Efficient Content Processing for Adaptive Video Delivery

Santhana Chari, VP Engineering Digital Video Systems, ARRIS Ramesh Panchagnula, Senior Video Algorithm Engineer, ARRIS

Contents

Introduction

Rapid increase in last-mile bandwidth availability to consumer homes has made delivery of near broadcast quality video over the public IP network possible. Adaptive Bit Rate (ABR) delivery is prevalently being used for delivering video over private and public IP networks. ABR delivery requires creation of multiple representations of the same video content so that the representation whose characteristics closely match the client capabilities and instantaneous network bandwidth availability can be adaptively chosen by the client device. It is clear that generating aforementioned multiple representations make the encoding or transcoding process more complex since it necessitates substantial increase in the computational requirements depending on the resolutions and rates of the multiple representations.

Current adaptive streaming deployments are based on proprietary delivery mechanisms developed by Apple, Microsoft and Adobe. The corresponding delivery schemes are HTTP Live Streaming (HLS), HTTP Smooth Streaming (HSS), and HTTP Dynamic Streaming (HDS). Recently ISO has released a standard based specification called Dynamic Adaptive Streaming over HTTP (DASH) [1] for adaptive video delivery. All four delivery schemes use the same underlying concept of generating multiple bitstreams for the same content and use a manifest/playlist file to communicate this information to the client. The bitstreams are fragmented into small segments of short time duration, typically 1-10 seconds in length. Based on the information contained in the manifest file the client device can switch between different representations on the fragment boundaries to adapt to variations in available network bandwidth. All these four schemes utilize H.264 AVC video coding and Advanced Audio Coding (AAC) as encoding standards. Dynamic profile selection and the co-operative transcoding techniques presented in this paper apply equally to all the ABR delivery schemes mentioned above.

Profile	Spatial Resolution Hor X Ver	Bitrate	H.264 Encoding Profile
1	1920 x 1080	5 mbps	High
2	1280 x 720	4 mbps	High
3	1280 x 720	3.5 mbps	High
4	960 x 540	3 mbps	High
5	864 x 486	2.5 mbps	High
6	640 x 360	1.2 mbps	Main
7	640 x 360	1 mbps	Main
8	416 x 240	1 mbps	Main
9	320 x 180	700 kbps	Main
10	320 x 180	500 kbps	Main/Baseline

Table 1: A typical use-case of various resolutions and bitrates of multiple streams used in ABR delivery

Table 1 shows a typical use-case of the ABR delivery. The number of representations or profiles used can vary anywhere from 4 to 16 with the use-case shown above using ten different representations. As it can be seen in the table, some of the representations are at same spatial/temporal resolutions with different bitrates (2 and 3, 6 and 7, 9 and 10). Some of the other representations have a dyadic spatial relationship (#2, #6 and #9) with the spatial resolutions of one of the representation being a multiple of 2 of the other. Straightforward implementations of ABR transcoders replicate multiple transcoders processors (ASICs) or independent software modules to generate multiple outputs. This approach, however, does not exploit the fact that the same input video stream is being transcoded to generate multiple outputs. Several processing operation required to generate multiple output streams can be shared resulting in large gains in processing efficiency details of which will be presented in the subsequent sections.

Need for Dynamic Profile Selection

As shown in Table 1 a fairly large number of profiles are used in ABR video delivery. These profiles, with their corresponding resolutions and bitrates, are usually chosen a priori to match the capabilities of various display devices that are expected to

access the content. Another factor that is considered in the selection of profiles is a target bits-per-pixel number that is chosen to be large enough to provide adequate video quality and at the same time small enough not to introduce undue network congestion.

