ArunKumar and Gunaseelan *EURASIP Journal on Image and Video Processing* (2018) 2018:17
https://doi.org/10.1186/s13640-018-0256-z

EURASIP Journal on Image and Video Processing

RESEARCH Open Access

Optimized buffer allocation for video multicasting applications with virtual memory implementation

A. ArunKumar* and K. Gunaseelan

Abstract

Memory requirement is a key issue when servicing more number of nodes in a heterogeneous environment. We have to ensure an optimum buffer size in order to reduce the initial latency. In this paper, we propose a novel approach, which involves, augmenting a virtual memory (VM) with the existing physical memory of the video buffer. In addition to the benefits of improved queuing size, it also paves the way for better quantization, ordering of frames, and higher error resiliency. Instead of allocating the virtual memory as a single file, it is fragmented according to the number of nodes in a specific multicast group. This avoids the flushing of an entire buffer to begin a new streaming. Furthermore, this avoids the requirement of proxy servers for each multicast group. Hence, this reduces the burden of the server, when a new user joins the multicast session later, requesting same content. VM also facilitates a platform to support old and advanced graphical interface irrespective of device capability. A bit rate reduction technique is also applied to video streams which enables seamless streaming even in degraded network. Unlike existing buffer adaptation techniques, this method does not require a feedback loop, as most of the rate adaptation is processed by the VM. Extensive simulation results show that this research work helps in significant improvement in throughput, PSNR (peak signal-to-noise ratio) when compared to existing buffer management algorithms such as RBA (rate-based adaptation), BBA (buffer-based adaptation), Elastic, and FESTIVE (Fair, Efficient, and Stable adapTIVE algorithm).

Keywords: Optimized buffer allocation, Virtual storage space, Video multicasting, Bit rate reduction

1 Introduction

Video communication has attained rapid growth with the advancement in digital compression and network speed which has lead to more high-quality video streams to be delivered across the globe. Tons of video is streamed by the internet every nanosecond. Generations have moved forward from just streaming to interactive operations such as video conferencing, webinar, multi player gaming (such as Twitch tv and YouTube gaming), and live streaming from mobile phones into social network. However, these technologies can be effectively presented to the user only if the network conditions are stable. Current wireless network is always unpredictable because once an eNodeB advertises to join a multicast group, the number of users hitting the memory of the server is unknown. AMP (adaptive media playout) which is found in literature [1] is used to compensate for bit rate fluctuations which was developed on the basis of underflow estimation and the buffer allocation in VOD (video-on-demand) systems [2] which predicts the size of the current buffer based on future predictions. Likewise, scheduling methods such as round-robin, sweep method, and GSS (Global Site Selector) reduce the memory requirement by allocating resources based on estimation of minimum memory requirement and number of users. This is not feasible because the number of users vary from time to time and more the number of users the lesser the bandwidth available to each user. ADT (adaptive data transmission) described in [3] transmits the video in bursts. However, this requires the buffer information to be collected from the user and updated periodically which wastes the resource during down time of the network.

In most cases, pre-roll technique has been employed in which the initial part of the video is delayed until the

*Correspondence: nurak23@gmail.com
Department of Electronics and Communication Engineering, CEG, Anna University, Chennai, India

buffer is partially loaded and a feedback message is sent once the buffer reaches the threshold, and based upon this information, the server tunes the stream and estimates the further transmission. The video processor at times becomes overloaded due to multiple stream request owing to stochastic network. Hence, the need for additional memory arises to incorporate look-up table, estimation calculations until the video is fully streamed. Initially, the server streams a video to a multicast group in few different bit rates. Based on the capability of the UE (user equipment), it settles for one of the streams as explained in [4]. It provides an alternate way of reducing the delay by selecting frames from the pre-encoded video stream and tailoring it on the fly. However, it is done based on the assumptions of optimal frame selection which is confined to a particular window size. QoE (quality of experience) or user satisfaction is an important utility function which measures the quality of video on a MOS (Mean Opinion Score) scale which is a linear mapping against PSNR (peak signal-to-noise ratio) measured in decibel. DASH (dynamic adaptive streaming over hyper text transfer protocol) [5] is the current streaming protocol implemented by various internet video networks such as YouTube. QoE-driven adaptive HTTP media delivery provides adaptation at the server section while the client is unaware of the changes and plays the media content. Here, the eNodeB is unaware about the media content, whereas the scheduler is content aware. It generates several representations of each video requested by the client at different quantization parameters to adapt to the network conditions. Our proposed algorithm schedules and reschedules on the fly by expanding/compressing the buffer so that the entire content is streamed in the midst of varying network conditions.

The rest of this paper is organized as follows: Section 2 provides related work on rate adaptation and buffer management. Section 3 presents detailed problem description and video parameters. Section 4 presents the methods by which we tackle the problem. Computer simulation and evaluation is discussed in Section 5. Section 6 concludes the paper.

2 Related work

Rate adaptation in video technology is an unending research topic, where new ideas are proposed and implemented across different platforms of video streaming applications. In this section, we give a brief overview of the pitfalls of existing approaches and how our approach differs from it.

2.1 Single stream approach

Bit rate adaptation is achieved in [5] using transcoding technique, where the encoded stream available at the server is decoded and re-encoded to match the new bit rate requirement of the user. Transcoding is a useful tool in switching from one network to another [6]. The pitfall is that it increases the system overhead, and parallel computation mechanism is required for varying the bit rate and it is tedious for a constantly switching network.

2.2 Simulcast approach

The video is pre-encoded into different bit rates, and the user settles in for a suitable rate at which it can receive the video to its maximum capability [7]. However, this approach does not address the problem of fluctuating bandwidths and the hand-off mechanisms has to be employed as stated in [8] which adds to the burden of network.

