36.30337	17	36.30403
5	37.43405	18	36.27345
6	37.60236	19	36.21928
7	37.27049	20	36.34713
8	36.97328	21	36.23696
9	35.71809	22	35.26425
10	35.87423	23	35.55378
11	36.13401	24	35.45987
12	37.39451	25	35.4101
13	36.18317	26	36.1933
		Average	35.99601

$$\text{PSNR} = 10.\log_{10}\left(\frac{\text{MAX}_1^2}{\text{MSE}}\right) = 20.\log_{10}\left(\frac{\text{MAX}_1}{\sqrt{\text{MSE}}}\right) = 10.\log_{10}\left(\text{MAX}_1\right) - 10.\log_{10}\left(\text{MSE}\right) \quad (8)$$

Structural Similarity Index (SSIM): The SSIM index is computed on different windows of an image (Wang *et al.*, 2003). The measure of SSIM between two windows x and y of common size N×N is:

$$\text{SSIM}(x,y) = \frac{\left(2\mu_x\mu_y + c_1\right)\left(2\sigma_{xy} + c_2\right)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \quad (9)$$

Where:

μ_x	=	The average of x
μ_y	=	The average of y
σ^2_x	=	The variance of x
σ^2_y	=	The variance of y
σ_{xy}	=	The covariance of x and y
$c_1$and c_2	=	Two variables to stabilize the division with weak denominator given by $c_1 = (k_1L)^2$, $c_2 = (k_2L)^2$
L	=	The dynamic range of the pixel values (generally $2^{\#\text{bits per pixel} - 1}$)
k_1	=	0.01, k_2 = 0.03 by default

Tool used: A tool called Video Quality Measurement Tool (VQMT) (20) is used to find the PSNR and SSIM values for the original and the processed video. Here, the original video at the source and decoded video at the receiver are stored for evaluation purpose. Both videos are compared frame by frame and performance measures were analyzed.

Table 3: SSIM without adaptation

Frame	Value	Frame	Value
1	0.888891	14	0.886104
2	0.88527	15	0.887899
3	0.884744	16	0.887474
4	0.880751	17	0.888953
5	0.661802	18	0.890429
6	0.878823	19	0.888053
7	0.879605	20	0.891229
8	0.879985	21	0.892589
9	0.880873	22	0.891411
10	0.778868	23	0.890079
11	0.882094	24	0.888729
12	0.88308	25	0.890884
13	0.887302	26	0.890456
		Average	0.873037

PSNR observation: PSNR measurements are performed on with or without the adaptation process of the system. This is computed for a specific number of frames. Without adaptation, an average PSNR of approximately 29 dB is observed (Table 1) but with adaptation, the average PSNR value approaches 36 dB (Table 2) which may be desirable in high quality video communication. The PSNR calculation is carried out on frame by frame index, thus avoiding any frame mismatch errors which are bound to occur when videos are compared.

Considering the low end streaming server and client systems in our implementation process, the PSNR of 36 dB states the usefulness of the proposed system. The performance parameter measurements were carried out for the first 26 frames and it can be observed that the system achieves a targeted PSNR. These measurements are subject to the complexities due to compression and encoding, as they may result in empty frames in between the normal frames. Thus it is better to convert the data into raw video format and then find the PSNR.

SSIM readings: Using the same VQMT tool, the SSIM values were also calculated. The source and the client video are compared frame by frame and individually SSIM is calculated. The average value of SSIM without adaptation is nearly 0.86 as shown in Table 3.

With adaptation, the average SSIM value seems to close in on 1 (Table 4). This is a significant increase of nearly 0.1 approximately. Similar to PSNR, SSIM value also fluctuates slightly with subsequent frames but system achieves the targeted high value.

PSNR comparison plot: The comparison of, PSNR of the video stream without and with adaptation are graphically shown in Fig. 7. It shows that with adaptation, the PSNR

