A Channel Variation-aware Algorithm for Enhanced Video Streaming Quality

Mary Riad[1] Hatem Abu-Zeid[1] Hossam S. Hassanein[1] Mazhar Tayel[2] Ashraf A.Taha[3]
[1]School of Computing, Queen's University, Kingston, ON, Canada
[2]Faculty of Engineering, Alexandria University, Alexandria, Egypt
[3]City of Scientific Research and Technology Applications, New Borg El-Arab City, Egypt
riad@cs.queensu.ca, 8ha13@queensu.ca, hossam@cs.queensu.ca, profbasyouni@gmail.com, ashraf_taha_2000@yahoo.com

***Abstract*— The demand for video streaming has been soaring in recent years. However, there is a gap between traffic demand and link capacity due to time-varying throughput fluctuation. This fluctuation results in poor quality video streaming services. The most well-known technique to improve the quality of video streaming is: adaptive video streaming. This technique adjusts the video bit-rate to the time-varying link capacity available to each user, based on the user's feedback about the link quality. Unfortunately, current quality adaptation algorithms do not incorporate channel time variation. This paper proposes a quality adaptation algorithm that adjusts video streaming quality levels for each user by measuring the channel variation experienced by that user. The algorithm periodically measures the channel throughput and calculates throughput *variance* between pairs of successive measurements. A quality selection decision is then made by comparing the channel variance to a threshold. This determines whether the quality switching will be aggressive or conservative. Through extensive testing using real datasets, representing multiple video sessions, we show that our proposed algorithm reduces the number of quality switching decisions made while maintaining a high average bitrate, compared to other adaptation schemes.**

***Keywords*—adaptive streaming, bitrate adaptation, bandwidth fluctuation, quality scheduling.**

I. INTRODUCTION

It is estimated that by 2017, mobile video will generate over 66% of overall mobile traffic [1]. As video continues to dominate the overall traffic, it will become a major contributor to network congestion. The main challenge facing video streaming is playback interruptions due to high network bandwidth fluctuations, which results in discontinuous playback in media streaming. Such bandwidth fluctuations affect user perceived Quality of Service (QoS), where the media is delivered to the client at the same rate of quality level, irrespective of the fluctuating capacity of the link. This results in video stalling when high quality fragments are transmitted during poor channel conditions as in progressive streaming.

Consequently, it is imperative that novel paradigms for video delivery maintain levels of acceptable Quality of Experience (QoE). Multi-quality Adaptive Video Streaming (AVS) has emerged as a promising paradigm. The purpose of AVS is to seamlessly adapt streaming quality to the current channel data rate. In HTTP-based adaptive streaming [2], the video content is divided into a sequence of small file segments, each containing a short interval of playback time. Each segment is made available at multiple bitrates, and depending on the channel capacity, the suitable segment quality is selected for transmission. Adaptive video streaming promises to improve user's end-to-end experience by performing the segment quality scheduling at the client side. Typically, in AVS, each client tries to estimate the available bandwidth and then chooses the video rate accordingly. However, making accurate estimations of the available bandwidth is challenging during congestion, resulting in poor user experience [3].

A quality scheduler in a media player, is able to select a suitable quality level that matches the network bandwidth capacity. In essence, it utilizes as much of the maximum available bandwidth as possible to provide the user with a high average quality video; and avoid rapid oscillations in quality. When network conditions vary greatly, configuration of the quality scheduler is challenging as there is a trade-off between maximizing the quality and reducing the number of quality switches.

To overcome these challenges this paper proposes a quality adaptation scheme that selects the most suitable quality level during video playback, by monitoring not just the channel bandwidth but the channel *variations* as well. We measure the Transmission Control Protocol (TCP) throughput of the media segments requested by the video media player, and estimate the variance of the bandwidth as well. This information is used by our algorithm to intelligently adapt between conservative and aggressive quality switching. This helps users sustain a stable quality of experience by preventing video adaptation clients from switching to the highest quality levels once they experience a high channel bandwidth. The approach depends on the

978-1-4673-6773-8/15/$31.00 ©2015 IEEE

variation of channel bandwidth to make a quality switching decision. If the user experiences a high variance; the quality adaptation algorithm selects the bitrate conservatively and when channel variation is low, the selection is done aggressively. This is made possible through a sensitivity factor that sets the range of aggressiveness and conservativeness in switching.

