Optimized Segmentation of H.264/AVC Video for HTTP Adaptive Streaming

Jan Lievens[*], *Shahid M. Satti, Nikos Deligiannis, Peter Schelkens, and Adrian Munteanu*

Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel (VUB)
Pleinlaan 2, B-1050 Brussels, Belgium.
Future Media and Imaging Department iMinds,
Gaston Crommenlaan 8, 9050 Gent-Ledeberg, Belgium.
*jan.lievens@etro.vub.ac.be

Abstract—**HTTP adaptive streaming (HAS) is based on transmission of independently decodable segments of a video sequence. In this paper, we investigate the coding cost of segmentation in H.264/AVC coded videos and present an optimized segmentation framework. The proposed framework minimizes the coding cost associated with video segmentation. This is achieved by an optimized positioning of the Instantaneous Decoding Refresh (IDR) slices. In particular, a first-round coded bitstream is used to determine the frame positions where the temporal prediction modes fail to achieve high compression efficiency. In the second coding round, IDR frames are inserted on the determined positions to optimize the coding efficiency of the segmentation. In a second methodology, variable IDR frame rates for different video layers are investigated. Experimental results show that the proposed strategies yield moderate coding gain compared to the uniform segmentation used in current HAS implementations.**

Keywords— HTTP adaptive streaming, HAS Segmentation, IDR slices, H.264/MPEG-4 AVC

I. Introduction

Streaming media, such as streamed videos, constitute a major proportion of the total internet traffic. In the recent years, internet access has become a commodity for mobile devices. Recent studies predict mobile data magnitude to grow more than double every year [1]. Furthermore, it is speculated that, by 2014 video streaming will account for approximately 66% of the total mobile traffic [2]. In this respect, mobile video streaming protocols have witnessed a considerable evolution in the past few years. In traditional streaming protocols, such as Real-Time Streaming Protocol (RTSP), a server keeps track of the state of a client during a streaming session. In these protocols, the server is the intelligent party, which adjusts the transmission rate with respect to the available bandwidth and other network parameters. Typically, in these protocols, once a streaming session is established, the server transmits the video packets using either UDP or TCP as transport protocols.

In contrast to traditional protocols, HTTP adaptive streaming (HAS) [3-4] works by managing media delivery through standard HTTP web servers using TCP/IP. An HTTP web server stores different video quality layers split in segments of a fixed time-length. A segment is an independently decodable portion of the video. Download of a particular segment of a certain video layer, as requested by the client, is performed as a standard HTTP download [5]. With this approach, the transmission rate can be easily varied and adapted to current network conditions after each download. HAS is preferred over the earlier developed protocols for video streaming due to two main advantages, namely, a high reliability of video transfer and a smooth interaction with firewalls and network address translation (NAT) mechanisms. Another advantage of HAS is that it is a stateless protocol, i.e., the server does not store any information regarding the client or its requests. This is especially convenient for load balancing: i.e., every request can be handled by any available server, without keeping track of which server is serving which particular client. Microsoft's Smooth Streaming [6], Apple's Live Streaming [7] and Adobe's Dynamic Streaming [8] are the currently available commercial implementations of HAS protocol systems. In relation to these commercially available systems, the MPEG standardization body had started an overarching standardization effort under the name "Dynamic Adaptive Streaming over HTTP" (MPEG-DASH). In April 2012, the standardization committee finalized its work by publishing ISO/IEC 23009-1:2012. 3GPP release 10 has adopted-MPEG-DASH for use over wireless networks.

In H.264/AVC, an independently decodable part of the video must start with an Instantaneous Decoding Refresh (IDR) frame or slice [9]. In general, an IDR slice has a cost associated with it which is the additional bitrate it requires compared to its coding as a P or a B slice. In the currently available HAS implementations [6-8], the segment sizes are fixed for all layers of video, e.g., in Apple's Live Streaming, segments are 10s in length. In general, the size of a segment specifies a compromise between the coding efficiency and the network adaptability of a coded video stream. Namely, the longer the segments, the higher is the coding efficiency and the poorer is the network adaptability, and vice versa.

