Optimized adaptive HTTP streaming for mobile devices

Velibor Adzic, Hari Kalva, Borko Furht
Multimedia Lab, Department of Computer and Electrical Engineering and Computer Science,
Florida Atlantic University, FL, USA

ABSTRACT

In this paper we present a solution to improve the performance of adaptive HTTP streaming services. The proposed approach uses a content aware method to determine whether switching to a higher bitrate can improve video quality. The proposed solution can be implemented as a new parameter in segment description to enable content switching only in cases with meaningful increase in quality. Results of our experiments show clear advantages of using additional parameter in DASH implementation. The proposed approach enables significant bandwidth savings with minimal decrease in quality. It guarantees optimal path of adaptation in various scenarios that can be beneficial both for network providers and end users.

Keywords: Adaptive streaming, HTTP streaming, DASH, mobile video

1. INTRODUCTION

Users across world are watching more video away from usual devices – TVs and computers. Significant portion of video consumption is done on mobile devices. Modern smartphones have powerful processors that are capable of decoding high resolution video streams. Coupled with this, we witness emergence of network technologies that are bringing broadband access to mobile users. Recently published report[1] found a 93 percent increase in mobile video streaming during the first half of 2011, accounting for 39 percent of all mobile bandwidth, which makes video the largest single user of mobile bandwidth. Same study showed that YouTube related traffic accounts for 22 percent of all mobile data bandwidth and 52 percent of total video streaming. Trends of mobile bandwidth consumption are show in Figure 1 (courtesy of Allot communications).

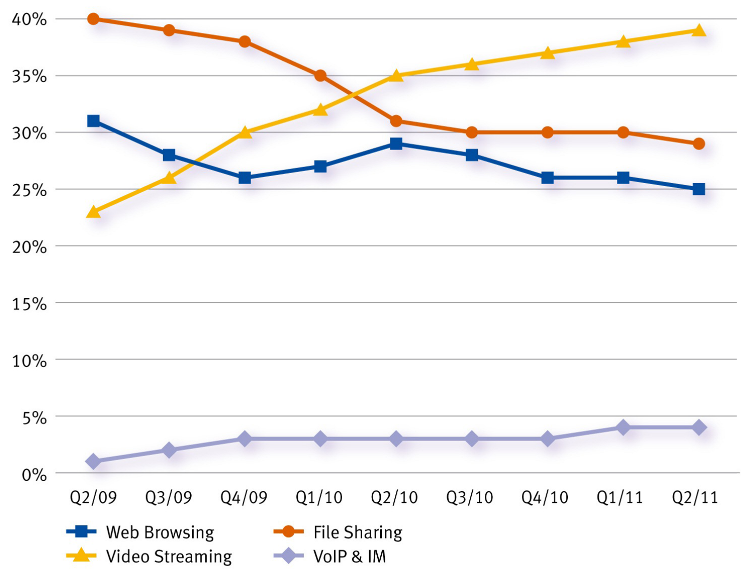

Figure 1. Mobile data usage trends broken down by top applications (Q2/09 - Q2/11).

It is clear that we need optimized platforms for video content delivery that will satisfy increased demand for bandwidth resources. According to the Internet Phenomenon report[2],"In North America, Netflix is now 29.7% of peak downstream traffic and has become the largest source of Internet traffic overall. Currently, Real-Time Entertainment applications consume 49.2% of peak aggregate traffic, up from 29.5% in 2009 – a 60% increase". In the meantime Facebook, Amazon and Wal-Mart started online video streaming services for subscribers. With this in mind it's reasonable to predict further rise of video-related traffic on Internet, influenced by both desktop and mobile users. However, in order to fulfill high demand, mobile network operators have begun enforcing policies such as bandwidth throttling, overage charges, and transcoding.

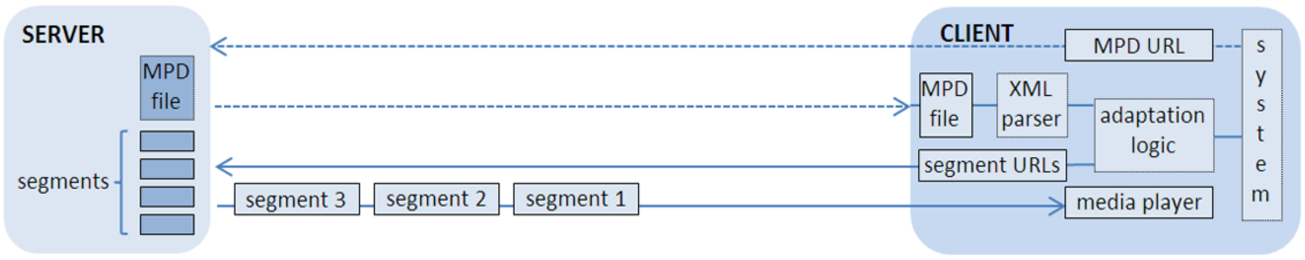

Figure 2. Dynamic Adaptive Streaming over HTTP client/server model.