We conduct performance comparisons with existing quality adaptation algorithms in terms of stability in quality and bandwidth utilization. Our experimental results, on real datasets, demonstrate that our proposed scheme enables achieving more stable quality switching by decreasing the number of quality oscillations while maintaining a high average quality. The results also show the applicability of our proposed scheme in mobile scenarios where users suffer from time-varying link conditions.

The remainder of this paper is organized as follows. Section II describes existing quality adaptation algorithms. In Section III, the proposed quality adaptation scheme is introduced and its functionality is explained. Section IV describes the datasets used and details the experimental setup and reports the experimental results. Section V concludes the paper with a final review and presents future work.

II. RELATED WORKS

This section covers previous work on quality adaptation algorithm in HTTP streaming. In [4] the authors conducted an experimental evaluation of commercial players focusing on how each player reacts to available bandwidth. They conclude that some players are rather conservative in their bitrate switching decisions, which results in low average bitrates. Other players fail to converge to stable bitrates even after the available bandwidth has stabilized. Liu et al. [5] proposed an algorithm which compares the segment fetch time with the media duration contained in the segment to detect congestion, and probe the spare network capacity. The authors of [6] evaluate Microsoft Smooth Streaming [7], Adobe HTTP Dynamic Streaming [8], Apple HTTP Live Streaming [9] and Dynamic Adaptive Streaming over HTTP (DASH) [10] using real bandwidth traces. The schemes are compared in terms of how they react to high bandwidth fluctuations, and if they utilize the maximum available bandwidth with a minimum number of quality switches under vehicular mobility. They concluded that none of the four systems achieved the maximum available bandwidth with the minimum number of quality switches. In [11] the authors designed a DASH system which takes into account the effects of the transition of quality levels on the Quality of Experience.

Other research efforts relate quality metrics with user engagement. Reference [12] establishes a correlation between streaming quality metrics and user engagement. The analysis shows that there is a negative correlation between buffering and play time, and a positive correlation between video rate and play time. In [13] the authors summarize three confounding factors to unify the impact of quality metrics on user engagement: type of video: live vs. on demand, type of device: desktop or mobile, and user's connectivity: wired vs. wireless. Through their proposed QoE model, they guide the choice of which Content Delivery Network (CDN) to stream content from and the selection of video bitrate; this improves user engagement by 20%. The work in [14], describes GTube, a video streaming system for a receiver equipped with GPS, to predict the near-future bandwidth availability and plan quality adaptation accordingly. The authors use two algorithms that depend on the amount of available predicted bandwidth values to perform quality prediction. The scheme uses a window of N values of predicted bandwidth to accomplish quality adaptation.

III. VARIANCE-BASED QUALITY ADAPTATION ALGORITHM

The proposed quality adaptation algorithm receives bandwidth values that are measured and logged at the client's end. Upon receiving the measured bandwidth values, the quality adaptation algorithm determines the quality bitrate of the next media segment to be fetched each time after receiving the previous segment. The algorithm starts with a preconfigured video rate which is the lowest available bitrate. The algorithm is illustrated in **Algorithm 1**, and the notation used is presented in **Table I**.

Suppose we have a stream to be transmitted and it is encoded into C quality levels. We denote the quality levels as $0, 1, \ldots, C-1$ (0 is index of lowest level). r denotes the encoding bitrate for the current quality level used. r^-/r^+ indicate lower or higher bitrate values for the next quality level with respect to r. r_0 and r_{c-1} represent the lowest and highest bitrates. r' is the proposed quality level selected by our algorithm. ρ is the estimated bandwidth measured as the ratio of downloaded segment's data size and the delivery duration of that segment. The algorithm computes the variance σ^2 between the previous and current bandwidth values for all pairs of successive bandwidth measurements. The algorithm then constructs a vector of variances which is used to detect the variation between bandwidth values. To detect variation, the algorithm finds the variance range bounded by the minimum and maximum variance values in the vector, and chooses a suitable *cutoff point* within this range (line 2).