One of the challenges with hand-picking the profiles is that the selected bitrates and resolutions may be either too aggressive or conservative due to variability of video content. We used a proprietary video quality tool to study the level of variability that can be encountered in practice. This tool estimates the perceived quality of ABR video segments and it was run on several different live video programs. Each of the video programs was coded at the profiles shown in Table 1 with segment duration of 6 seconds. The video quality tool provides no-reference scores for each segment in a scale of 1 to 100. A quality score of 75 or above is found to be visually acceptable based on our experiments. Figure 1 shows the plot of measured video quality for two different videos, "Home & Garden" and "CNBC" using Profile 2 encoding parameters. The "CNBC" video sequence exhibits a very high quality at Profile 2 encoding parameters, whereas the "Home & Garden" sequence exhibits larger variations and a lower quality. This shows the difficulty associated with static selection of profiles. If the target is to achieve a video quality of say 85%, then the CNBC sequence could have been encoded at a lower rate saving transmission bandwidth.

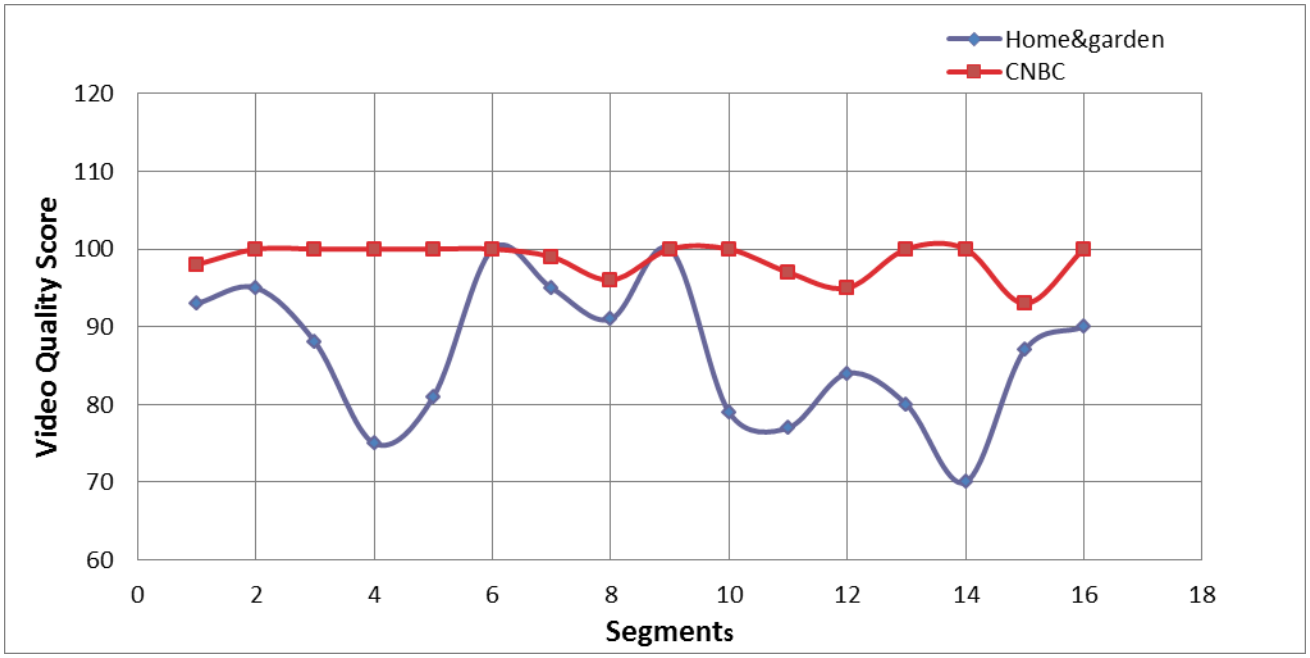

Figure 1: Video Quality of different sequences at encoding Profile 1 from Table 1

We also investigated how the video quality scores change for a given live video program with different profiles used in Table 1. In some of the sequences, the video quality of different profiles is fairly evenly spaced apart, whereas for many sequences several profiles have similar video quality. Figure 2 shows the video quality score for the first three profiles of the CNBC sequence. As expected Profile 1 has a very high video score. Profiles 2 and 3 have a fairly similar video quality score indicating that Profile 3 could have been coded at a lower rate.

