Optimized Adaptive Streaming with Intelligent Network Awareness

Prasenjit Chakraborty
Samsung R&D Institute India - Bangalore
Bengaluru, India
prasenjit.c@samsung.com

Shweta Aggarwal
Samsung R&D Institute India - Bangalore
Bengaluru, India
shweta.agg@samsung.com

Om Prakash
Samsung R&D Institute India - Bangalore
Bengaluru, India
om90.prakash@samsung.com

Abstract— **Video streaming service, considering both live as well as video-on-demand contents, has completely changed the internet world. However, buffering remains the biggest concern, which severely degrades the quality of experience. In particular, the amount of time spent in video buffering phase has the worst impact on the user engagement. This buffering phase becomes more visible while streaming in fluctuating networks, which is a common scenario when user watches streamed video while travelling or during weather aberration. In this paper, we propose an intelligent network aware adaptive streaming method which estimates the past network trend, optimizes the video queue caching mechanism and enforces video quality in client device. By doing so, the algorithm is able to reduce buffering events by average 40% and quality switches by almost 45%, providing an almost seamless video streaming playback experience.**

Keywords— ABR, DASH, HLS, HDS, SS, QoE, OEM, Queue, Buffering

I. INTRODUCTION

As per research made with BBC iPlayer usage [1] with data taken for over nine months (1.9 billion sessions of 32M monthly users), it was observed that mobile handset users often split their content consumption across different sessions. Such sessions are either first starts on fixed-line broadband and continues while on the move (53%), or starts on a cellular connection and continues on a fixed-line connection (47%). In summary, media consumption trend clearly shows major viewership is seen during day-to-day commutating or travelling.

Also, in Q1 of 2017, Mux commissioned an independent survey that asked 1,035 U.S. consumers about their viewing experience with online video [2]. As per the report shown in Fig. 1, re-buffering [5] i.e. stalling of streaming media during ongoing playback due to bad network, is the most important factor impacting the viewer's QoE. The survey wanted to evaluate the effect the buffering events on length of user's viewing session which is shown in Fig. 2. According to the report [3], streaming session length of a typical viewer is around 214 seconds when no buffering event occurs and

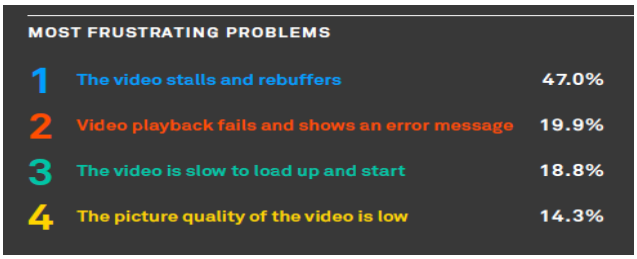


Fig. 1. Video Streaming Market Research Result

session length reduces significantly with the buffering interruptions. Moreover, in the report it is shown that 85% users stopped playback due to re-buffering and stalling events.

This user feedback has resulted in increased stress on the need to deliver a high QoE to satisfy consumers. There is already widespread adaptation of ABR Streaming approach for media content delivery over internet, which involves encoding the source video content at different bit rates and segmenting them and later in the client side, playback application switches between these different segments depending on the network to avoid buffering. There are several commercial servers available which employs ABR mechanism via different solution specification. This includes HTTP Live Streaming (HLS) by Apple, HTTP Dynamic Streaming (HDS) by Adobe, Smooth Streaming (SS) by Microsoft as well as Dynamic Adaptive Streaming over HTTP (DASH) among Content Provider (CP) favorites. All these specifications, at a broader level, follows the same ABR methodology and normally handles changes in network quality in broader level.