In this paper, we study a novel approach for optimized video segmentation in HTTP adaptive streaming applications which performs an optimized placement of IDR slices. The proposed approach lies in contrast to the uniform segmentation performed in classical HTTP adaptive streaming, leading to improved video coding performance.

Mainly two proposals are considered. In the first proposal, we consider the adaptive segmentation where we align the IDR placement with scene changes. In general, temporal prediction over scene changes is quite poor and the high cost of an IDR frame is justified. The actual coding then becomes a two stage process. In a first stage, scene changes and periods of poor temporal predictions are identified. In the second coding round, IDR frames are inserted on these positions to create the segmented video stream for HAS.

In the second proposal, we consider variable segment lengths across different video layers. Experimental evaluations on two high definition (HD) sequences are performed to establish the practical benefit of the proposed segmentation strategies.

The remainder of this paper is organized as follows: Section II presents the adaptive segmentation methodology for HAS, while Section III evaluates its performance compared to fixed-length segmentation. Section IV presents the idea of variable length segmentation across different video layers in HAS. Finally, Section V draws the conclusions of this work.

II. Adaptive Segmentation

A. Concepts

In H.264 coded video streams, random access is enabled through the use of IDR slices [9]. These IDR slices are intra coded slices which, when decoded, also flush all the buffers, i.e., this action clears out all the reference frames. IDR slices split the temporal prediction structure of the video in a part before and after the IDR slice. Since no prior dependencies are permitted, part of the bitstream which starts with an IDR slice can be independently decoded without causing any drift. Within a scene, the bitrate cost of IDR slices is high compared to the temporally predicted slices because of the introduced break in the temporal prediction structure. However, in case of scene cuts, a temporally predicted slice may result in a large bitrate when compared to that of an IDR slice. This is because of the poor predictions resulting from employing uncorrelated reference frames.

To optimize the coding cost of adding IDR slices in the temporal coding structure, we look to the rate-distortion (RD) based mode selection process of H.264. H.264 is flexible enough to encode each individual macroblock (MB) with a mode that is freely selected even though the slice is being coded as a P- or a B-slice. Based on the available references for each MB, the RD optimization algorithm of H.264 selects the best coding mode, e.g., intra mode, skip, prediction from one or two lists of reference pictures, etc. This means that during the encoding process, even if the slice type is fixed, the coding modes for the containing MBs are not. We will employ this mechanism to determine the best position for IDR slice coding points in the video stream.

B. Adaptation Methodology

The proposed adaptive system uses two video coders in parallel. When encoding a sequence, instead of just coding each frame once, the proposed system will also encode the same frame using intra slices only, as shown in Figure 1.

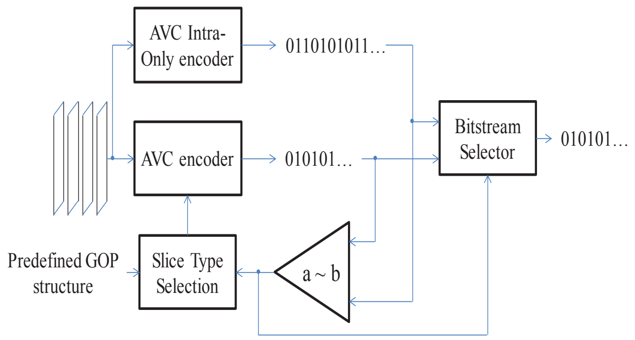

Figure 1: Adaptive system block diagram. "*a*" and "*b*" denote the bitrate costs of regular AVC and intra-only coded slices. An IDR slice is placed in the final bitstream if the coding costs *a* and *b* are relatively close.