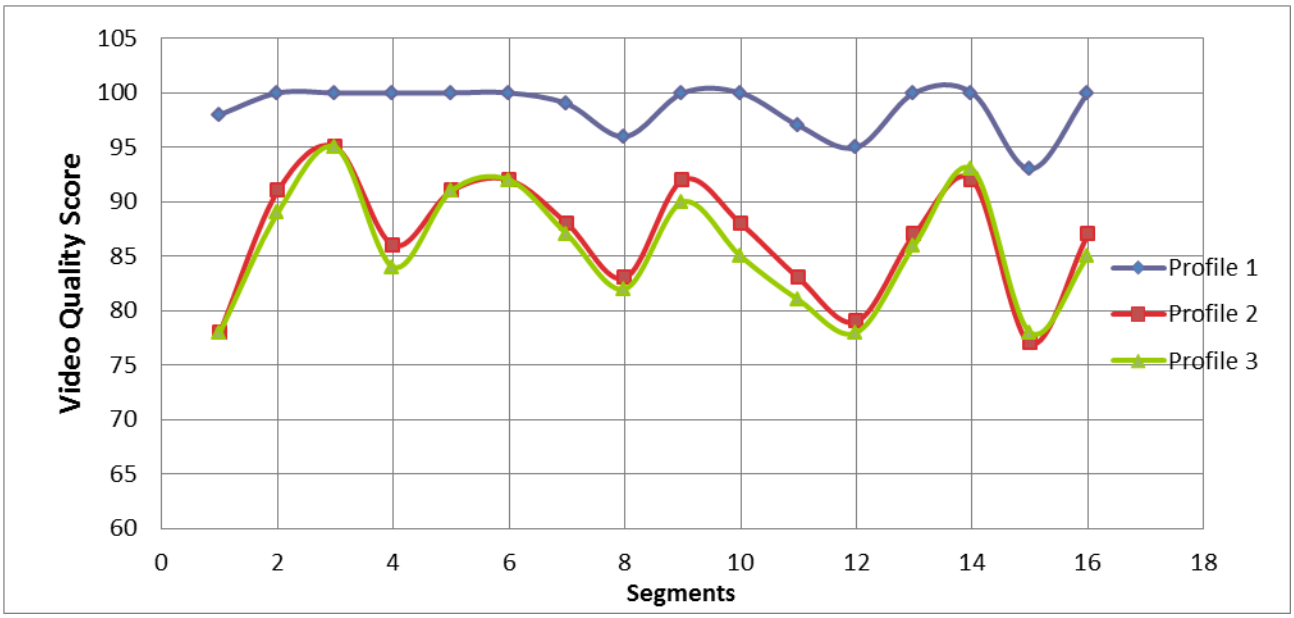

Figure 2: Video Quality of different profiles

As the ABR video delivery technology matures, these profiles can be selected in a dynamic content-adaptive fashion in real time. A transcoder/encoder device that uses a two-pass video processing architecture can derive estimates on expected video quality for different profiles and then dynamically choose the combinations of profiles in an adaptive fashion. In an alternate architecture, a transcoder can generate bitstreams corresponding to a large number of profiles. This can be followed by a downstream analysis device that inspects and compares the bitstreams corresponding to the individual profiles and then dynamically chooses a subset of profiles to be used. This can be accomplished in real-time with a small amount of processing delay.

Co-Operative Transcoding

Co-operative transcoding technology introduced in this paper leverages the common processing operations required to generate different output profiles in ABR transcoding. Before exploring the details of co-operative transcoding, let us take a look at the processing operations involved in generating a single output profiles. Figure 3, shows a high level diagram of various functional processing blocks required to perform transcoding from MPEG-2 or H.264 input stream to a H.264 output stream. The input streams can be either real-time "live" streams delivered over an IP/Satellite network or pre-encoded streams stored as files. Note that it is possible to perform such a transcode operation using the encoding parameters contained in the bitstream without fully decoding the input bitstream.