The quality switching decision will be made for each segment as follows: If σ^2 is larger than the cutoff point then the algorithm switches conservatively; else switch aggressively. In the conservative switching (lines5-11) the algorithms starts by multiplying the measured bandwidth value ρ by a conservative factor f which may be tuned as desired. This results in a new value ρ', which the algorithm will use next as the new measured bandwidth. The algorithm starts with current quality level r as the initial value, and compares ρ and r. If ρ is larger than r this means that the available bandwidth can support a higher video quality level, so keep increasing the quality till the first bitrate for which r' is less than ρ. If ρ smaller than r this implies the available bandwidth is not enough to transmit video segment at the current quality level, so decrease and switch to the first

Algorithm 1 Variance Quality Adaptation Algorithm

Input
$var\_array, f, r, \tau, t_{lastseg}, \rho, r_0, r_{c-1}, r^-, r^+$
Output: Selected bitrate r'
Begin
1. $r' = r$
2. $cutoff\_array$=*SelectCutoffPoints*(var_array)
3. $\rho' = \rho \times f$
4. **if** $variance > cutoff$ **then**
5. **if** $\rho' > r$ **then**
6. **while** $r^+ < \rho'$ and $r < r_{c-1}$ **do**
7. *switch up* one-level : $r' = r^+$
8. **else**
9. **while** $r^- > \rho'$ and $r > 0$ **do**
10. *switch down* one level: $r' = r^-$
11. **end if**
12. **else**
13. **if** $\rho > r$ **then**
14. **while** $r^+ < \rho$ and $r < r_{c-1}$ **do**
15. *switch up* one level: $r' = r^+$
16. **else**
17. **while** $r^- > \rho$ and $r > 0$ **do**
18. *switch down* one level: $r' = r^-$
19. **end if**
20. **end if**
21. **return** r'

bitrate for which r^- is larger. For the aggressive switching (lines 12-19), the algorithm starts with the measured throughput and performs the same steps as in the conservative switching but without applying the conservative factor. We will elaborate on the details of how we set the values of our algorithm parameters in the next section.

IV. Performance Evaluation

In this section, we introduce our system framework, and give an overview of our experimental setup and describe the metrics we focus on. The two algorithms we use for comparison are also presented.

A. System Description

The experimental evaluation of our proposed scheme has been carried out by employing the testbed shown in Figure 1. It comprises a Netflix server where all video contents are placed, a bandwidth rate limiter, a data capturing tool and a client. In our experiments we use the real backend servers of Netflix, which are real video streams. We employ the video sequences "House, M.D.", "House of Cards", and "House of Lies". The receiving host is an Ubuntu Linux machine running 3.13.0-32 kernel and internet protocol firewall ipfw tool which is a stateful firewall written for FreeBSD supporting both IPv4 and IPv6. This allows us to control the downstream available bandwidth that our host can receive. All of the video streams must pass through a rate limiter between the CDN and the client. We limit the bandwidth to 3Mb/s; this refers to the available bandwidth (*avail_bw)* and will be used throughout the paper to refer to the bitrate of the bottleneck at the host. The receiving host is connected to the Internet through Queen's University Fast Ethernet interface connection. All traffic from and to the HTTP server are captured and analyzed offline using Wireshark [15] and dumped on the receiving host. The dump files have been post-processed and parsed using a Python script.