1.1 Adaptive HTTP Streaming

Having recognized the trend of increased video streaming volume some of the leading companies implemented streaming over Hypertext Transport Protocol (HTTP) as a solution. This approach gives transparent access for any device that connects to Internet, circumventing firewalls that can block ports that are not used for usual protocols. Also, it makes utilization of network resources more predictable and allows optimizations. Early implementations used progressive download as a solution. In this scenario user starts downloading packets with video content and is able to start playback as soon as predefined buffer is filled. Download continues until user terminates session. It is easy to see what the drawbacks of this approach are – first, there is only one version of video stream offered (no matter what available bandwidth is) and even worse, user will download portions of video that might never be played back. A platform that would allow different versions of content and better user control was needed. This is why adaptive HTTP streaming was introduced. Examples of such platforms are Microsoft's Smooth Streaming[3], Adobe's HTTP Dynamic Streaming[4] and Apple's HTTP Live Streaming[5]. Almost all platforms went through several versions and revisions but the core concept is same today as it was in the beginning. Typical adaptive HTTP streaming server/client model is shown in Figure 2: stored on the server side are segmented (or fragmented) video sequences and description playlist (manifest file, Media Presentation Description (MPD)). User sends request and downloads a manifest file. Upon download, client software is responsible for parsing of MPD files and using adaptation logic to decide which segments from playlist to fetch.

Although all current implementations are based on common principles of pre-fetching video segments for playback, there are substantial differences that prevent cross-usage. In order to standardize a common platform, Moving Picture Experts Group (MPEG) formed working group that began standardizing Dynamic Adaptive Streaming over HTTP (DASH)[6]. This new specification is based for the most part on previously published 3rd Generation Partnership Project's (3GPP) specification[7].

The DASH specification defines a hierarchical data model for the presentation description. At the highest level there is Media Presentation that consists of one or more Periods. Period consists of Groups, Groups contain Representations and each Representation has one or more segments. Every level is described by a set of attributes, some of which are mandatory and most of which are optional. Information about content attributes can be parsed from levels above Segment. This includes information about video resolution, framerate, codec, etc. However, on a Segment level (which is only level that goes across temporal dimension) there is only information about Uniform Resource Locator (URL) from which segment can be downloaded. This means that all segments within one representation share same information, except for URLs. There is no information that would allow adaptation logic to discriminate between segments within same representation. We view this as a serious limitation that prevents full potential of adaptive streaming. Our proposal successfully removes limitations and allows additional savings and better utilization of network resources. In next section we will shortly present some published work related to improvements of DASH.

2. RELATED WORK

Publication of first draft versions of DASH specification prompted start of efforts toward optimization of adaptive HTTP streaming. Most of these publications[8, 9, 10, 11] propose models for better control of network parameters and are in essence quality agnostic. Proposed solutions include better adaptation logic on client side that implements enhanced bandwidth estimation, using specific codecs, namely H.264 Scalable Video Coding extension (SVC) for content preparation instead of H.264 Advanced Video Coding (AVC). The authors show a better caching performance of underlying network and hence better hit ratio for SVC. However these solutions do not guarantee that the optimum adaptation path takes into account quality metrics of segments. On the other hand, there are publications[12, 13] that address quality oriented performance of streaming video, comparing H.264 AVC and MPEG-4 (part-2) performances and analyzing H.264 AVC rate distortion (RD) characteristics in relation to resolution and framerate. However, these publications do not address adaptive HTTP streaming. To the best of our knowledge there is no published work related to quality aware optimization of adaptive HTTP streaming.