However when operations such as deinterlacing and arbitrary resolution scaling are required, then full decoding followed by re-encoding offers the most flexible solution. The Decode module converts the input compressed bitstream to pixels, followed by the Deinterlace/Scaling module which produces a progressive output at, possibly, reduced spatial or temporal resolution. This is followed by various encoding functions. Since the Motion Estimation module consumes a large portion of the encoder processing requirement, we have shown the encoding as two functional blocks, Motion Estimation and Other Encode Operations. "Other Encode Operations" include among others Macroblock coding mode decision, Quantization, Transform, Deblock Filtering and Entropy coding. Detailed description of H.264 encoding modules can be found in [2].

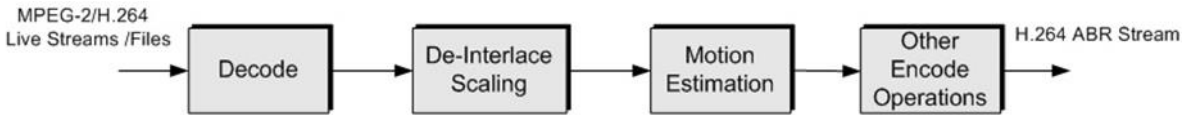

Figure 3: Major processing modules of a video transcoder

Table 2 shows the approximate processing cycles required by a broadcast quality encoder that is optimized to run on a flexible processing platform. These numbers are obtained by running our transcoder on several typical input streams and averaging out the processing cycles over those sequences. We used MPEG-2 HD streams as inputs and generated H.264 HD output streams. In our implementation the motion estimation is broken down in two parts (ME1 and ME2). The ME1 module uses input pictures for motion estimation which can be performed with look-ahead processing and the ME2 module then performs further refinement using reconstructed pictures based on the results of the ME1 module. The aggregate 61.4% processing required for Motion Estimation listed in Table 2 is split between ME1 and ME2 by 22.4% and 39% respectively.

Processing Function	Processing Units/ Macroblock	% of the total transcoding processing
Decoding and simple de-interlacing	3100	9.5%
Motion estimation (including quarter pel refinement for multiple block shapes and references)	20160	61.4%
Prediction, Intra Prediction, Quant and Dequant, Transform and Inverse Transform	6326	19.2%
Deblock and Entropy Coding	3245	9.9%

Table 2: Percentages of processing cycles used by different modules in a typical H.264 transcoder

To generate ABR output at multiple profiles, the transcoder shown in Figure 1 can be replicated several times to produce multiple output streams for the same input. Obviously in addition to replicating the processing, certain synchronization mechanism has to be in place so that different output streams have aligned "fragments" to allow the client to switch from one stream to another at the boundary of these fragments. However, a more efficient implementation of an ABR transcoder can be achieved by judicious sharing of processing modules.

Figure 4 shows an exemplary implementation whereby some of the common processing modules such as the Decode and De-Interlace are shared between multiple output streams. In addition, as it can be seen from Figure 4 the encoding modules are also tightly coupled. For example, if the ABR Stream 1 is a low resolution stream and ABR Stream 2 is a higher resolution (twice) stream, then the Motion Estimation operations need not be fully independent. The motion estimation results of Stream 1 can be used to derive or refine the motion estimation operations of Stream 2. Such co-operative operation may be somewhat more complex to implement, but results in savings in processing requirements and ensure consistency of encoding across multiple streams. As video transcoding is increasingly done in software or programmable processors, implementing transcode operations with shared functional blocks becomes feasible and offers substantial reduction in processing requirements.