2.3 Frame dropping

As the trade-off factor between network bandwidth and number of users reaches a particular threshold level, frames are dropped in a particular order like first B-frames (bi-directional frames), then P-frame (previous frame), and at last I-frame (independent frame). But dropping of P- and I-frames may cause the decoder to struggle in reconstructing the video with integrity, [9, 10] follows this approach of frame dropping after frame identification.

2.4 Memory requirement

Proxy servers are being used to reduce the burden of streaming server, and a portion of the video is partially stored and streamed from the proxy server [11] and this is also called as video staging [12]. Though this eliminates the initial buffering delay as video is resourced quickly from a nearby source, the continuity of the remaining video is not guaranteed under varying bandwidth. It is also done under the assumption that the user watches the entire video from the start to end. However, in practical cases, the user may skip the initial part and move towards a favorite clipping in the video and under this case, the delivery rate is the same as that without proxy.

2.5 Multicasting in LTE

LTE (Long Term Evolution Networks) allow multicasting using eMBMS (evolved multimedia broadcast multicast service). However, the device hardware should be eMBMS compatible and software upgrade is necessary. Also, it needs BM-SC (broadcast/multicast service center) to control the servers [13]. Moreover, only 60% of subframes per frame is eMBMS compatible, and it restricts the number of transmission blocks associated with the H.264/SVC (Scalable Video Coding) video layer [14].

2.6 Buffer adaptation algorithms

HAS (HTTP-based adaptive streaming) [15] uses BBA (bit rate adaptation) [16] and rate-based adaptation [17], where the former mandates the bit rate based on current

occupancy of playout buffer, whereas the later uses a-priori information in selecting the bit rate. There are no other specific bit rate selection criteria to optimize the buffer. Elastic buffers [18] use a proxy storage in between the server and the client, inorder to split the streaming load. FESTIVE (Fair Efficient and Stable adapTIVE) algorithm [19] is again a HTTP streaming method which uses immediate scheduling and randomized scheduling for buffer adaptation in small area networks.

3 Problem definition

Consider a live stream of various pre-encoded streams with rate R_i. Let B_i be the bandwidth of users connected to the transmitting server. When the number of users U_i increases, streaming speed decreases because the rate at which each frame is decoded solely depends upon packet arrival rate. The rate $R_i(U_i)$ at which each user receives the packet will not be practically equal due to factors such as equipment type, position of the user, and demand of user. During this period, buffering gets worse because of slow reception of packets and packet loss, which occurs due to severe congestion in the network. Optimal video quality algorithms available in the literature will receive only the base layer for which the user has to compensate in terms of quality. To stream a video, more or less the same amount of bandwidth is consumed. However, the time taken to stream a full video is larger compared to downloading the entire video. As the buffer gets filled up, it is flushed out once the frames are displayed by the media player. The AVI (Audio Video Interleave) format used in the internet uses 400 Mb whereas MPEG4 uses 100 Mb only. The type of compression used plays a vital role in terms of deployment cost, availability, and QoE (quality of experience).

We develop a virtual memory space in the inactive memory of hard disk, which acts as a virtual buffer. The need for this eliminates deleting of streamed content and paves the way to save it by the user for later viewing. This virtual buffer produces the flexibility to stream multiple videos from the same or different servers in a parallel fashion. Also, this gives the time to retransmit the lost packets so that the content available is exactly similar to the download copy. We use a stochastic addressing method to transmit the packets from virtual buffer to the playout buffer with maximum hit rates under various conditions. The virtual to reality mapping ($V = \mathrm{R}$) gives the user an option to store the stream to an external drive via USB (universal serial bus). However, subscription parameters can be applied to avoid illegal copyrights. We also show that how virtual buffer is effectively created and utilized when the user requests to stream a video of length L and the CBR (constant bit rate) traffic is modeled using Pareto distribution which shows, how 20% of streamed data is used to achieve the efficiency of 80% of the bandwidth needed to download the same. The streamed video is saved in the fragmented buffer; thus, it decreases the loading of network while downloading it. This avoids user consuming more bandwidth by downloading or requesting already viewed content to be streamed again. The progressive download (streamed content) is stored in our virtual memory; hence, those packets are skipped while making a download. Downloading a video is the fastest way to fetch the data from the server than by streaming the entire data online. This reduces the per user time slot in a multicasting domain and achieves high-speed content delivery.

3.1 Frame rate

The frame rate determines how often the image will be updated; fps (frames per second) and thus scales video conferencing quality in terms of display quality, when an object moves too fast. Video systems can often be configured to use a higher pixel resolution by dropping the frame rate. Although this trade-off can work well for static images, it degrades quality for dynamic images of moving objects. Slow frame rates deliver jerky video images that detract the conference experience. So if bandwidth is limited, it often makes sense to ensure a full 30 fps, even at the expense of some resolution.

3.2 Resolution format

Video resolution settings should also take content of video into account. But in a video conference room, resolution may need to display a range of detail. For example, less resolution is needed for seminar presentations with good viewing font sizes than for detailed microscopic video. QCIF (quarter CIF) offers better resolution than CIF (common intermediate format) but at the added cost of more number of fps and higher transport bandwidth.

3.3 Dead zone

It refers to the pause/delay in streaming, caused due to two conditions which are underflow and overflow. Usually, it is caused when the frames do not load consecutively in a chain fashion and breaks in between. This could be due to poor connection in the network.