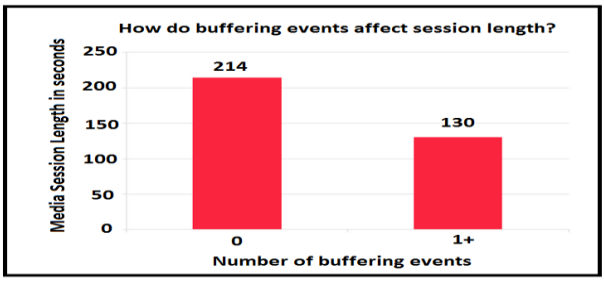


Fig. 2. Buffering Reducing Streaming Session Length

But all of these current ABR methodologies (Default-ABR) has deficiencies, which prohibits it from providing a true high quality experience, as evident from the current user survey and user experience data of Fig. 1 as well as Fig. 2. This becomes specifically visible during frequent network quality change, causing video quality switch also happening frequently. As per studies done on user viewing preference [4] [10], constant video quality is usually preferred to varying quality. Even constant or nearly constant lower quality is preferable to frequently varying higher quality.

Also, Default-ABR adaptation method generally considers the current network bandwidth to decide on the quality to be downloaded. So sometimes it can even lead to timing issue, where immediately after starting a high quality content download due to current high quality network bandwidth, suddenly the network quality changes to lower quality. As a result, high quality fragment downloading during the streaming in low network takes longer time and results in buffering. Specially while travelling or commutating, when network fluctuations are commonly felt, QoE of video streaming gets affected and buffering problems become more prominent.

Many authors have conducted experiments to improve user QoE while streaming. In [6], [7] and [9] authors have suggested buffer-based approach to enhance Default-ABR but initially user has to watch low quality video even if one is on good network. In [8], authors have shown new algorithm based on both rate and buffer occupancy. But, none of them has considered previous network trend in their studies. Our solution detects fluctuations and suggest optimized solution which enhances user QoE. Proposed solution performs well during stable as well as unstable network conditions.

In this paper, we propose ways to detect network fluctuations and an intelligent network aware adaptive streaming method (Optimized-ABR) by which we can solve the above shortcomings and achieve better QoE for the end-user during continuous network fluctuation. A simple algorithm is devised to detect network fluctuations using buffer occupancy and number of quality switches. A smart queue based solution is given to handle these fluctuations. This approach results in less buffering, less number of quality switches as well as ability to go back to Default-ABR technique in case of resumption of constant or near constant network availability.

The rest of the paper is organized as follows. Section II demonstrates our methodology to recognize network fluctuation and then based on the recognition how intelligently Optimized-ABR method can reduce quality switches as well as reduce buffering in detail. Section IV provides the experimental results on using this technique in real-network and then comparing with the Default-ABR technique result. Section V concludes the paper with further research plan on this technique.

II. PROPOSED METHOD

In this section, we present ways to detect and solve network fluctuations. In order to understand further subsections, firstly we discuss about the basic queue model being used in ABR streaming. In order to ensure smooth playback, a streaming client adds a buffering capability to absorb the effect of temporary mismatch between download rate and consumption rate. This buffering capability on client side is implemented using queue data structure. Number of downloaded data bytes in the queue will be more during peaks i.e. high network bandwidth and will drop significantly during low network bandwidth. Queue dynamics govern the loading time and video buffering. The below sub sections describe how the queueing process works in general and how our method enhances it for better QoE.

Fig. 3 shows the queue model where D(r) = Downloading rate, C(r) = Consumption rate; and B(r) = Buffered data. Initially, during start-up, when the queue is empty, we consider the Low Percent (LP) is reached. Again when data is getting received in queue via downloading, it keeps on posting buffering messages until High Percent (HP) is reached. The buffering support provided by queue is decided by these LP and HP values. So whenever LP is reached, playback moves to paused state and starts buffering to download more data for starting the playback. Again subsequently when HP is reached, a buffering message with 100% will be posted, which instructs the application to resume the playback. In other words, LP controls the buffering initiation point and HP, its termination point. HP also controls the start-up delay. Start-up delay is directly proportional to HP.