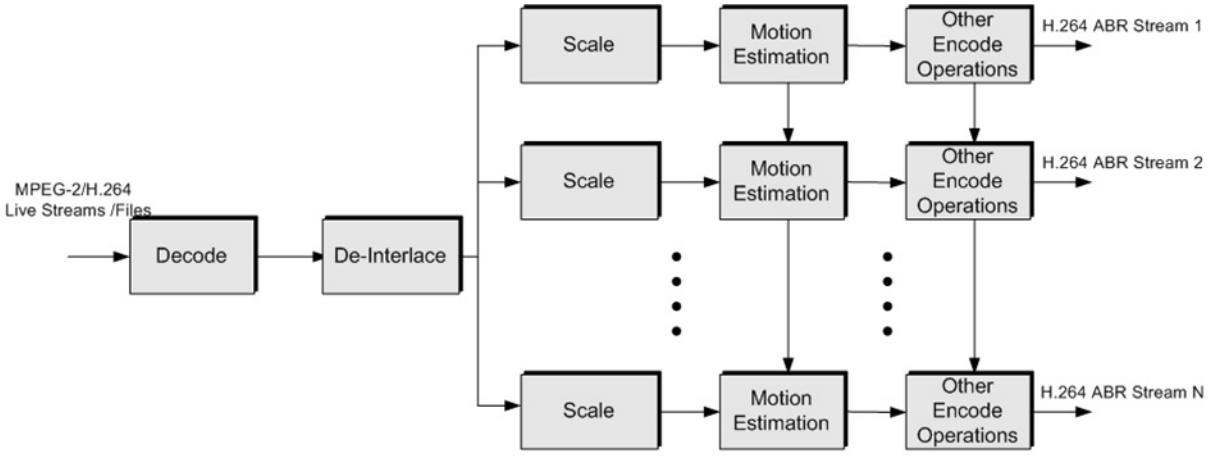

Figure 4: Co-operative transcoding using sharing of processing modules between different output streams.

In the subsequent sections, we will present more details on how co-operative transcoding can be implemented in a real-time system and provide quantitative results on how much processing can be saved.

ABR Streams at Different Bitrates

In this section we present quantitative details on how Co-operative transcoding can be used to efficiently generate multiple streams, when subsets of streams are required at different bitrates with the same spatial/temporal resolution. Referring back to Table 1, this corresponds to the case of streams 2 and 3, 6 and 7 and 9 and 10. As mentioned in the previous section and shown in Figure 1, it is straightforward to share operations like Decoding and De-Interlacing among all the streams. Processing module that is labeled as Scale in Figure 1 is the one that performs conversion of any required spatial and temporal resolution. The "scaling" operation can be shared among streams that have the same spatial/temporal resolution, but coded at different bitrates.

In addition to sharing the obvious processing modules mentioned above, we also attempted to share more computationally intensive modules such as Motion Estimation. Our goal is to transcode a given input bitstream at multiple output rates, say, b1, b2, b3 and b4 (b1 < b2 < b3 < b4). Instead of independently re-transcoding at four different rates, we first transcoded the input stream to the highest bitrate of b4. During this transcoding, best motion vector for each individual block shape (H.264 allows different blocks shapes such as 16 x 16, 16 x 8, 8 x 16, 8 x 8 and smaller for inter motion estimation) and reference picture are computed, and the best motion vector along with the cost (SAD or SATD) are saved. While generating transcoded streams at rates b1, b2 and b3, we re-used the motion vectors and scaled value of costs that were generated during transcoding at bitrate b4. Note that all of these can be done as a part of a real time transcoding system.

Intra Coding candidate directions and costs, and direct mode costs are re-computed for individual bitrates. The motion vector costs obtained from bn stream are scaled using the following formula:

$$MV\_cost\ (bi) = MV\_cost\ (bn) * f\ (QPn, QPi)$$

where QPn is the average quantizer value for a picture while the stream is encoded at bitrate bn. The function f(.,.) was modelled by computing the motion estimation cost when the reference and current picture were coded at different QPs for a large number of pictures. The scaled costs for various non-intra motion vectors are compared with the Intra and Direct mode costs to arrive at the final coding mode.