3. QUALITY AWARE MODEL

As we mentioned before, current DASH specification does not provide apparatus for discriminating between segments in same representation. This limits information that adaptation logic can use for constructing optimal path. Parameters that can be used for adaptation are video characteristics such as bitrate, resolution, framerate and codec type. While this set of parameters is very useful for “blind” characterization of video stream, it doesn’t provide sufficient information about quality of video bitstream. As we know quality of encoded video is affected by aforementioned parameters, but also by video content. In order to describe spatial and temporal content characteristics we would have to use relatively complicated parameters that would additionally clutter specification that is already reaching the limits of usability. It would require specifying additional attributes at segment level. On the other side, we can use parameters that are result of encoding process. Modern encoders take into account spatial and temporal characteristics of video content and some of the coding parameters are closely correlated with quality levels – in case of H.264 AVC coding, which we are using for experiments, the quantization parameter (QP) is directly related to objective quality of compressed stream (although similar can be said about most of the modern codecs that are based on transform coding, such as VP8 used for WebM).

The intrinsic limitation of adaptive HTTP streaming comes from the fact that it is expected to work with close to constant bitrates. All versions of content that are provided on the server must have stable bitrate. This means that for preparation of content we must use constant bitrate coding (CBR). On the opposite side of the RD spectrum we have variable bitrate coding (VBR), which keeps objective quality (and hence QP) constant with high standard deviation of average bitrate. From quality of experience (QoE) standpoint that doesn’t take into consideration network delays constant quality is preferred over constant bitrate. However, for online streaming of video, especially over HTTP it would be unacceptable to have variations in average bitrate as it would produce unexpected delays and jitter (Figure 3).

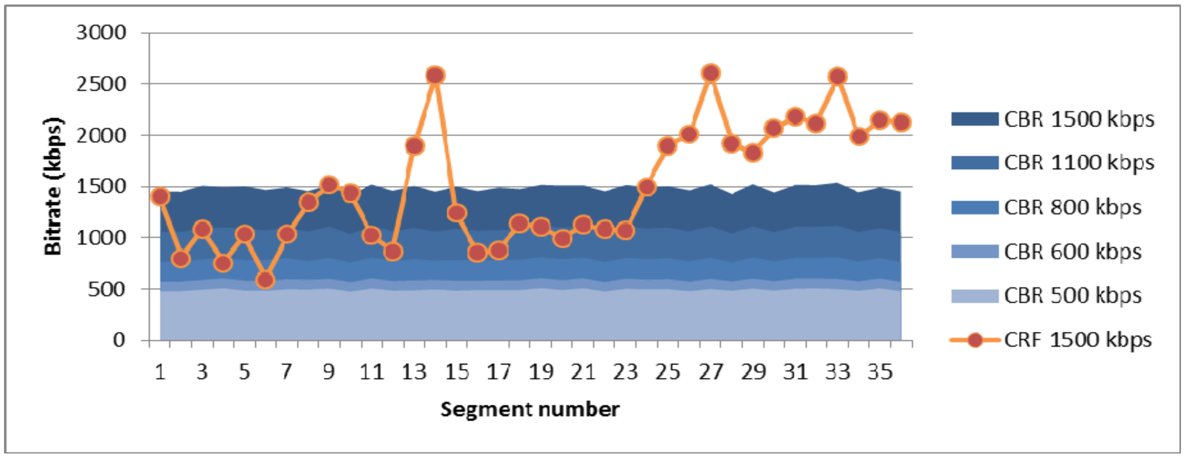

Figure 3. Average bitrate for test sequence encoded in five different CBR versions and one VBR version.