Suppose the current queue data be "$\beta_{current}$", maximum queue data be "β_{max}"

- If ($\beta_{current}$*100/ β_{max}) $\leq$ LP; buffering starts
- If ($\beta_{current}$*100/ β_{max}) $\geq$ HP; buffering ends

Having β_{max} same in both the high and low resolution components would clearly result in latter having more cached video data. Cache size variation for video fragment downloaded in different resolutions can be seen in Table I. For data collection, Gstreamer media framework [11] was used for determining fragment size and duration. The above buffering conditions will be used in subsequent section to detect the fluctuations on the network.

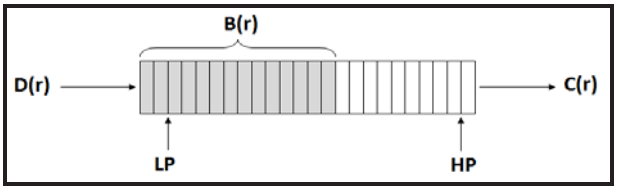


Fig. 3. Video Queuing Model

TABLE I. CACHE SIZE VARIATION WITH FIXED QUEUE.

Dailymotion Site content (fragment duration ~10 sec)			
Content Quality	***Fragment Size***	***Video Queue Size***	***Video Cache Duration***
Low	211 KB	5 MB	~24 sec
Medium	358 KB	5 MB	~14 sec
High-Medium	578 KB	5 MB	~9 sec
High	1.26 MB	5 MB	~4 sec

A. How to recognize network fluctuation ?

Considering the queue Fig. 3

- If $D_{(r)} > C_{(r)}$, then $B_{(r)}$ will keep on increasing till β_{max} is reached.
- If $D_{(r)} < C_{(r)}$, then $B_{(r)}$ will keep on decreasing till LP and buffering starts.

This network fluctuation can be identified in 2 ways. We can either put signals at HP and LP and observe the pattern of respective signals within a fixed time interval (α); or capture the number of quality switches in a playback. We will be considering each case separately and doing the comparative analysis.

- Option 1: Signal based technique

In this technique, two signals are created during queue class initialization and emitted when LP or HP condition hits. The pattern of signals we are focusing here will be "HP followed by LP" and "LP followed by HP". We maintain the pattern occurrence count for a definite time interval (α), which can be checked against a pre-decided maximum occurrence limit. If the count surpasses the limit then we are in unstable network zone. We can further optimize it by using Lower Buffering Limit (LBL) as signal emission point. LBL is a value greater than LP that helps us in capturing the fluctuations at an early stage and hence, more effectively.

- Option 2: Quality switching based technique

Number of resolution switches is directly impacted by the network conditions. Higher the network variations more will be the number of switches. So keeping the count of quality switches can be used as a deciding factor. However, this technique is sometimes misleading. As the number of variants present increases, in good network condition, resolution switching is commonly seen in upper variants (subjected to client side implementation of switching logic) but user experience is not hugely affected.

It is challenging to identify the valid quality switches. For example, if there are total 10 variants, considering user visual impact, switching between lowest and 2nd lowest variants would not be much different. But variation between 1st and 4th lowest variant maybe easily identifiable. We have solved this problem by grouping the variants and not increasing the count if switching is happening in the same group of variants. Basically, grouping is a simple methodology to partition all available quality variants based on the content encoding parameters. This helps us in logically enforce quality switching among different quality partitions to achieve good results. Here, for our experimental purposes we have divided the variants in three groups namely low, medium and high. There can be cases where numbers of groups are equal to number of variants. So to know the optimum methodology to confirm about network fluctuation, both Option-1 and modified Option-2 is used in conjunction to achieve the desired result. Algorithm 1 shows the simple algorithm for determining network fluctuations.

B. Proposed Optimization Technique

In this section, we will present the algorithm for solving network fluctuation issues. Default-ABR methodology uses the current downloading rate to estimate the next segment bandwidth value. But during network fluctuations, this approach does not provide best user QoE results as the current conditions may not reflect the upcoming network problems. Inaccurate estimates will lead to re-buffering events and frequent quality switches. To overcome this drawback in default ABR methodology; we propose a different client-side bitrate adaptation approach.