The parallel execution, as shown in Figure 1, allows us to evaluate the cost of the regular coding compared to the intra-only case. If the system notices that the bitrate spent on the regular coding is comparable to that of the intra-only case, it assumes that the currently coded slice is suited to be coded as an IDR slice. By comparable, it is meant that the relative rate difference between the two is below a pre-defined threshold (typically 5%). In general, this will occur when temporal prediction fails, usually as a consequence of a scene change. Because the proposed system measures the actual coding costs of both temporally predicted coding and intra only coding, the proposed system allows for constructing a bit stream containing additional IDR slices while slightly increasing the bit rate as compared to a bitstream in which these IDR slices have not been placed. Since we consider one slice per coded picture, which is a common practice in many H.264 video coding applications [10], the selection of IDR slice marks the beginning of a segment in HAS. We notice that segments formed using the proposed adaptive system may not have equal sizes as in case of conventional HAS implementations [6-8].

III. Experimental Evaluation

A. Experimental Setup

To measure the effectiveness of the adaptive segmentation system described above, we have implemented such a system based on the JM-16.1 H.264 reference software. To realize an adaptive segmentation bitstream, each segment will need to be re-encoded with the newly set of IDR slice boundaries, as described in Section II.B. The experiments have been performed with different quantization parameters (QP) to generate different bitrates. The coded bit streams thus maintain a quasi-constant peak-signal-to-noise-ratio (PSNR) over all slices, while the bitrate cost of each coded slice may significantly vary over time.

One of the video sequences on which we have performed tests was specifically crafted to have multiple scene changes. We have composed this video sequence with frames from "Basketball Drive (1920x1080)" and "Cactus (1920x1080)" sequences, switching from Basketball Drive to Cactus at frame 25 and back to Basketball Drive at frame 50. The overall length is 75 frames. This produced sequence is used to discuss short term effects of scene cuts on IDR positioning

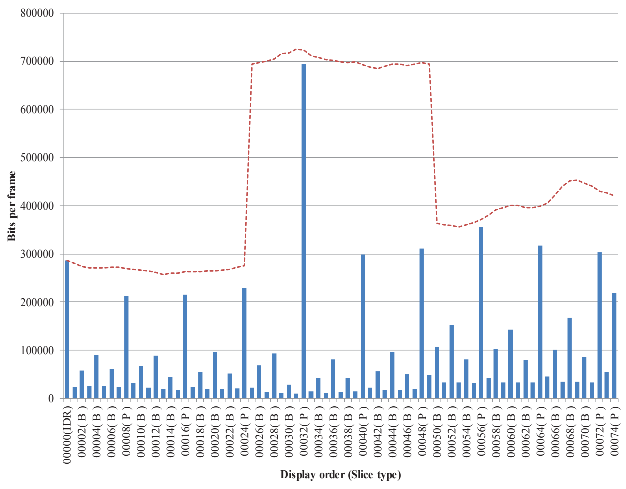

Figure 2: Comparing the normal H.264 coding of a frame to the intra-only coding. The blue bars and the red curve denote the costs for the normal and intra-only H.264 modes, respectively.

cost. We have also performed tests on “Kimono1 (1920x1080)” sequence, which has a natural scene change on frame 140.

B. Results and discussion

In Figure 2 and Figure 3, presented above, the cost (expressed in bits) to encode a frame is shown on the vertical axis and the horizontal axis is the time stamp in the display order. Figure 2 reports the cost of the normal H.264 coding compared to the intra-only coding. The test sequence is the fabricated BasketballDrive-Cactus-BasketballDrive sequence. For the normal H.264 coding there is only one IDR slice present at the beginning of the entire sequence. This allows H.264 to optimally select prediction modes for the intermediate slices, fully exploiting the temporal redundancies. It is immediately noticeable from Figure 2 that conventional H.264 coding is much more efficient than intra-only coding. However, there are positions where the cost of conventional coding is comparable to the intra-only case, e.g., for frame 32 and for frame 56, at the first P-slices after the scene changes. According to our system, these frame positions make good points for the IDR slice insertion.