Table I: Algorithm Notations

Algorithm Notations	
var_array:	Array of variance values based on computation on successive time windows $[t_{i-1}, t_i]$ over video stream
τ:	segment size (length)
$t_{lastseg}$	Time of downloading the last segment
ρ	throughput is computed as $\tau / t_{lastseg}$
r:	Current bitrate used by the algorithm
r_0:	lowest bitrate
r_{c-1}:	Highest bitrate
f :	conservative factor
r^-:	lower bitrate value for the next quality level with respect to r
r^+:	higher bitrate value for the next quality level with respect to r

We work on streaming video from the Netflix streaming service platform [16]. It is clear that Netflix makes use of multiple CDNs, where video content is streamed over HTTP from third-party CDN providers. Netflix players stay attached to a fixed CDN for video content delivery. To observe the basic service behavior, we created an account, and played a movie from the Netflix webiste. When a client requests a video, the service provider authenticates the user account and directs the client to a CDN hosting the video. The video service provider informs the client which video streaming rates are available and issues a token for each rate, through a debug window provided to monitor the current video playback rates [17]. We use Wireshark to capture the video traces and log it in a file to be analyzed offline. The format of Netflix fragment requests is in the form of: *"timestamp /GET /filename/byte_range?token"*. The video contents are encoded into eight video bitrates starting from 0.235 standard definition, 0.375, 0.56, 0.75, 1.05, 1.75, 2.35 and 3.0Mbps high definition.

We stream three videos of different durations: stream 1 is approximately 300 sec, stream 2, 250 seconds, and stream 3, 200 seconds. We save the captured traces in a log file, and analyze them using Wireshark. Each line uses this format: *"timestamp/GET /filename/byte_range?token"* and it corresponds to a 4 second chunk of video, and the request is done segment-by-segment. Next, we measure the bandwidth of each segment comparing the size of that segment from the *byte_range* in the request, and divide it by the timestamp field value that indicates the time it was captured. This is done for each segment, and we generate our dataset traces. We set the available bandwidth to 3Mbps which is the upper limit on the received instantaneous throughput to guarantee that our measured bandwidth, at any point in time, cannot exceed this value. We start our algorithm with a preconfigured quality (bitrate) which is denoted by r in our algorithm to 0.235Mbps that corresponds to lowest quality level. We set r_0 and r_{c-1} to 0.235, 3.0 Mbps that correspond to lowest and highest quality levels (bitrates) provided by Netflix.

The conservative parameter denoted by f in our algorithm enhances the selection of quality level by our

algorithm by making it less sensitive to the available network bandwidth, resulting in a more conservative selection of quality level. This conservative parameter takes values between 0 and 1, a conservative factor of 1 will produce the same behavior as in the aggressive switching, while a value close to 0 results in a very conservative selection of quality bitrates. This leads to an underestimation of the available network bandwidth which then leads to a bitrate selection that is far below the optimal, which would affect bandwidth utilization. Consequently, a conservative factor of 0.7 was chosen. We perform our experimental evaluation under the same set of *avail_bw* conditions and parameters, when compared to other adaptation algorithms.

B. Evaluation Metrics

We study two metrics: average used bitrate, and number of quality switches. The average bitrate is measured as the sum of the average data rates selected by the user during the whole streaming session, and is computed as follows [6]:

$$\mu_{bitrate} = \frac{\sum_{i=0}^{N} f(s_i) * t_i}{t_n} \quad (1)$$

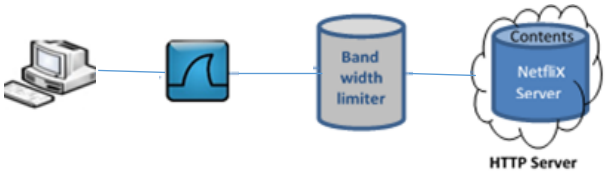


Figure 1: Experimental setup

where $i \in [0, N]$ is the segment index, t_n is the length of streaming session, t_i denotes the length of segment i, $f(s_i)$ is a function that returns the bitrate of segment i, and s_i denotes segment i.