Algorithm will be devised in such a way that if the network is stable during the course of playback, Default-ABR or bandwidth based approach will be used otherwise we will switch to our proposed solution. Once network stabilizes, we will come back to default setting i.e. using bandwidth as the deciding factor.

Optimized-ABR approach:

In proposed solution, we will use buffer occupancy to decide the next segment bandwidth. The idea behind this is to reduce buffering count caused due to inaccurate bandwidth estimates during fluctuating network conditions. We will divide the variants according to section III (Option 2) where we have discussed about the importance of grouping variants. Qualities are grouped on the basis of their switching impact on user QoE i.e. if switching is happening within the qualities of same group, the impact on user experience will be less. Moreover, we can select the lowest bitrate quality amongst the variants of same group. Further, we will partition the queue in same number of parts as that

Algorithm 1 Network fluctuation determination

Input: $Signal_{prev}$: Previously emitted signal
$Signal_{now}$: Currently emitted signal
$Quality_{prev}$: Previously played video quality
$Quality_{now}$: Current video quality
Output: Network fluctuations found.

```
Divide all variants in different groups.
Start Timer for α time.
Signal_prev ← LBL
repeat
  repeat
    if Signal_now ≠ Signal_prev then
      Signal_prev ← Signal_now
      Signal_counter ← Signal_counter + 1
      if Signal_counter ≥ maxval then
        return Network fluctuation found.
      end if
    end if
    if Quality_now ≠ Quality_prev then
      Quality_prev ← Quality_now
      Quality_counter ← Quality_counter + 1
      if Quality_counter ≥ maxval then
        return Network fluctuation found.
      end if
    end if
  until Timer expires.
  Restart Timer.
until playback ends.
```

of number of quality groups. For instance, we have taken three groups say low, medium and high. The corresponding partitioned queue is shown in Fig. 4. First part of the queue will be filled with segments from lower video quality group, second part with medium quality and when data inside queue surpasses the second region i.e. when we have enough data to sustain, we will switch back to Default-ABR approach. There is a trade-off between rebuffering and playing high quality. So, more the size of low and medium queue parts, lesser will be the re-buffering events. Optimized-ABR aims to improve the user QoE factors such as buffering frequency and video quality switching frequency.

Our proposed technique selects lower bandwidth segments when the chances of queue getting drain are higher, thus, reducing the possibility of buffering.

To address the issue of frequent quality switches, we enforce video quality to remain in a particular group for a longer duration. Thus, instant jumps from one quality group to another are minimized. Optimized-ABR approach aims at caching more data during high network bandwidth by

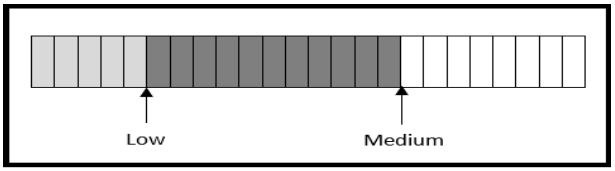


Fig. 4 Queue during Optimized-ABR (3 groups)

lowering down the quality so that we can sustain the bad network intervals.

To illustrate it further, let us consider the example given in Table II. Based on Algorithm 1, we can identify the network fluctuation. Let us assume, at the current stage, we are getting good bandwidth for download.

Let the current high Download rate be $D_{(r/high)}$. So in bandwidth based approach, at this point, good network speed will be utilized in downloading high quality video, So it will take $\Lambda_{(t/high)}$ sec to download a fragment, where,

$$\Lambda_{(t/high)} = 1.26 \text{ MB} / D_{(r/high)}.$$

But, as per our proposed optimization technique, we are forcing the download to medium Quality group, even when $D_{(r/high)}$ is maintained same. So in our proposed methodology, fragment download will take only $\Lambda_{(t/high-medium)}$ time, where,

$$\Lambda_{(t/high-\ medium)} = 578 \text{ KB} / D_{(r/high)}.$$

Now, let's assume, due to network fluctuation, after T seconds, network download rate will suddenly drop to very low value.