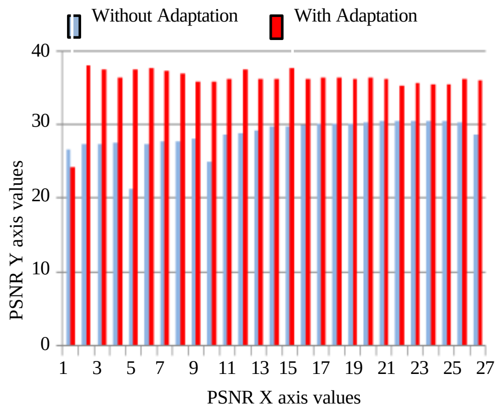

Fig. 7: PSNR of first 27 frames

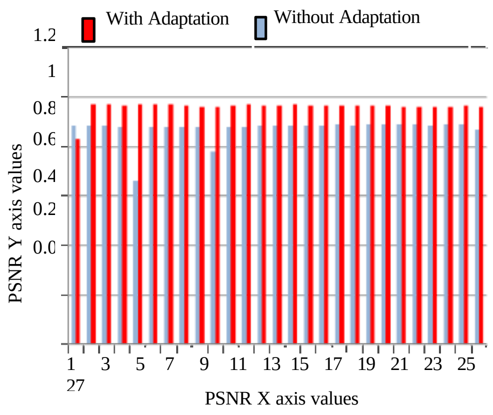

Fig. 8: SSIM of first 27 frames

Table 4: SSIM with adaptation

Frame	Value	Frame	Value
1	0.835199	14	0.966807
2	0.973332	15	0.970689
3	0.97119	16	0.965643
4	0.967383	17	0.966794
5	0.970903	18	0.966183
6	0.97159	19	0.96713
7	0.970696	20	0.966242
8	0.96833	21	0.964931
9	0.962352	22	0.960343
10	0.963191	23	0.961563
11	0.964497	24	0.959898
12	0.970917	25	0.961336
13	0.96674	26	0.965507
		Average	0.961515

Table 5: Mean opinion score

MOS	Quality	Impairment
5	Excellent	Imperceptible
4	Good	Perceptible but not annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Bad	Very annoying

value is significantly higher than that without adaptation. During the experimentation an average of 25% increase in PSNR is observed here. Since, the proposed system maintains an average PSNR of 36 dB it is suitable for the high quality video communication.

Table 6: MOS of 20 students

USERS	MOS	Users	MOS
1	4	11	2
2	3	12	3
3	4	13	3
4	2	14	4
5	3	15	4
6	4	16	4
7	4	17	4
8	4	18	3
9	3	19	4
10	3	20	3
		Average MOS	3.4

SSIM comparison plot: The comparison of SSIM of the video stream without and with adaptation is shown in Fig. 8. It shows that with adaptation, the SSIM value is significantly higher than that without adaptation. On an average there is increase in SSIM by 9% than the traditional approach during the experiment conducted on the live video streaming. The observe average SSIM of >0.96 % could be highly desirable in many high end application like tele-medicine.

Subjective analysis-mean opinion score: The MOS is generated by averaging the result of the standard subjective tests where a number of spectators rate the observed video quality of video samples streamed over the communications medium being tested. A spectator is required to give each video a rating using the following rating scheme (Table 5).

The proposed system was tested with a set of 10 students as test base. They were shown the video streaming system without and with adaptation.

They were asked to rate the system on a scale of 5. Most of the responses were positive and some were average. Figure 9-10 shows the four frames of the stored foreman video and decoded sequence respectively. The four frames of live video stream at the place of experiment with corresponding decoded sequence are shown in Fig. 11 and 12.

After assessing the video quality both without and with adaptation, several target users (students) have given positive feedback. Their ratings were taken in the form of Mean Opinion Score. The MOS given by 20 students is given below in Table 6. The average value turned out to be 3.4 which is nearing ‘good’ category.

Fig. 9: First 4 frames without adaptation (stored stream)

Fig.10: Analysis of first 4 frames with adaptation (stored stream)

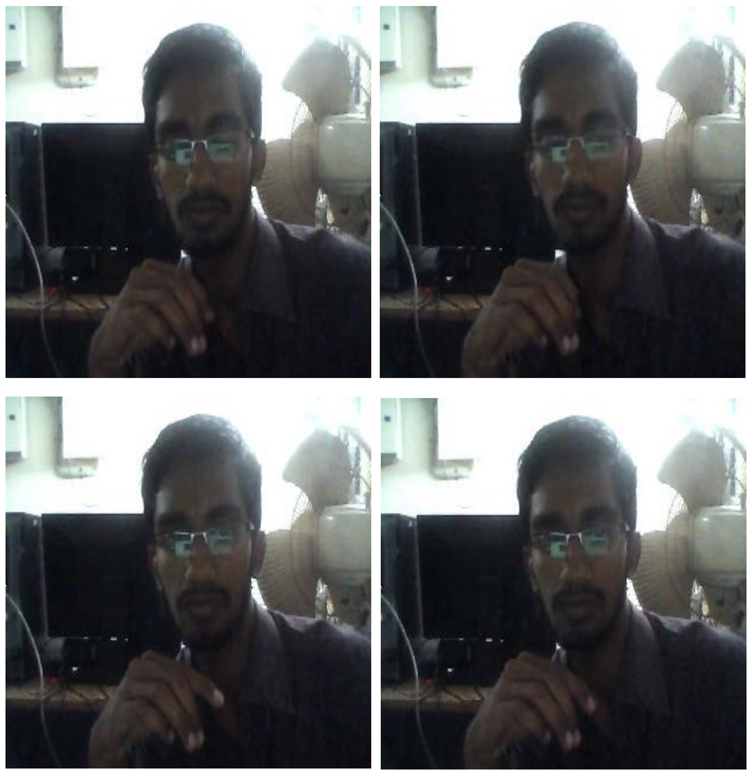
Fig.11: First 4 frames without adaptation (live stream)