3.3.1 Overflow

When the buffer memory is fully loaded with frames, while the server continuously sends data, the frames that miss the buffer queue are lost and the decoder simply quits those frames and moves forward to maintain the synchronization. The frames that are skipped cause black out period which causes dead zone.

3.3.2 Underflow

When a video is streamed through a bottle neck bandwidth, most of the frames are dropped and the frames

partially fill the buffer, which causes choppy video. This often occurs when a live footage is streamed during poor network conditions.

4 Methods

The following are the contributions of this paper: (1) solves the memory requirement problem and avoids proxy servers by creating a virtual memory space at the server and UE, and (2) if a B-frame is dropped during transmission, it is reproduced using a-priori information obtained from previous and next frame using variance window.

4.1 Calculation of buffer size

The server establishes its own buffer queue size B_S every "t" seconds depending upon the mean buffer size of the number of clients connected to it by

$$B_s = I\left[(S_i) * S_{im}(t_0)\right]^{N_c} \tag{1}$$

where I is an integer, S_i is the number of streams, S_{im} is the number of multicast sessions, t_0 is the initial time, and N_c is the number of clients authenticated at the initial moment.

4.2 Calculation of frame size and bit rate

The frame size F_S is given by

$$F_S = Res \times C_D \tag{2}$$

where *Res* is the maximum resolution of the display screen of the user and C_D is the color depth of the display.

The bit rate B_R is given by

$$B_R = F_S \times F_R \tag{3}$$

The buffer size of each user equipment is calculated as follows

$$B_{UE} = B_R \left[\sum_{t=0}^{T} \frac{1}{F_s * B_t}\right]^2 \tag{4}$$

B_R is the incoming bit rate, F_s is the average frame size of the incoming stream, and B_t is the transport bandwidth and it varies with time t; T is the average time to playout the incoming stream. Now, the sender checks for any new clients in its multicast group and ensures if they can be served efficiently with its available queue size. In general, queue size is larger when the bandwidth provided by the ISP (Internet service provider) is good enough and the number of clients sharing that particular network is minimum. When the network conditions become worse, current technologies simply adjust the MCS (modulation and coding scheme) and execute a buffer adaptation algorithm. This paper introduces a concept of segregating video, audio, and text in the virtual memory space and sending only efficient frames to the encoder. Similarly, at the receiver side, the process of rebuilding these frames is done in the virtual memory space and then passed to the decoder for further extraction. The benefits of this approach is two-fold; firstly, it reduces the burden of encoder and decoder during poor network conditions. Secondly, it improves the QoE of the user by providing better performance than existing adaptation schemes.

4.3 Authentication

The eNodeB advertises the available multicast services and all interested UEs which are authenticated (content based, location based, subscription based) within the cell to join this service and forms unique multicast groups.

4.4 UEI harvest

The UEI (user equipment's useful information), such as software version, maximum data rate, buffer queue size, are harvested from individual members of each multicast group.

4.5 Setting up of B_s

Now, we calculate the mean buffer queue size of each member of the multicast group. The queue size is based upon the data rate available at the user.

4.6 Updating B_s

As the number of users increases, the B_s is updated accordingly. During a multicast session, clients join/leave the group randomly. Apart from TCP/UDP (Transfer Control Protocol/ User Datagram Protocol) file handler, other systems deploy a continuous feedback to monitor both the connection state and side information in recovering the dropped frames. We do not use such information retrieval feedback systems, as the VM intelligently drops and retrieves such frames during bottleneck bandwidth.

4.7 Clustering

Video, audio, and text files are clustered into unique compartments into the virtual memory space.

4.8 Fragmentation

Inside each compartment, the individual files are fragmented to utilize the available space efficiently.

4.9 Request rate

During network congestion, the number of hits in the virtual memory will skip variance values from 0 to $V_{\min}$ (i.e., minimum variance). This is done because zero variance indicates similar values, and it is the repetition of frames of same scene contributing to a 30-fps shot.

4.10 Encoding

The useful base layer information extracted from the above process is encoded satisfying the alpha condition.

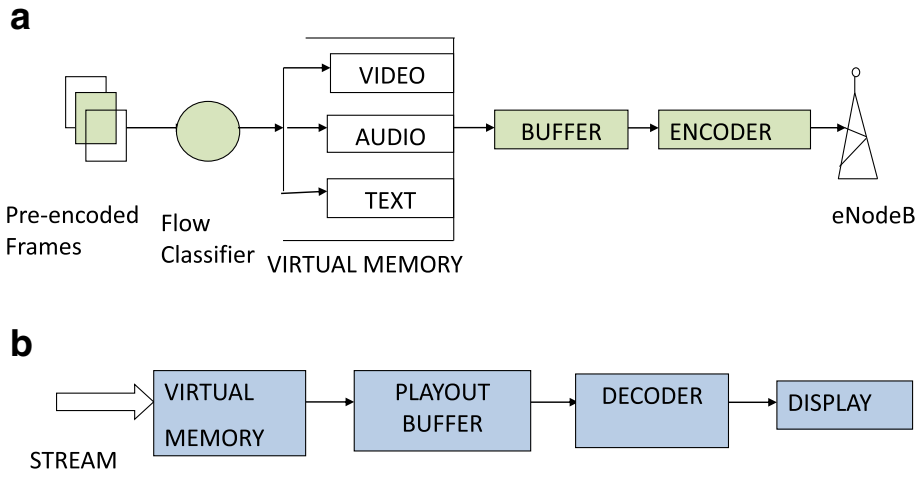

Fig. 1 VM architecture at both server and client. **a** Server section, **b** UE section

4.11 Decoding

The zero variance received is decoded into dummy frames as generated by stereoscopic decoding for motion compensation. This is then reconstructed by Wyner-Ziv method using hash codes.