The number of quality switches is a metric that describes the variance of the session. High values indicate very frequent switching which can lead to a decreased QoE [6] [18]. This is calculated as follows:

$$g(s_i) = \begin{cases} 1 \; if \; i = 0 \\ 1 \; if \; f(s_{i-1}) \neq f(s_i) \\ 0 \; else \end{cases} \quad (2)$$

We compare our algorithm with two well-known adaptation algorithms, namely Liu [5], and Adobe which is presented in **Algorithm 2**. In the algorithm proposed by Liu et al., the authors use $\mu = MSD/SFT$ as the transmission metric to decide whether to switch up or down (*MSD* is the media segment duration and *SFT* is the segment fetch time). If $\mu > 1 + \varepsilon.$, where $\varepsilon = \boldsymbol{max}\left\{\frac{b_{r_{i+1}} - b_{r_i}}{b_{r_i}}\right\}, \forall \, \boldsymbol{i} = [\mathbf{0}, \mathbf{1}, \ldots \boldsymbol{N} - \mathbf{1}]$, and ε denotes the switch up factor, then the chosen video rate will be switched to the next higher level. Herein, $\boldsymbol{b_{r_i}}$ denotes the encoded media bitrate of representation i, and N denotes the highest representation level. If $\mu < \gamma_d$, where γ_d denotes the switch down threshold, the chosen bitrate reduces to a level to meet $\boldsymbol{b_{r_i}} < \mu b_c$, where b_c is the bitrate of the current representation. We implemented the quality adaptation used by Adobe's Open Source Media Framework (OSFM) [8] as the second algorithm. The algorithm compares two ratios: 1) the download ratio, which is equal to the time of last segment downloaded divided by the amount of time it took to download the segment, and 2) the switch ratio: which is the quality rate selected by the algorithm divided by the rate of current quality. The quality switch decision is then made according to the procedure in Algorithm 2.

Algorithm 2 Quality Adaptation Algorithm in OSFM

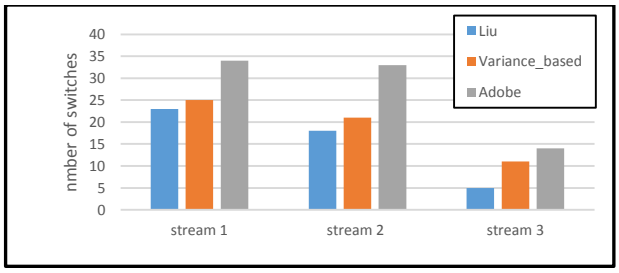
Input
€: segment duration
$t_{laststeg}$: Time of downloading the last segment
r : Current bitrate used
r': Proposed bitrate
r^0 : lowest bitrate
r_{c-1}: Highest bitrate
β: is computed as $€/t_{laststeg}$
Output : selected bitrate r'
Begin

```
if β < 1 then
  if r > r_0 then
    if β < r⁻ / r then
      r' = r_0
    else
      r' = r⁻
    end if
  end if
 else
  if r < r_{c-1} then
    while r' < r_{c-1} do
      r' = r⁺
      if β < r'/r then
        break
      endif
    end while
  end if
end if
return r'
```

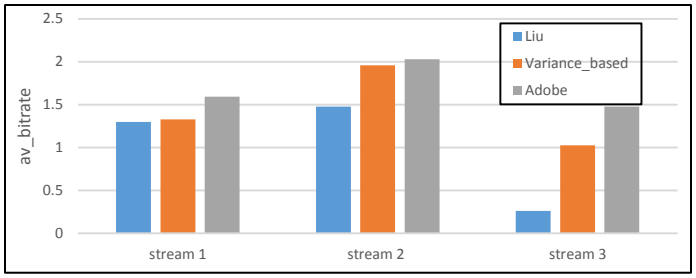
C. Simulation Results

We ran the quality adaptation algorithms on the three streams captured by the testbed and compared the results. It was observed that Liu's algorithm increases the quality in a stepwise manner up. The advantage of this stepwise approach is that the video quality will be increased much more smoothly, while the average bitrate is very low. So it cannot fully utilize the available bandwidth. The results of quality selection by Adobe showed that it seems to be unpredictable in switching. It does not take into account stability, and switches to the highest possible representation in an aggressive way. In addition it does not put a safety margin on the bitrate selection, this causes a high number of quality switches.