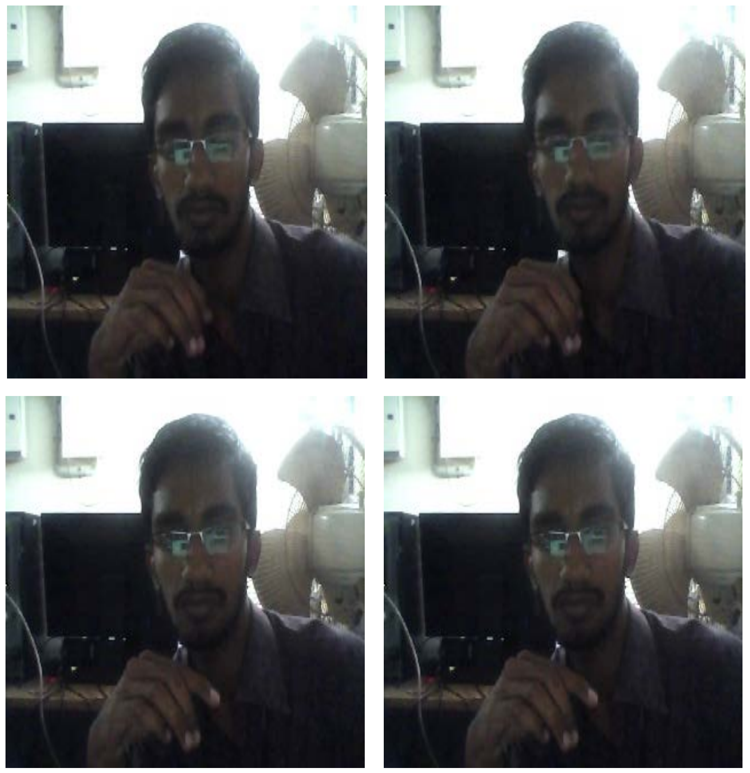
Fig.12: First 4 frames with adaptation (live stream)

CONCLUSION

An adaptive streaming system was implemented based on the local maxima and local minima. Our approach has shown a 25% increase in PSNR and 9% increase in SSIM values, respectively. This is an appreciable gain when considering the limited streaming servers and systems used in our experimental setup. Experimentation on live as well as stored video was conducted and performance observed using continuous measurements. Discrete measurements at packet level were avoided deliberately because of the difficulties posed by the unfavorable network conditions.

Specifically, changes occurring for a very short period of time don't demand parameter modifications at the source side. If a change is accommodated during transient period, it may prove to be futile. As our algorithm is tested in a real-time system, all the real-time impairments which cannot be identified by the simulation environment, are taken care of.

SUGGESTIONS

The proposed system has no third party dependency because we have used an open source framework for streaming purposes. This can be used as a model for a client-server/peer-to-peer streaming system, as there is not much services dedicated to streaming server type of systems, in our implementation. This system can be extended into a two-way streaming system so that adaptation can be done on both sides and the system then completely becomes a peer-to-peer system. It is also plan to support tele-medical video streaming in the proposed framework.

ACKNOWLEDGEMENTS

The research presented here is a part of the research project funded by the University Grant Commission (UGC), Government of India, New Delhi. We would like to thank the UGC for the financial support and permission to publish this study.

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.

Dong, K., J. He and W. Song, 2015. QoE-aware adaptive bitrate video streaming over mobile networks with caching proxy. Proceedings of the International Conference on Computing, Networking and Communications (ICNC) 2015, February 16-19, 2015, IEEE, Garden Grove, California, pp: 737-741.

Liu, C., I. Bouazizi and M. Gabbouj, 2011. 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.

Miller, K., E. Quacchio, G. Gennari and A. Wolisz, 2012. Adaptation algorithm for adaptive streaming over HTTP. Proceedings of the 19th International Conference on Packet Video Workshop (PV) 2012, May 10-11, 2012, IEEE, Munich, Germany, ISBN: 978-1-4673-0299-9, pp: 173-178.

Oyman, O. and S. Singh, 2012. Quality of experience for HTTP adaptive streaming services. Commun. Mag. IEEE., 50: 20-27.

Ramamurthi, V. and O. Oyman, 2014. Video-QoE aware radio resource allocation for HTTP adaptive streaming. Proceedings of the IEEE International Conference on Communications (ICC) 2014, June 10-14, 2014, IEEE, New York, USA., pp: 1076-1081.

Rodriguez, D.Z., Z. Wang, R.L. Rosa and G. Bressan, 2014. The impact of video-quality-level switching on user quality of experience in dynamic adaptive streaming over HTTP. EURASIP. J. Wireless Commun. Netw., 2014: 1-15.

Thang, T.C., Q.D. Ho, J.W. Kang and A.T. Pham, 2012. Adaptive streaming of audiovisual content using MPEG DASH. Consum. Electron. IEEE. Trans., 58: 78-85.

Wang, Z., E.P. Simoncelli and A.C. Bovik, 2003. Multiscale structural similarity for image quality assessment. Proceedings of the Conference Record of the Thirty-Seventh Asilomar Signals, Systems and Computers 2004, November 9-12, 2003, IEEE, New York, USA., ISBN: 0-7803-8104-1, pp: 1398-1402.

Zhu, H., Y. Cao, W. Wang, B. Liu and T. Jiang, 2015. QoE-aware resource allocation for adaptive device-to-device video streaming. Netw. IEEE., 29: 6-12