4.12 Buffer overflow/underflow avoidance

As the virtual memory stores the packets in a cluster and fragmented fashion, and since it has the ability to dynamically increase the memory space, a lengthy movie can be streamed without buffer overflow/underflow if it satisfies the following condition, where S is the entire segment of the video, $B_{UE}(i)$ is the bandwidth of individual user equipment, and B_S is the buffer size of the server.

$$B_{UE}(i) \leq S \leq B_S \tag{5}$$

Figure 1 shows the design at both the server and client side. The pre-encoded frames of a video multicast session is segregated as video, audio, and text using a flow classifier in the server side virtual memory. This is done in order to avoid sync problems, if a frame is lost. All these frames are addressed before passing them to the P-BUFF (Physical BUFFer), thus making the encoding process quite easier. These encoded packets are streamed through eNodeB.

Similarly, when the user streams the video in a multicast session, all the reverse process is carried out before reaching the decoder buffer (otherwise called as playout buffer). The decoded content is then displayed after satisfying the buffer fullness conditions which includes the ordering of frames as numbered during the encoding process. Thus, the virtual memory acts as a platform for arranging the frames received during retransmission of lost packets and sees to it that the playout buffer never feeds the decoder with dead zone.

4.13 Significance of VM

Appending a virtual memory at the server and at the client's range not only increases the buffer size but also contributes to the parallel streaming of multicast sessions, which in general requires complex storage systems and switching. A small-scale multicasting service provider will now be exempted from investing in building huge storage rooms and routing them to multicast different streams. Let us visualize the concept as follows.

4.13.1 Dedicated multicast buffers

Now, it is possible to achieve the concept of having DBM (dedicated multicast buffers). Thanks to the enhanced virtual memory architecture which uses memory mapping as a key to reduce software overheads and thereby improving the overall bandwidth of the system. Streams S_i of multicast sessions S_{im} where i ranges from 1,2...N, which indicates each stream can be of infinite length, within a scalable number of multicast sessions of finite length as expressed in Eq. (5).

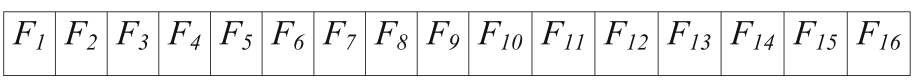

Fig. 2 Frame structure with GOP = 15

Table 1 Inference of different bit rate reduction and frame reconstruction techniques

Stream	Method	BR (Bps)	PSNR (dB)	VAR (S_i)	(SD)	BUFF_DELAY(s)
Foreman	IC	698754	34.07	0.52	0.72	1.07
	WZ	659738	35.68	0.65	0.80	0.39
	VM+WZ	634428	38.45	0.99	0.99	0.01
Stefan	IC	397642	32.65	0.26	0.5	1.68
	WZ	349893	36.43	0.34	0.58	0.12
	VM+WZ	325331	39.01	0.51	0.71	0.1
News	IC	743215	37.95	0.86	0.92	2.27
	WZ	717554	39.76	0.96	0.97	0.1
	VM+WZ	691266	41.2	1.2	1.09	0.02

4.13.2 Custom initialization

In a generalized method of initiating a process, the buffer state is set to zero and advertises that it is unoccupied. This occurs before beginning a multicast session and also occurs when a web page is refreshed. But this algorithm uses an epic custom initialization, where each buffer state of previous multicast group is displayed and they can be manually flushed out. The pre-occupied state is then set so that the coder (encoder/decoder) does not call the transcoding process once again. This is not the case in existing systems which may use a cache memory.

4.13.3 Buffer establishment

The unoccupied buffers which are listed gets addressed by the pre-encoded frames at the server and by the encoded stream at the client. Further, additional buffers are established in the consecutive multicast sessions. This is given by Eq. (1). There can be many such buffers, depending on the streaming content, its length, and other streaming parameters such as bit rate and frame rate.

4.13.4 Self adaptation

Unlike other buffer adaptation algorithms, which require a feedback loop and forces the server to tune in accordingly, the proposed system is self adaptable. It allocates each buffer with optimal capacity. This paves the way to stream multiple streaming sessions in a multicast hub and avoids feedback implosion problem.

4.13.5 Delay compensation

SVC (Scalable Video Coding) or DASH is used in current systems for bit rate adaptation. SVC produces a base layer and few enhancement layers to adapt to the bit rate of the client. Whereas in MPEG (Moving Pictures Expert Group) DASH, the video is broken down into small packets and re-encoded with multiple bit rate options to