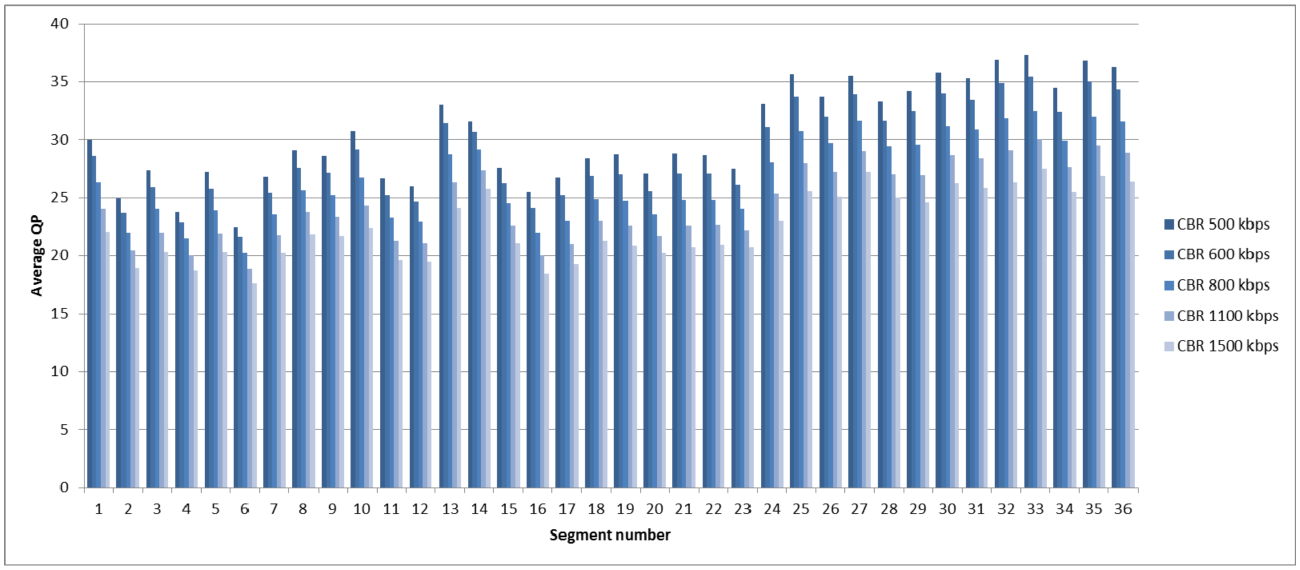

Figure 4. Average QP levels used for frames in segments.

Figure 3 shows CBR and VBR encoding of the same sequence; five CBR bitrates are used and VBR averages to 1500 Kbps. Detailed parameters of selected encoded sequence will be presented in next section. The version of VBR coding used employs constant rate factor (CRF) optimization. The video is divided into segments of 10 seconds each for evaluation purpose. Here, it is important to notice that all five CBR sequences have stable average bitrate for all 36 segments, while VBR sequence exhibits significant fluctuations. This shows that there is indeed significant discrepancy between segments if we want to achieve constant quality. Segments that contain frames with high level of motion and frequent scene cuts are encoded with higher bitrates. In order to maintain same bitrate level, CBR forces an increase in average QP (decrease quality) for the same frames. Average QP per segment for CBR sequences is shown in Figure 4. This further means that quality difference between segments at same temporal position in different representations varies. We used structural similarity index[14] (SSIM) to measure objective quality of the video segments. Figure 5 shows the average of SSIM difference between successive bitrates for CBR coded video.