The temporal coding structure of the video sequence is also obvious from Figure 2. In all coding experiments we employed hierarchical B slices with a Group of Pictures (GOP) size of 8. The QP of the hierarchical levels increases by one for each level. One can notice that these B slices are much smaller in coding size than the P slices which use only one reference frame. This difference in coding efficiency of P and B slices is large because of two main reasons: 1) B slices are predicted from two reference frames at once, one in the future and one situated in the past. This allows the encoder to more efficiently handle the failed motion estimation due to occlusions [9]. 2) P slices rely on a single reference frame which is located much farther away in time, hence there is less temporal correlation available to exploit. Note that, in the context of 2), in case of a scene change, the prediction for the MBs in the P slice will be poor. These MBs will most likely default to intra mode prediction in the

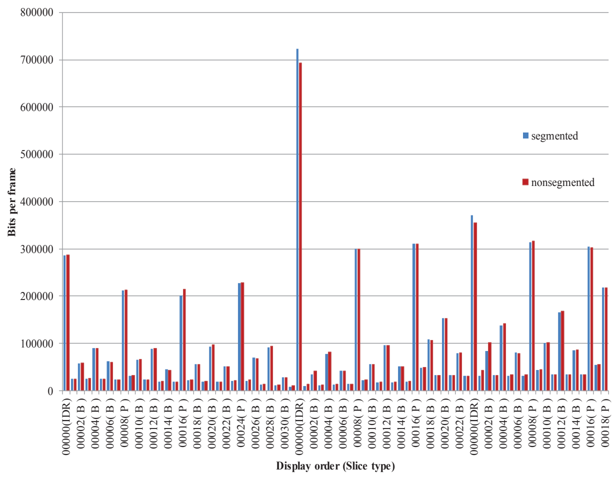

Figure 3: Frame-by-frame cost of the adaptively segmented video compared to the non-segmented video using the open-GOP.

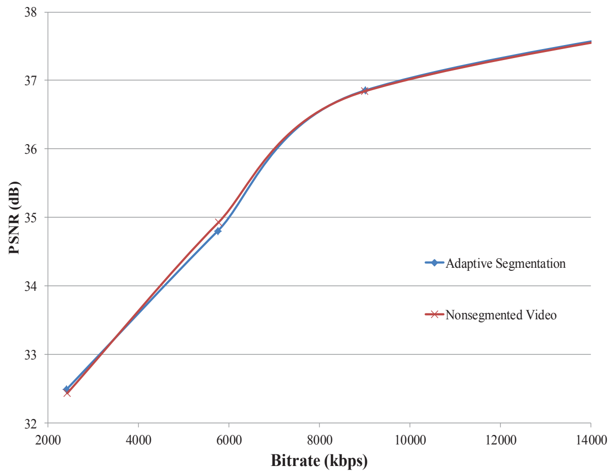

Figure 4: Basketball Drive-Cactus-Basketball Drive PSNR curves for segmented and non-segmented coding with open GOP. Tested QPs {22,27,32,37}.

normal H.264 coding. This in turn means that the additional IDR slices are most likely inserted, by the adaptive methodology, at the positions of the P slices, i.e., at the GOP boundaries. From these results we can also expect that inserting IDR slices at optimized positions in the GOP will improve the performance over inserting them at pre-defined, equally spaced locations.

For the fabricated test sequence BasketballDrive-Cactus-BasketballDrive, Figure 3 shows the cost of inserting the IDR slices at the detected positions, i.e., at the positions where cost of a normal H.264 coding is comparable to intra-only coding. In our test the IDR insertion occurs at frames 32 and 56. One can see from Figure 3 that the coding cost of adding two additional IDR slices to this 75 frame sequence on a frame-by-frame basis is small. In fact, the higher cost of the IDR frame is offset by the increase in quality it has as a reference frame, meaning that slices predicted from this new IDR frame gain in the coding efficiency compared to the non-segmented case. These tests have been performed with an open-GOP coding structure, allowing slices that are

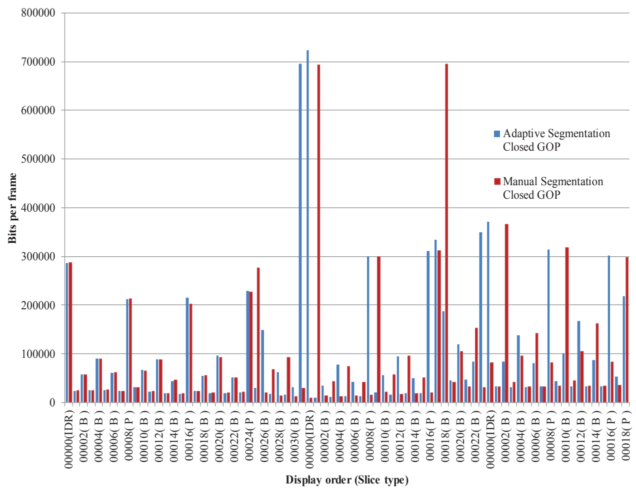

Figure 5: Comparison of two segmentation strategies for a closed-GOP structure.