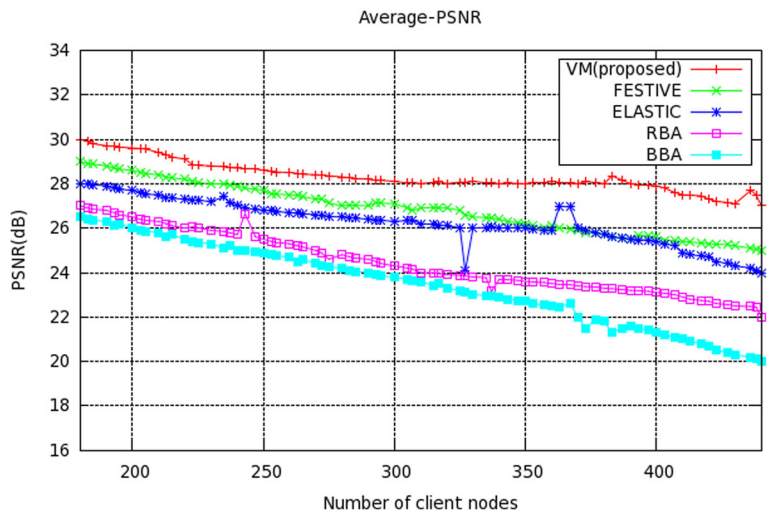

Fig. 3 Average PSNR of buffer management algorithms

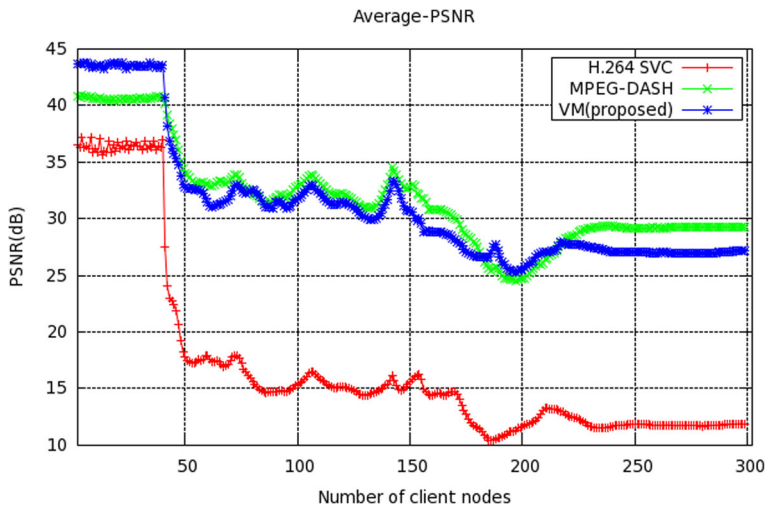

Fig. 4 Average PSNR of bit rate reduction techniques

sync with the UE of the client. These are then strung in together to form a consecutive video stream with variable bit rates.

However, the user has no control over the quality of the video because if at time t_1 the quality is HD (high definition), later, when more clients join the multicast group, the quality steps down to SD (standard definition) at time t_2. Moreover, while streaming a HD content, the cache memory gets refreshed at equal intervals to fill it with the forth coming streams. Hence, there is no assurance for the user to revisit the initial part of the displayed video at a later time without any buffering. This highly relies on the present bandwidth available at that particular time slot. This paper uses the traditional method of bit rate control, by dropping the frames by the boundary conditions as stated in our algorithm and in addition uses enhanced WZ (Wyner-Ziv) coding to reconstruct the dropped frames more precisely and also to provide a better compensation for delay induced in reconstructing those frames.

To reduce the bit rate, certain frames are dropped based on the decision of our algorithm and they are reproduced using Wyner-Ziv coding with default hash and variance methods, to produce high accuracy.

Consider the following example as shown in Fig. 2, IBBBBPBBBBPBBBBI here GOP (group of pixels) = 15, $M = 5$, and $N = 16$, where M is the distance between two anchor frames (I or P) and N is the distance between two I-frames. Here, all the B-frames need not be transmitted

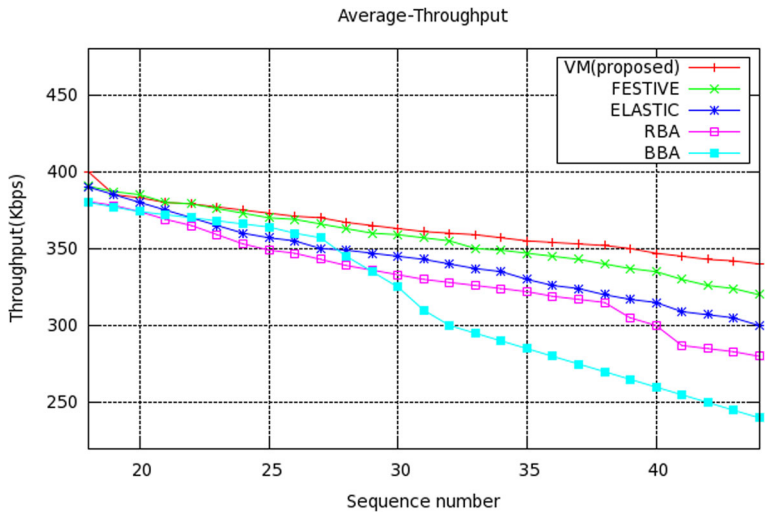

Fig. 5 Average throughput of buffer management algorithms

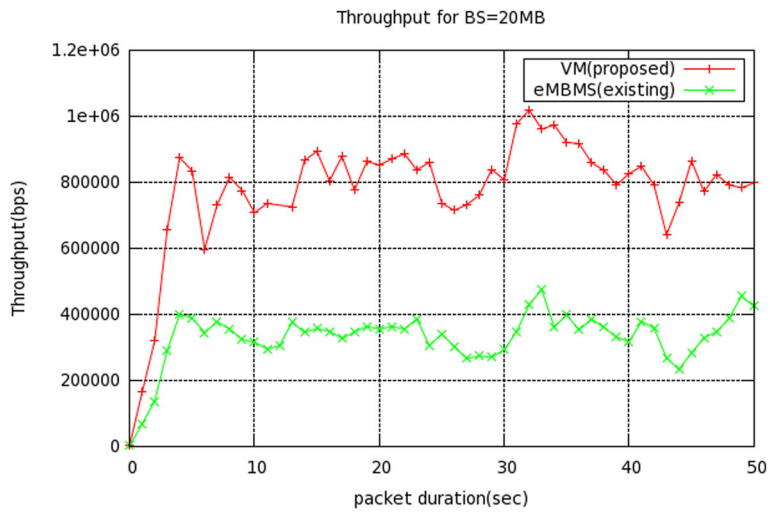

Fig. 6 Average throughput for a fixed buffer size of 20 mb

and they can be correctly reconstructed with zero variance information.