On the other hand, the proposed algorithm produces a stream pattern that is the comparable to the measured bandwidth, providing the user with a high average video quality by making good utilization of the available bandwidth. Additionally, rapid oscillations in quality are controlled because our algorithm depends on variation between bandwidth values to make the switching decision. This is due to the conservative factor which maintains a safety margin for switching so our algorithm does not react to bandwidth fluctuations on short notice. The conservative



(a) Number of quality switches



(b) Average bitrate

Figure 2: Number of quality switches and average bitrate for the three algorithms.

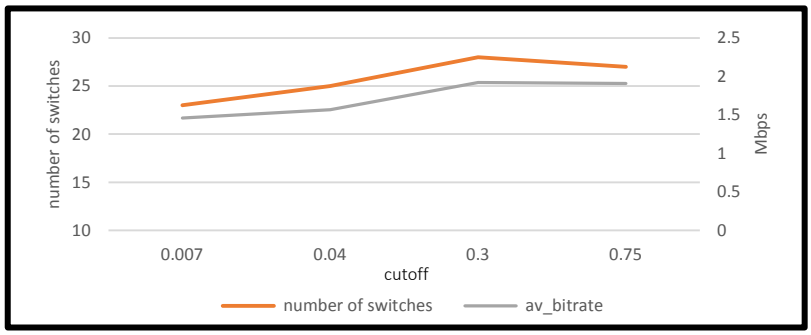


Figure 3: The effect of cutoff on average bitrate and quality switches for stream 3.

factor is not applied when the bandwidth variations are low, which improves bandwidth utilization. By periodically measuring the variation between successive pairs of bandwidth values, if the value is greater than the cutoff, the switch decision is conservative and is done as explained previously in Algorithm 1. If the value is less than the cutoff, the switching is done aggressively.

A summary of the aforementioned results is illustrated in Figure 2(a) which shows that our algorithm has a lower number of quality switches compared to Adobe. For the first stream, the number of switches for Adobe was 34 switches while for the variance-based algorithm is just 25 and the number of switches for Liu's is equal to 23. For the second stream, the number of switches for Adobe was 33 switches and for the variance-based it is 18 switches, which is considerably less than Adobe. Finally for the third stream, the switches was 11 for variance-based, 5 for Liu's and 14 switches for Adobe. We study the effect of our algorithm on the average bitrate compared to the other algorithms in Figure 2(b). For the first stream the average bitrate of our variance-based algorithm was 1.96 Mbps compared to Adobe average bitrate which is 2.03 Mbps, and for the second stream the average bitrate of our variance algorithm was 1.3 Mbps compared to Adobe 1.6 Mbps, and for the third stream it is 1.025 Mbps vs Adobe 1.47 Mbps. The average bitrate for Liu's was considerably less for all streams.

Overall the proposed algorithm achieves a good trade-off point between the number of quality switches and the average bitrate. This was achieved at a cutoff point equal to 0.3. We also investigated the effect of the cutoff on the average bitrate and quality of switches. The results are illustrated in Figure 3, which shows that for stream 3, at cutoff =0.007, 0.04,0.3, 0.75 the number of quality switches increases and the average bitrate increases. The reason for this is at low cutoff we are being too conservative and all measured throughput values will be greater than at this cutoff, which results in fewer switches. This in, turn affects the selected average bitrate. On the other hand increasing the cutoff, makes the algorithm more aggressive and the average bitrate increases and the number of switches also increases.