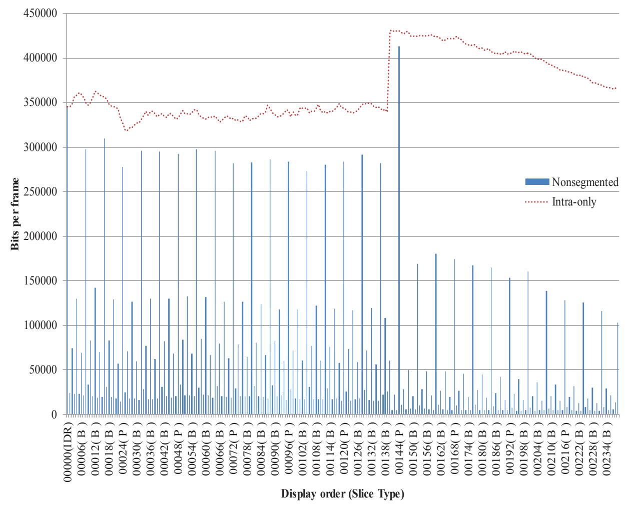

Figure 7: Bitrate cost for the IDR slices in Kimono1 sequence.

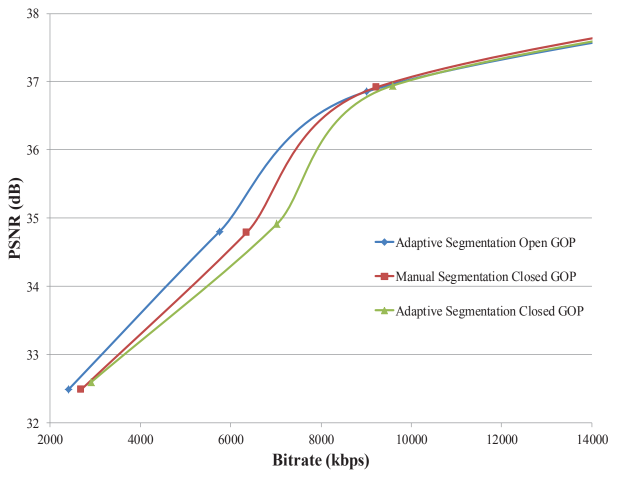

Figure 6: Comparison of open GOP and closed-GOP for adaptive and manual segmentation. In the manual segmentation, IDR slices are placed at the exact scene change positions.

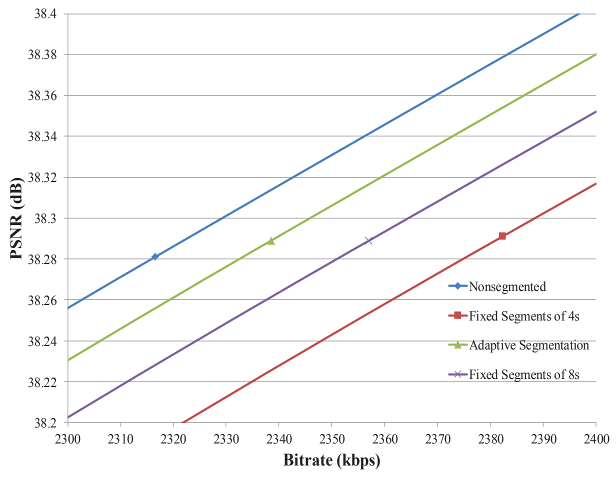

Figure 8: Coding gains of adaptive segmentation on Kimono1 sequence compared to non-segmented and uniform segmentation cases.

temporally situated *before* the IDR slice to predict from it.