Here, F_1S_i will always be the I-frame which corresponds to the first slot of memory address which consumes a space in the virtual memory $\alpha > 0$, where α is the video shape parameter. The next occurrence of P-frame is calculated from the M value which will be $F_{M+1}S_i$, and it corresponds to the 6th slot of memory address with $\alpha > 0$. Now, variance is calculated to reconstruct the intermediate frames, and as we know, variance $\sigma^2 = 0$ if all the entries are the same between F_1S_i and $F_{M+1}S_i$.

Since the entire segment S of a video stream is a random variable whose value S_i is always greater than individual multicast segments S_{im}, it follows a survival function given by $F(S_i)$.

$$F(S_i) = Pr(S > S_i) = \begin{cases} \left(\frac{S_{im}}{S}\right)^{\alpha} & if\ S_i \geq S_{im} \\ 1 & if\ S_i < S_{im} \end{cases} \tag{6}$$

The mean value of a random variable of the above function is

$$\mathrm{E}(S_i) = \begin{cases} \infty & if\ \alpha \leq 1 \\ \frac{\alpha S_{im}}{\alpha - 1} & if\ \alpha > 1 \end{cases} \tag{7}$$

and the variance of the random variable is

$$\mathrm{Var}(S_i) = \begin{cases} \infty & if\ \alpha \in (1, 2] \\ \frac{S_{im}}{\alpha - 1}^2 & if\ \alpha > 2 \end{cases} \tag{8}$$

and if $\alpha \leq 1$, then $\sigma^2 = 0$.

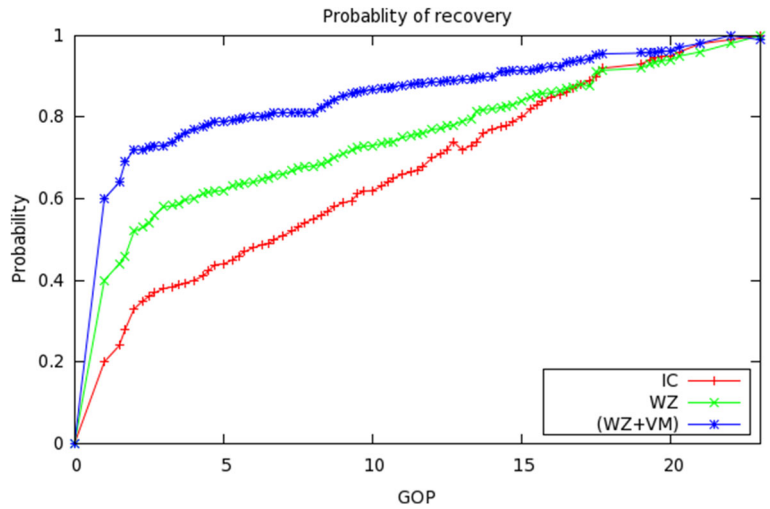

Fig. 7 Probability that a dropped frame is reconstructed effectively

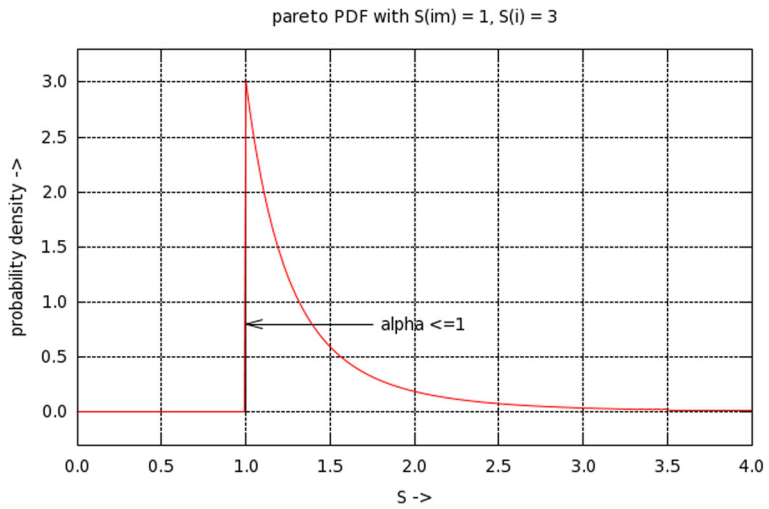

Fig. 8 PDF with $S_{im} = 1$ and $S_i = 3$

5 Results and discussions

The use of virtual memory at the server and clients along with the existing physical buffer has been implemented using NS-2.35 for demonstration purpose. In order to prove VM outperforms other systems, extensive real-time simulation was performed under different scenarios. The bit rate reduction is compared with very well-known techniques such as IC (intra-frame coding), WZ, and VM with WZ using test sequences namely foreman, stefan, and news as shown in Table 1.

5.1 Average PSNR

We analyze our proposed system with most common video quality estimation metrics; average PSNR which is shown in Fig. 3. It shows that our proposed method VM outperforms other buffer management algorithms such as FESTIVE, Elastic, RBA, and BBA. It is to be noted that even when the number of nodes is exceeding 500 count, still VM manages to provide a stable PSNR, unlike other methods which include a feedback loop. When the clients are minimum in a multicast session, all the methods have a stable control over PSNR. However, it then gradually drops during a period of time when more clients keep on winning the access to join the multicast group.