To evaluate the efficacy of the proposed co-operative transcoding, we used three test sequences, basketball, Norway, and Newmobcal. The test sequences were

selected to include fast moving sports content, slow-moving high texture content with skin-tones and widely available standard test sequence. All three sequences are transcoded from 1080i MPEG-2 inputs at 20 mbps to a 1080i H.264 outputs at various bitrates. The output bitrates are chosen to be at 5, 6, 7, and 8 mbps. Two different objective metrics were used for the evaluation in addition to subjective visual testing. For objective measurement, we used the SSIM (Structural Similarity Index) [3] and PSNR (Peak Signal-to-noise ratio). Results for only SSIM are presented here due to space constraints.

Output Bitrate in mbps	SSIM with independent transcoding	SSIM using Co-operative Transcoding	SSIM Difference	Savings in processing
8.0	0.9302	0.9302	0	0%
7.0	0.9262	0.9249	-0.0012	70.9%
6.0	0.9168	0.9148	-0.0020	
5.0	0.9040	0.8986	-0.0053	

Table 3: Processing gains with Co-operative Transcoding

The second column in the table above shows SSIM values obtained with independent transcoding of streams at 5, 6, 7 and 8 mbps averaged over different input test sequences. The third column shows the SSIM corresponding to the use of Co-operative transcoding. Here the first output stream is generated at 8 mbps, and the motions_vectors and scaled costs are used in generating the rest of the streams. The fourth column shows the difference in SSIM performance due to the use of scaled costs. As expected the difference in the SSIM performance increases with the difference in the bitrates, but still are substantially small. Here for the first stream is generated at 8.0 mbps, there is no gain in processing since that stream was transcoded independently. For the other streams, processing required to perform decoding, de-interlacing and motion estimation (refer to Table 2) are saved resulting in close to 70% reduction in processing requirement per individual output stream.

It should be noted that there are several possible avenues for further improving the SSIM performance while retaining the savings in processing presented in Table 3. Results generated in Table 3 are based on generating the 8 mbps output stream independently and then re-using the motion vectors at all other rates. Since the re-use performance deteriorates with the increase in the difference in the bitrates,

performance can be improved by starting with a rate that is at the mid-point of the four required output rates, i.e., 6.5 mbps. Note that this rate does not need to be one of the required output rates, but the motion vectors computed at this rate will be re-used to generate the streams at the required output rates. Another area of improvement is minor refinement of the motion vectors at each individual rate and re-computation of the MV_cost instead of simply using the scaled costs. This would result in a very small increase in processing requirement.

ABR Streams at Different Resolutions

Another scenario that has been considered in this paper is the use of Co-operative transcoding to save compute processing by sharing operations between streams that are generated at different spatial resolutions. As shown in Table 1, multiple streams for ABR are generated with the same aspect ratio, and frequently there are streams where the spatial resolutions are multiples of each other by a factor two, e.g., 1280 x 720 @ 30 fps, 640 x 360, and 320 x 180. Sharing of transcoding operations between streams of different resolutions are somewhat more difficult and more complex compared to the technique presented in the previous section.

Similar to discussion in the previous section, sharing of Decode and De-Interlace operations among different streams is trivial and straightforward. The Scale operation, however, cannot be shared since the spatial resolutions of the output streams are different. Re-use of motion vector information becomes somewhat more involved with streams of different resolution. H.264 uses different block-shapes for motion estimation starting from a sub-blocks of 4 x 4 pixels to all the way up to 16 x 16 pixels containing the whole Macroblock. Many different intermediate block shapes such as 8 x 8, 8 x 4, 4 x 8, 16 x 8 and 8 x 16 pixels are also supported.

Motion vectors computed for a 16 x 16 block in 1280 x 720 resolution can be re-used for a 8 x 8 block (at the corresponding location) in the 640 x 360 resolution and for a 4 x 4 block (at the corresponding location) in the 320 x 180 resolution. In the implementation used in the paper, motion estimation is comprised of two parts, namely ME1 and ME2. ME1 performs motion estimation for different block shapes using original (not reconstructed) input pictures at integer pixel accuracy and ME2 involves refinement of the search result from ME1 using reconstructed pictures including sub-pixel accurate motion vector computation. Such a processing structure allows for look-ahead processing whereby a large number of

frames, typically one or two GOPs, can be buffered for motion estimation during real time processing.