V. Conclusion

In this paper we proposed a new quality adaptation scheme. This scheme measures the variation among the captured bandwidth values, and then makes a quality selection decision. We have shown that we obtained a tradeoff between the number of quality switches and the average bitrate at certain cutoff points. We evaluated our scheme via experiments on real datasets, and compared it with two other algorithms, namely Liu's and Adobe. Our experiments show that the proposed algorithm was successful in minimizing the number of quality switching decisions made while maintaining a high average bitrate. In our future work we will explore ways to make the trade-offs between the average video bitrate and quality switches more seamless. This can be achieved by making the algorithm tunable, allowing individual users to adjust the parameters according to their preferences.

References

[1] "CISCO, Cisco visual networking index: Global mobile data traffic forecast update, 2014–2019.," CISCO,

[Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/index.html. [Accessed 26th May 2015].

[2] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP --: Standards and Design Principles," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, San Jose, CA, USA, 2011.

[3] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown and R. Johari, "Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard," in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, Boston, Massachusetts, USA, 2012.

[4] S. Akhshabi, A. Begen and C. Dovrolis, "An Experimental Evaluation of Rate-adaptation Algorithms in Adaptive Streaming over HTTP," in *MMSys 11*, San Jose, CA, USA, 2011.

[5] C. Liu, I. Bouaziz and M. Gabbouj, "Rate Adaptation for Adaptive HTTP Streaming," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, San Jose, CA, USA, 2011.

[6] C. Muller, S. Lederer and C. Timmerer, "An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments," in *Proceedings of the 4th Workshop on Mobile Video*, Chapel Hill, North Carolina, 2012.

[7] "Microsoft Smooth Streaming," [Online]. Available: http://www.iis.net/download/ smoothstreaming. [Accessed Dec 2011].

[8] G. Hamer, "Open Source Media Framework: Introduction and overview," [Online]. Available: http://www.adobe.com/devnet/video/articles/osmf_overview.html. [Accessed 26th May 2015].

[9] "HTTP live streaming," [Online]. Available: https://developer.apple.com/streaming/. [Accessed 13th Apr 2015].

[10] Stockhammer and Thomas, "Dynamic Adaptive Streaming over HTTP --: Standards and Design Principles," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, San Jose, CA, USA, 2011.

[11] Mok, R. K. P, Luo, Xiapu, Chan, W. W. Edmond, Chang and C. Rocky K, "QDASH: A QoE-aware DASH System," in *Proceedings of the 3rd Multimedia Systems Conference*, Chapel Hill, North Carolina, 2012.

[12] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica and H. Zhang, "Developing a Predictive Model of Quality of Experience for Internet Video," in *SIGCOMM Comput. Commun. Rev*, New York, NY, USA, 2013.

[13] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan and H. Zhang, "Understanding the Impact of Video Quality on User Engagement," *SIGCOMM Computer Communications Reviews,* vol. 41, pp. 362-373, 2011.

[14] J. Hao, R. Zimmermann and H. Ma, "GTube: Geo-Predictive Video Streaming over HTTP in mobile environments," in *Proceedings of the 5th ACM Multimedia Systems Conference*, 2014.

[15] *Wireshark,* 2015, Available: https://www.wireshark.org/download.html.

[16] K. A. Vijay, G. Yang, V. Matteo, H. Volker, S. Moritz and L. Z. Zhi, "Unreeling netflix: Understanding and improving multi-CDN movie delivery," in *Proceedings of the IEEE INFOCOM*, Orlando, FL, USA, 2012.

[17] T.-Y. Huang, A BUFFER-BASED APPROACH TO VIDEO RATE ADAPTATION, CA United States, 2014.

[18] L. N. Simula Res. Lab., "Spatial flicker effect in video scaling," in *(QoMEX), 2011 Third International Workshop on quality of multimedia experience*, 2011.

[19] "HTTP Dynamic Streaming / Features," [Online]. Available: http://www.adobe.com/products/hds-dynamic-streaming/features.html. [Accessed 20th Apr 2015].

[20] D. Cicco, S. Mascolo and V. Palmisano, "Feedback Control for Adaptive Live Video Streaming," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems, MMSys '11*, San Jose, CA, USA, 2011.