Let us now compare our proposed VM with other bit rate reduction techniques such as MPEG-DASH and H.264/SVC. The average PSNR of bit rate reduction methods is compared in Fig. 4. It is inferred that our proposed method VM provides more than 40 dB initially, when the number of clients are limited. However, it tries to stay close with that of MPEG-DASH, as the number of clients further increases.

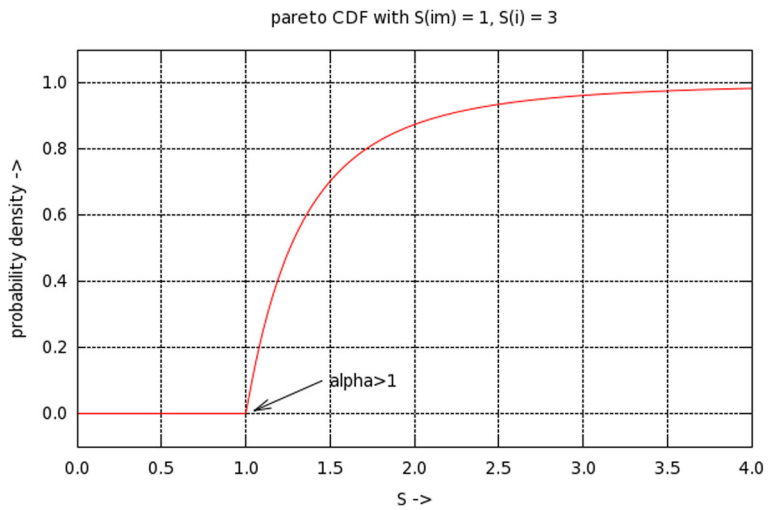

Fig. 9 CDF with $S_{im} = 1$ and $S_i = 3$

Table 2 Performance comparison with other buffer optimization algorithms

		H.264-SVC		MPEG-DASH		VM	
GoP	Stream	A-PSNR (dB)	A-throuput (Kbps)	A-PSNR (dB)	A-throuput (Kbps)	A-PSNR (dB)	A-throuput (Kbps)
8	Foreman	37.02	248	38.94	395.76	40.47	801.3
	Stefan	38.79	388.8	40.28	585	41.53	869.84
	News	40.05	540	41.98	748.9	43.76	996
16	Foreman	32.06	202.6	37.16	256.76	38.45	380.02
	Stefan	34.27	387.45	38.55	423.59	39.01	478
	News	35.87	512.6	40.62	580.46	41.2	636
32	Foreman	18.26	160.3	31.63	210	33.09	389.9
	Stefan	24.43	226.7	32.1	337.21	35.6	414.8
	News	28.87	302	34.89	392	37	588

5.2 Average throughput

Throughput or goodput is the maximum rate at which a video sequence is furnished. In Fig. 5, we compare our proposed method with other buffer management algorithms. It is clear that VM processes the incoming sequences at the average rate of 365 Kbps, which is much higher than that of the existing algorithms.

In a LTE network, multicasting can be done in many different ways. One of the popular method is eMBMS method in which each MAC-PDU (media access control protocol data units) has to be mapped to a TB (transmission block) [13]. The eMBMS method uses a local cache memory to store and receive media. However, it uses a separate software and as discussed in Section 2, these devices should have the same compatibility. In our method, virtual to reality ($V = \mathrm{R}$) mapping is done to cluster each multicast session and does not use any different software to access the virtual memory. It calls and appends using simple call functions used to store/retrieve from the physical buffer. From Fig. 6, it is clear that mapping from virtual memory to physical buffer memory is at a higher rate compared to eMBMS.

5.3 Frame reconstruction probability

We have to ensure that the frames dropped by the sender are reconstructed effectively at the client. In traditional IC (intra-frame coding), the frames are dropped by assuming a statistical model between end-to-end encryption. WZ (Wyner-Ziv) uses hash codes to decode the intermediate frames. But without a feedback, when done in VM implemented system, it decodes perfectly using the variance information, derived from our survival function described in Eqs. 6–8. In Fig. 7, VM implemented system with WZ codec has the highest probability to recover the dropped frames with lesser number of frames available to interpret (GOP < 5).

5.4 Pareto distribution

The PDF (Pareto probability density function) and CDF (cumulative distribution function) for our survival function for a multicast group with three different video sequences for α boundary conditions are plotted in Figs. 8 and 9, respectively.

Let us now compare our VM method with other buffer management algorithms which uses a constant feedback and adapts the buffer accordingly. By varying the GoP, the average PSNR and average throughput are shown in Table 2.

6 Conclusion

Additional memory can be used to assist the end-to-end video delivery to alleviate the burden of buffer adaptation from the high bandwidth requirement and rate variability of video traffic. We proposed a framework of virtual memory in streaming server and at the client to reduce the bit rate of video to be streamed, by dropping of frames, based on the variance information, which in turn reduces the bandwidth requirement of clients connected to each multicast group and assists the coding process. By reducing the bit rate by 20% during transmission, we can stream video at much lower transport bandwidth under loaded conditions, by adding a virtual memory framework. Thus, this system follows Pareto principle which is a 20–80% rule which eliminates the need for additional memory requirements in the proxy servers.