In our experiments with Co-operative transcoding at multiple different resolutions, we re-used the results of ME1 motion estimation. The ME1 motion estimation is computed at the highest resolution (say 1280 x 720) and the motion vector results are scaled and used at the lower resolutions (640 x 360). The ME2 motion estimation is performed independently for all different output streams.

Output Resolutions and Bitrate	SSIM with independent transcoding	SSIM using Co-operative Transcoding	SSIM Difference	Savings in processing
1280 x 720 @ 4.0 mbps	0.9670	0.9670	0.0	0%
640 x 360 @ 1.2 mbps	0.9462	0.9452	- 0.0010	31.9%
640 x 360 @ 1.0 mbps	0.9397	0.9389	- 0.0008	
640 x 360 @ 0.8 mbps	0.9354	0.9344	- 0.0010	

Table 4: Processing gains with Co-operative transcoding for output streams at different spatial resolutions

Table 4 shows the results of processing gains that can be accomplished through Co-operative transcoding. Three input test sequences, all of which are 1920 x 1080, described earlier are used. The second column in the table above shows the SSIM measure obtained with independent transcoding of streams. The third column shows the SSIM corresponding to the use of Co-operative transcoding. Here for the first stream that is generated at 1280 x 720 @ 4.0 mbps, there is no gain in processing since that stream was transcoded independently. Rest of the streams at 640 x 360 re-used the decoding, de-interlacing and ME1 operations from the 1280 x 720 stream. From Table 2, it can be seen that the decoding, de-interlacing and ME1 operations require about 31.9% of processing. It should be noted that the deterioration in the SSIM performance is bit-rate independent and is very small.

In Table 4, the processing gains are tabulated under the assumption that each 640 x 360 stream is generated by re-using the decisions of the 1280 x 720 stream. As shown in Table 3, coding decisions made for one of the 640 x 360 streams can be

used in generating the other streams, thereby providing larger reduction in processing requirement which is not captured in the above table.

Summary

As the high speed Internet bandwidth availability to consumer homes and the processing capabilities of tablet/mobile devices increase, higher and higher resolutions of video will be delivered over IP networks to homes. Use of ABR delivery allows the support for multiple devices and networks of different bandwidth capabilities, but increases the computational complexity required at the transcoder to generate multiple streams. In this paper we demonstrated the limitation of static selection of ABR profiles and introduced methods for dynamic content-adaptive selection of these profiles.

We also introduced a novel Co-operative transcoding technique. Co-operative transcoding can be used to substantially reduce the computational requirements of an ABR transcoder without any appreciable loss in video compression performance. Algorithmic techniques to support multiple output streams at different bitrates with the same resolution, and multiple output streams at different bitrates and resolutions were discussed. Quantitative results for savings in computation processing along with video quality performance were also provided.

Related Readings

- Transcoding Choices for a Multiscreen World - This paper explores the applications for home- and network-based transcoding, and previews some of the innovations that are emerging to help providers transcode their content more efficiently and effectively in the multiscreen world.

- Dynamic Profile Selection & Cooperative Transcoding: An Efficient Future for the Multiscreen World – this paper presents a co-operative transcoding technique that can substantially reduce the processing requirement of generating multiple representations as well as produce a uniform quality of user experience across the multiple representations.

- Improving Adaptive Video Delivery through Active Management– This paper looks at how service providers can use adaptive delivery technology to deploy unified video processing workflows, which they can use to manage large-scale video delivery over unmanaged networks.

References

[1] T. Stockhammer, I. Sodagar, "MPEG DASH: The Enabler Standard for Video Delivery Over the Open Internet," IBC Conference 2011, Sept 2011.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra, "Overview of the H.264 / AVC Video Coding Standard", IEEE Transaction on Circuits and Systems for Video Technology, Vol 13, No 7, July 2003.

[3] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli "Image Quality Assessment: From Error Visibility to Structural Similarity", IEEE Transaction on Image Processing, Vol 13, No 4, April 2004.