So the number of video fragments which would have cached in default ABR method would be $N_{(Default-ABR)}$, where,

$$N_{(Default-ABR)} = T / \Lambda_{(t/high)}.$$

But with the proposed methodology, it would be, $N_{(Optimized-ABR)}$, where,

$$N_{(Optimized-ABR)} = T / \Lambda_{(t/high-\ medium)}.$$

Now since $(\Lambda_{(t/high)} >> \Lambda_{(t/high-\ medium)})$, so $(N_{(Default-ABR)} << N_{(Optimized-ABR)})$, i.e. the number of cached fragments will obviously be much more in the proposed design compared to default design. This will result in less buffering as more cache would help in sustaining in the bad network portion of fluctuation.

C. Why to disable the Optimization Technique ?

In previous sub section, we have proposed an optimization technique which helps in improving user QoE when user is in fluctuating network condition. But when normalcy has returned to the network condition, if we won't disable our logic, it will keep on downloading lower resolution fragment even when user is getting high

downloading speed. So disabling the logic is of equal importance as implementing it. Normalcy of network can be understood as same logic mentioned in Section II (B), when buffer occupancy surpasses the medium queue region. This marks as the point, the queue returns to the Default-ABR methodology.

III. Experimental Results

For our experimental purpose, we have used two smartphones of same model having same hardware specifications (Tizen OS having Quad-core Spreadtrum 1.5 GHz Processor having WVGA Resolution). One device was working on Default-ABR mechanism and other on Optimized-ABR. For streaming content, we have used Dailymotion, Hotstar and Jio applications and have conducted many experiments playing different video

TABLE II. Content Size and Resolution Variations.

Dailymotion Site content (fragment duration ~10 sec)				
Content Quality	*Fragment Size*	*Video Resolution*	*Overall Bitrate*	*Video Encoder Settings*
Low	211 KB	320x284	173 kb/s	AVC(Baseline @L1.3) , 5 Ref frames
Medium	358 KB	512x288	294 kb/s	AVC(Baseline @L2.2) , 3 Ref frames
High-Medium	578 KB	848x480	476 kb/s	AVC(High@L3 .1) , 4 Ref frames , CABAC
High	1.26 MB	1280x720	1064 kb/s	AVC(High@L3 .1) , 5 Ref frames , CABAC

contents but same video on both devices at a time.

In order to capture the real-time network fluctuations in real field testing, the devices were taken on a moving vehicle, both playing same adaptive bitrate content and working on same network operator. The study was conducted using multiple service providers to cover the various network conditions and results are demonstrated for Reliance JIO and Bharti Airtel networks. The comparative analysis was done for both the devices.

Fig. 5 and Fig. 6 depict the results for Airtel and Reliance respectively; where Part (A) shows the network bandwidth trend over time (IST). Part (B) and (C), show the duration for which buffering was noticed for devices without solution and with solution respectively. Part (D) demonstrates the advantage of Optimized-ABR over Default-ABR.