The effect on the overall PSNR-bitrate curve by the proposed segmentation methodology is presented in Figure 4. It can be seen that the difference between the non-segmented coding and segmented coding is minimal. This confirms that our approach of inserting IDR slices is valid. Moreover, by increasing the number of IDR slices, the proposed technique improves the random accessibility at negligible cost.

Remark: We note that the original scene changes occur at frames 25 and 50. Concerning the inserted positions of the IDR slices with respect to the *actual* scene changes, one could conjecture if it is better to align the IDR positions with the exact frames where the scene changes happen. Imagine that the exact scene change occurs at some B slice. If we insert an IDR slice at this frame (which would normally be coded as a B slice) it would result in an incomplete hierarchical structure, potentially harming the coding efficiency. This was also an observation in our experiments. On the other hand, inserting an IDR at a frame which would normally be coded as a P slice, would leave the GOP structure intact. This in turn means that the H.264 encoder can work with hierarchical motion prediction in an RD optimal manner [10].

In Figure 5, we compare the coding cost per frame for the BasketballDrive-Cactus-BasketballDrive video sequence for the adaptive and the manual segmentation (which inserts the IDR slices at the exact scene changes). The employed GOP structure is the closed-GOP. By closed-GOP, we mean that no prediction is allowed beyond segment boundaries.

It can be seen from Figure 5 that the peaks in bitrate cost are not aligned for the two coding methods. Besides the obvious difference in IDR positions in the two strategies we can observe additional peaks at the end of the segments – see Figure 5. Since, in the closed-GOP there are few predictions at the segment boundaries, these additional peaks are due to the closed-GOP coding. The effect of closed-GOP structure on the overall PSNR-rate curve is shown in Figure 6. In Figure 6, the performance of the adaptive strategy is compared to manual segmentation for both open- and closed-GOP. The manual segmentation using the open-GOP is not depicted as it will coincide with the corresponding curve of the closed-GOP. The reason why adaptive segmentation gives poor performance for the closed-GOP is because in this case we break the temporal coding structure at the worst possible location. Frames positioned between the scene change and the IDR slice cannot be predicted from the inserted IDR slice, which would result in the missed opportunities to exploit the temporal correlation efficiently. The manual segmentation performs worse than the adaptive segmentation because it also breaks the temporal coding structure, in which, the RD mode selection of H.264 could have performed optimally. On the other hand, the adaptive segmentation system only inserts IDR slices when the RD optimizer would generate a similar cost in rate. This shows that the video segmentation should be RD-driven as proposed in our approach and not controlled using conventional scene detection techniques [11].

Next we present the coding results for the video sequence Kimono1. From Figure 7, we can see that the scene change happens at frame 139. At 24 frames per second, Kimono1 is a 10 seconds long sequence, and shows a more realistic image of the coding results brought by adaptive segmentation. We would like to point out that conclusions which were drawn for BasketballDrive-Cactus-BasketballDrive sequence also hold for Kimono1.

To confirm the benefits of the proposed segmentation methodology in the HAS, in Figure 8, we compare the PSNR-bit rate curves of coding Kimono1 with different fixed segmentation sizes (to highlight the differences, a zoomed area in the RD graphs is shown in the figure). The longer the segment size, the better the coding performance is – see Figure 8. This is because the longer the segment the lower is the chance of breaking existing temporal correlation. The exception is the adaptive segmentation case, where the segment size is chosen to coincide with the absence of good temporal prediction possibilities. We would like to mention that the use of adaptive segmentation is still valid even if the frequency in scene changes is relatively small, albeit more as a tool for squeezing out the last bit of coding performance gain from H.264 rather than an integral part of the usage of the codec.

IV. VARIABLE LENGTH SEGMENTATION ACROSS LAYERS

Fixed-length segmentation adds a lot of overhead to video coding due to the large cost of IDR slices. This is especially true in HAS scenarios, i.e. where the same video sequence is stored multiple times at different bitrates.