Abbreviations

ADT: Adaptive data transmission; AMP: Adaptive media playout; BBA: Buffer-based adaptation; B-frame: Bi-directional frame; BM-SC: Broadcast Multicast - Service Center; CBR: Constant bit rate; CDF: Cumulative distribution function; CIF: Common intermediate format; DASH: Dynamic adaptive streaming over HTTP; DMB: Dedicated multicast buffer; eMBMS: Evolved multimedia broadcast multicast service; FESTIVE: Fair Efficient Stable adapTIVE; GOP: Group of pixels; GSS: Global Site Selector; HAS: HTTP-based adaptive streaming; HD: High definition; HTTP: Hyper text transfer protocol;

IC: Intra-frame coding; I-frame: Independent frame; LTE: Long term evolution; MAC-PDU: Media access control protocol data unit; MOS: Mean opinion square; MPEG: Moving Pictures Expert Group; P-BUFF: Physical BUFFer; PDF: Probability density function; P-frame: Previous frame; PSNR: Peak signal-to-noise ratio; QCIF: Quarter CIF; QoE: Quality of experience; RBA: Rate-based adaptation; SD: Standard definition; SVC: Scalable Video Coding; TCP: Transfer control protocol; UDP: User Datagram Protocol; UE: User equipment; UEI: User equipment information; USB: Universal serial bus; VoD: Video on demand; V-BUFF: Virtual BUFFer; VM: Virtual memory; WZ: Wyner-Ziv

Acknowledgements
The authors would like to sincerely thank the anonymous reviewers for their helpful and constructive comments that greatly contributed to improving the paper.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Authors' contributions
AA conceived of the study, participated in its design, coordination, and interpretation of the data, drafted the manuscript, and performed the measurements. KG supervised the work. Both authors read and approved the final manuscript.

Competing interests
Both authors declare that they have no competing interests.

Publisher's Note

Received: 23 June 2016 Accepted: 20 February 2018
Published online: 02 March 2018

References

1. J Yang, H Hu, H Xi, L Hanzo, Online buffer fullness estimation aided adaptive media playout for video streaming. IEEE Trans. Multimed. **13**(5), 1141–1153 (2011)
2. S-H Lee, K-Y Whang, Y-S Moon, W-S Han, I-Y Song, Dynamic buffer allocation in video-on-demand systems. IEEE Trans. Knowl. Data Eng. **15**(6), 1535–1551 (2013)
3. FM Tabrizi, J Peters, M Hefeeda, Dynamic control of receiver buffers in mobile video streaming systems. IEEE Trans. Mob. Comput. **12**(5), 995–1008 (2013)
4. K Song, L Golubchik, in *Proceedings of 19th International Conference on Computer Communications and Networks (ICCCN)*. Towards user-oriented live video streaming (IEEE, Zurich, 2010), pp. 1–6
5. A El Essaili, D Schroeder, D Staehle, M Shehada, W Kellerer, E Steinbach, *Quality-of-experience driven adaptive HTTP media delivery. International Conference on Communications (ICC)*. (IEEE, Budapest, 2480)
6. R Asorey-Cacheda, FJ Gonzalez-Castano, in *Proceedings of the Eighth IEEE Symposium on Computers and Communications (ISCC)*. Real-time transcoding and video distribution in IEEE 802.11b multicast networks, vol. 2 (IEEE, Turkey, 2003), pp. 693–698
7. E Amir, S McCanne, R Katz, in *Proceedings of the SIGCOMM*. An active service framework and its application to real-time multimedia transcoding, vol. 28 (ACM, NY, 1988), pp. 178–189
8. R Aravind, M Civanlar, A Reibman, Packet loss resilience of MPEG-2 Scalable Video Coding algorithms. IEEE Trans. CSVT. **6**(5), 426–435 (1996)
9. J Walpole, R Koster, S Cen, C Cowan, D Maier, D Mcnamee, C Pu, D Steere, L Yu, in *Proc. of SPIE*. A player for adaptive MPEG video streaming over the internet (IEEE, Washington, DC, 1998), pp. 3210–3222
10. D Wu, S Member, YT Hou, W Zhu, Y Qin Zhang, JM Peha, S Member, Streaming video over the internet: approaches and directions. IEEE Trans. CSVT. **11**, 282–300 (2001)
11. W-H Ma, DHC Du, Reducing bandwidth requirement for delivering video over wide area networks with proxy server. IEEE Trans. Multimedia. **4**(4), 539–550 (2002)
12. Z-L Zhang, Y Wang, DHC Du, D Su, Video staging: a proxy-server-based approach to end-to-end video delivery over wide-area networks. IEEE/ACM Trans. Networking. **8**(4), 429–442 (2000)
13. D Lecompte, F Gabin, Evolved multimedia broadcast/multicast service (eMBMS). in LTE-advanced: overview and Rel-11 enhancements. IEEE Commun. Mag. **50**(11), 68–74 (2012)
14. A Tassi, C Khirallah, D Vukobratovi c, F Chiti, JS Thompson, R Fantacci, Resource allocation strategies for network-coded video broadcasting services over LTE-advanced. **64**(5), 2186–2192 (2014)
15. S Livi, L Sassatelli, G Urvoy-Keller, *HTTP adaptive streaming and access routermanagement: the users' and network's perspectives*. (ICCCI, Wuhan, 2016), pp. 118–123
16. T-Y Huang, R Johari, N McKeown, M Trunnell, M Watson, A buffer-based approach to rate adaptation: evidence from a large video streaming service. ACM Conf. SIGCOMM. **44**(4), 187–198 (2014)
17. T Mangla, N Theera-Ampornpunt, M Ammar, E Zegura, S Bagchi, *Video through a crystal ball:effect of bandwidth prediction quality on adaptive streaming in mobile environments*. (ACM, NY, 2016), pp. 1–6
18. V Mancuso, G Bianchi, Streaming for vehicular users via elastic proxy buffer management. IEEE Commun. Mag. **42**(11), 144–152 (2004)
19. J Jiang, V Sekar, H Zhang, Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive. IEEE/ACM Trans. Netw. **22**(1), 326–340 (2014)