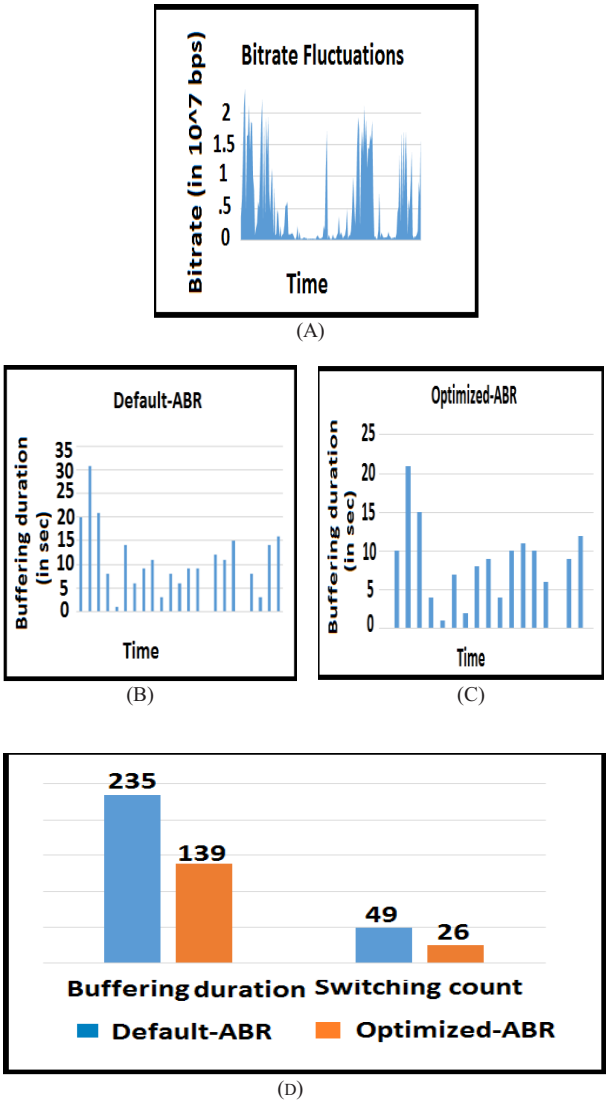
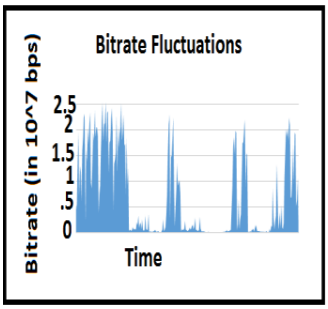


Fig. 5 Experimental Result-1 with Airtel

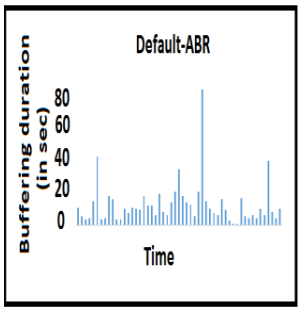
Experiment-1 uses 30 and 60 as the low and medium partition values and Experiment-2 used 50 and 80 respectively. In Experiment-1, the buffering duration reduction seen is around 40% and around 47% for quality switch count.

For Experiment 2, the values are approximately 42% and 50% respectively. This demonstrate that the buffering time and buffering count seen in Default-ABR is more than Optimized-ABR. Moreover, the quality switch count has also been reduced by a significant amount.

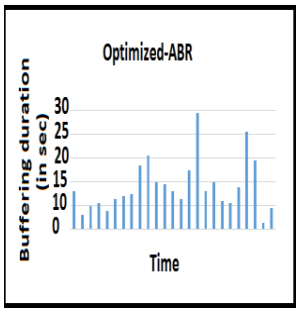
On doing the collaborative analysis, the improvement in re-buffering events varies from 20-60% and switching frequency varies from 40-50%. Hence, with our proposed solution the values of buffering time, buffering count and switching count are decreasing.



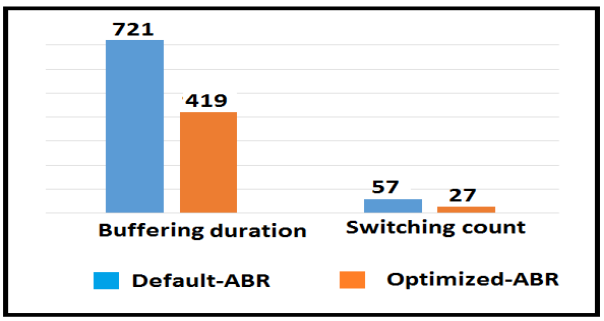
(A)



(B)



(C)



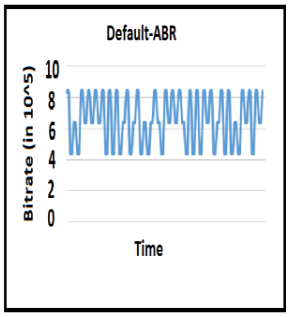
(D)