In the previous section, it is demonstrated that we can certainly improve the video coding performance within a layer using longer length segments or even by employing the proposed adaptive segmentation. However, longer segment lengths for all layers would sacrifice network adaptability in case of bandwidth variations. Because the absolute cost of coding IDR slices is not equal among all bitrates, we propose that the segment lengths for the higher bitrates are chosen to be larger than the segment lengths of the lower bitrate layer.

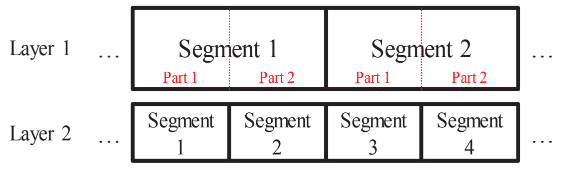

Figure 9: Segments in different layers. The relative lengths of the segments are indicated by their width, while the heights show their relative bitrates.

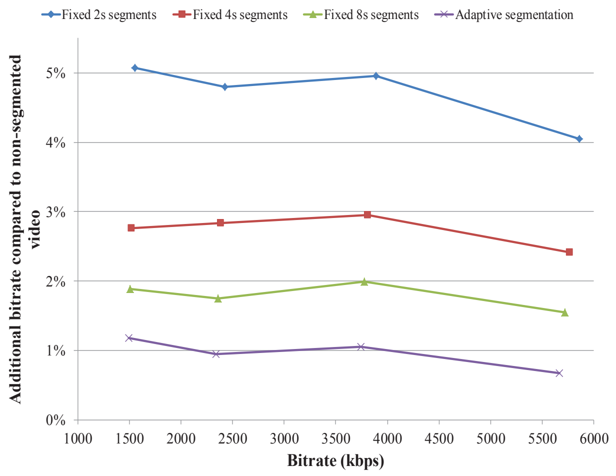

Figure 10: Bitrate cost of segmentation at different bitrates for the Kimono1 sequence.

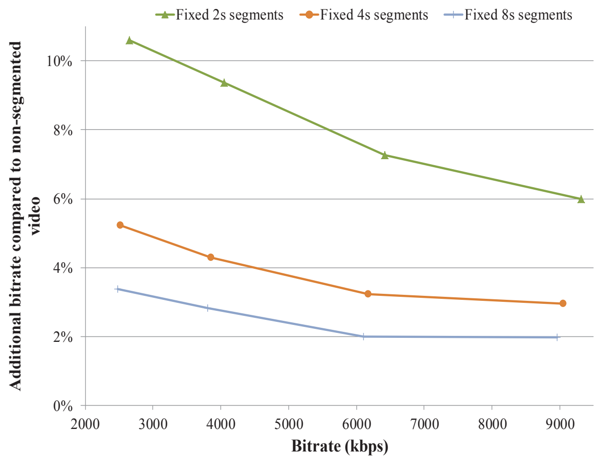

Figure 11: Bitrate cost of segmentation at different bitrates for the BQTerrace sequence.

A schematic diagram of this proposal is depicted in Figure 9.

For such a structure to work in a practical HAS system, the client needs to be made aware of the different segments lengths across different video layers.

To make this new system compatible with the existing HAS implementations which work only with uniform segmentation, each segment of a higher bitrate layer, in the proposed framework, can be split into a number of parts (or subsegments) of equal length that are independently transmittable. Note that these parts may not be decoded independently and merely serve to increase network adaptability.

In Figure 9, a simple case of two layers is shown, where layer 1 carries two parts of temporally equal length. It is straightforward to extend this framework to an arbitrary number of layers. One drawback to this schema is that there are less random access points in the higher bitrate layers, diminishing the adaptability of the HAS system to switch back to a higher quality once the transmission bandwidth improves. However, since it can be considered more important for an HAS system to be able to accommodate bandwidth shortages in the network and their associated buffer underflows, the ability to switch to a lower rate layer can be considered to be more important. Thus it can be understood that in our proposed scheme the flexibility to switch to a lower bandwidth layer is maintained at segment boundaries and segment part boundaries, while the adaptability to switch to a high bandwidth layer is diminished.

To illustrate the gains obtained using the variable length segmentation across different rate layers, we carried out the test on the Kimono1 sequence with four layers, the results for which are presented in Figure 10. As an example, it is clear that using 2s segments at 3.70 Mbps and 4s segments at 5.63 Mbps is better than using 2s segments at both rates. In Figure 11, we present results for a similar experiment on the “BQTerrace (1920x1080)” sequence. BQTerrace is a 60 frames per second test sequence whose compression benefits heavily from motion compensation. We note that on this sequence, for the tested thresholds (up to 5%), the adaptive segmentation does not insert additional IDR slices compared to uniform segmentation; hence, its results coincide to those of uniform segmentation.

As can be seen from the Figure 11, the cost in bitrate for creating short segments is even higher than that for Kimono1. In general, the optimal combination of segment lengths for different layers depends on the network variability. Namely, if the available network bandwidth is slowly varying, e.g., a managed wired network, the client does not need to switch often between different layers and the non-segmented case or using long segments are certainly the best solutions. In case bandwidth varies rapidly, the optimal segmentation will tend to be short. In the essence, we can conclude that, an optimal combination of the segment lengths for different layers, in the proposed framework of Figure 9, can be determined by accounting for the dominating behavior of the network.

V. Conclusion

The paper proposes a novel approach for adaptive video segmentation in HTTP adaptive streaming applications, based on optimized placement of IDR slices. The proposed technique leads to an improved coding performance when compared to the uniform segmentation performed in classical HTTP adaptive streaming.

The proposed adaptive segmentation relies only on internal meta-data of H.264/AVC codec; there is no need for external tools to perform scene change detection. We have shown, using the HD sequences, that better PSNR-rate performance can be achieved by the proposed system when compared to manual scene change segmentation which aligns the IDR slices with the scene changes.

Open research issues are related to the practical deployment of the proposed adaptive segmentation in HAS, including study of the delivery infrastructure, trade-offs between complexity and network adaptability, and structure and management of manifest files when using variable length segmentation across layers.

VI. Acknowledgements

This work is funded by iMinds (Belgium) and the IWT (Belgium) through the MISTRAL project. We would like to thank Jan De Cock (MMLAB-UGent) for providing us the HD test sequences. Additionally, we would also like to thank Filip De Turck (IBCN-UGent) as well as Werner Van Leekwijck and Chris Hawinkel (Alcatel-Lucent Belgium) for important discussions and remarks on the work.

References

[1] Cisco White Paper: Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2009-2014, (http://bit.ly/bwGY7L)

[2] ISO/IEC JTC1/SC29/WG11 N11338, Call for Proposals on HTTP Streaming of MPEG Media, April 2010, Dresden, Germany.

[3] O. Oyman and S. Singh, “Quality of experience for HTTP adaptive streaming services”, *IEEE Communications Magazine*, vol. 50, no. 4, pp. 20–27, April 2012.

[4] V. Adzic, H. Kalva, and B. Furht, “Optimizing video encoding for adaptive streaming over HTTP,” *IEEE Transactions on Consumer Electronics*, vol. 58, no. 2, pp. 397-403, May 2012.

[5] IETF RFC 2616: "Hypertext Transfer Protocol – HTTP/1.1", R. Fielding et al., June 1999.

[6] A. Zambelli. IIS Smooth Streaming Technical Overview. Technical report, Microsoft Corporation, March 2009.

[7] R. Pantos. HTTP Live Streaming. Internet Draft draft-pantos-http-live-streaming-04, June 2010.

[8] David Hassoun. Adobe Developer Connection. Adobe Systems. http://www.adobe.com/devnet/flashmediaserver/articles/dynstream_advanced_pt1.html. Blog. [Accessed 24 June 2010].

[9] ISO/IEC 14496-10, "Advanced video coding for generic audiovisual services," ITU-T Rec. H.264 and MPEG-4 AVC.

[10] I. E. Richardson, The H.264 Advanced Video Compression Standard, Wiley, 2011.

[11] U. Gargi, “Performance characterization of video-shot-change detection methods,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 1-13, February 2000.