Fig. 6 Experimental Result-2 with Reliance JIO

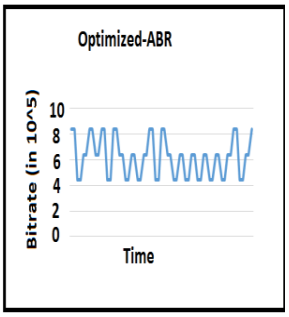
IV. Conclusion And Future Work

In this paper, we introduced an Optimized-ABR algorithm which helps in improving user QoE during fluctuating network conditions over current state-of-the- art Default-ABR methodology. We have proposed ways of knowing network fluctuations by using LP and HP parameters and by using quality switch count. We further formulated methods to handle the fluctuation manipulating queue parameters by dividing queue in various groups and forcing video quality to relatively lower variants. At last, two experimental results are shown to demonstrate the effectiveness of our proposed solution.

In summary, compared to existing state-of-the- art Default-ABR methodology, current proposed Optimized-ABR methodology has resulted in reduction of buffering event occurring by almost 40% as well as switch count



(A)



(B)

Algorithm \ Bitrate	High Bitrate	Medium Bitrate	Low Bitrate
Optimized-ABR	20m 49sec	4m 43sec	31m10sec
Default-ABR	26m 27sec	3m19sec	26m56sec

C) Comparison between 'A' and 'B' w.r.t. different Quality playback duration

Fig. 7 Different Quality playback duration comparison

reduction of around 45% respectively on average.

It can easily be inferred from the results, with this technique being employed, User QoE would be enhanced up to a good extent as there is significant reduction in buffering time as well as quality switching frequency.

Now coming to the future work scope, Fig. 7 shows the video rate played during Experiment-2 and below table depicts the total time for which a particular quality has been played. High bitrate content is played for more time in Default-ABR than in Optimized-ABR. So there is a definite scope of improvement for optimizing the optimized methodology further.

Future research plan includes using neural networks to get better user QoE in every aspect.

Acknowledgment

I would like to thank Dr. Narasinga Rao Miniskar for his guidance while reviewing this paper.

References

[1] D. Karamshuk, N. Sastry, A. Secker, and J. Chandaria, "On factors affecting the usage and adoption of a nation-wide TV streaming service," in Proc. IEEE INFOCOM, Apr./May 2015, pp. 837–845

[2] https://static.mux.com/downloads/2017-Video-Streaming-Perceptions-Report.pdf

[3] https://mux.com/blog/buffering-reduces-video-watch-time-by-40-according-to-research/

[4] M.-N Garcia, F. De Simone, S. Tavakoli, N. Staelens, S. Egger, K. Brunnstr¨om, A. Raake, "QUALITY OF EXPERIENCE AND HTTP ADAPTIVE STREAMING: A REVIEW OF SUBJECTIVE STUDIES", in Quality of Multimedia Experience (QoMEX), 2014

[5] https://gstreamer.freedesktop.org/documentation/application-development/advanced/buffering.html

[6] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson,“A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service,” in Proc. ACM Conf. on SIGCOMM, August 2014.

[7] H. Le, D. Nguyen, N. Ngoc, A. Pham, and T. C. Thang, "Buffer-based Bitrate Adaptation for Adaptive HTTP Streaming," Proc. of the IEEE international Conference on Advanced Technologies for Communications, pp. 33-38, October 2013.

[8] H. T. Le, H. N. Nguyen, N. P. Ngoc, A. T. Pham, and T. C. Thang, “A Novel AdaptationMethod for HTTP Streaming of VBR Videos over Mobile Networks,” MobileInformation Systems, vol. 2016, Article ID 2920850, 2016.

[9] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, “A control-theoretic approach for dynamic adaptive video streaming over HTTP,” in Proc. ACM SIGCOMM, 2015.

[10] M. Seufert, T. Hoßfeld, and C. Sieber, “Impact of Intermediate Layer on Quality of Experience of HTTP Adaptive Streaming,” in 11th Intl. Conference on Network and Service Management (CNSM), Barcelona, Spain, 2015.

[11] “GStreamer multimedia framework.” http://gstreamer.freedesktop.org/