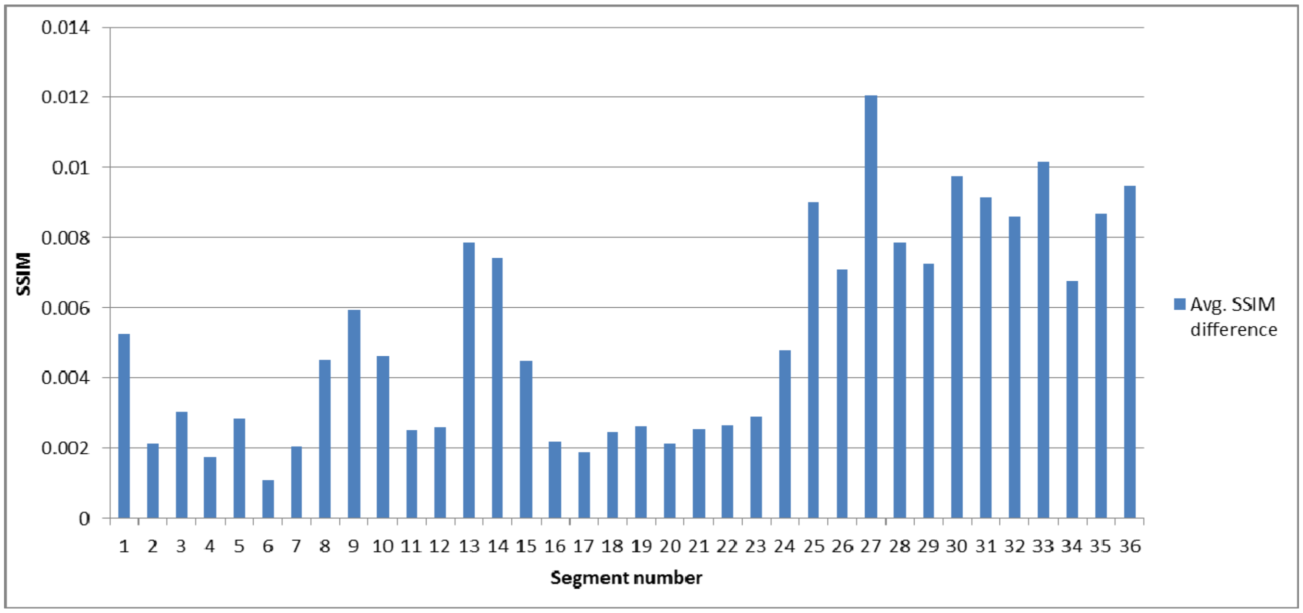

Figure 5. Averaged SSIM index difference between segments in all five CBR sequences

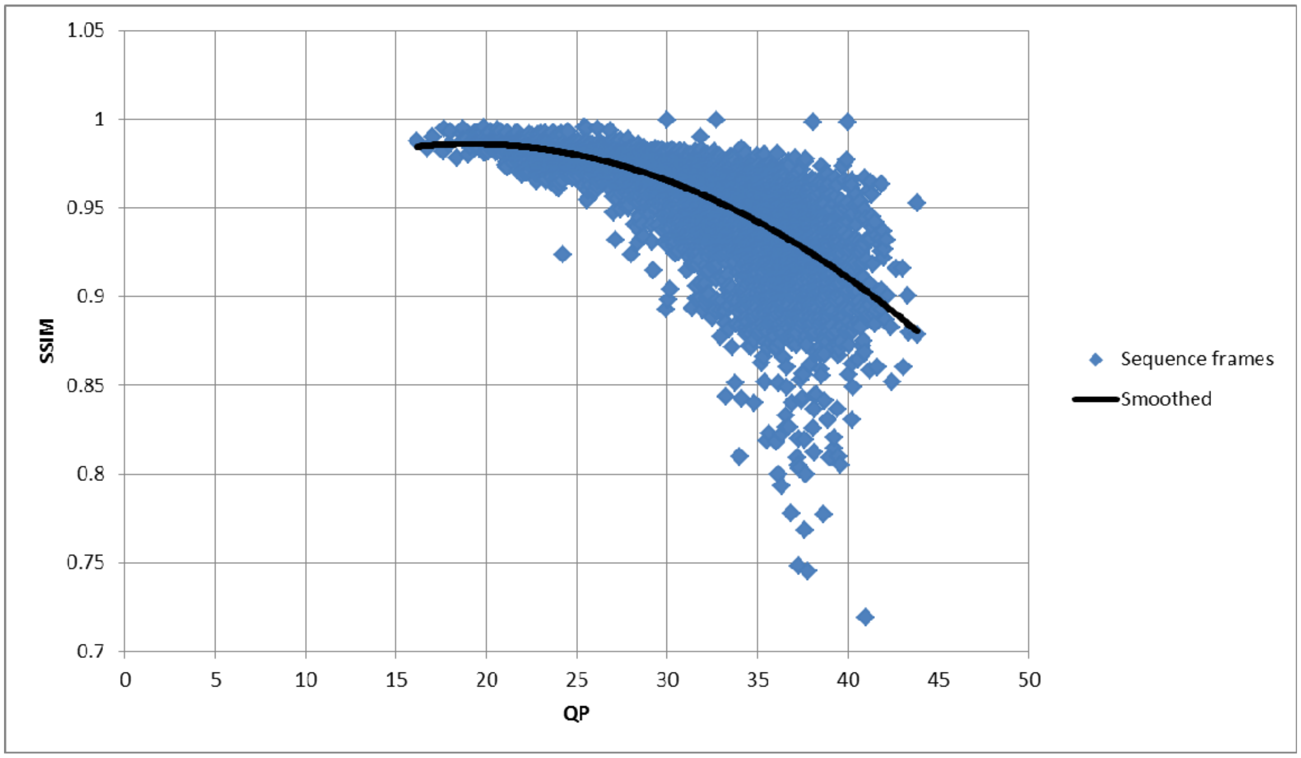

Figure 6. QP – SSIM relation graph. Blue spots represent frames in selected video sequence.

As we can see from Figures 4 and 5, the delta SSIM differences are lowest for segments that are encoded with lower average QPs. This can be explained if we take a look at QP to SSIM relation graph for all frames in video sequence. The relation is not linear – as we increase QP, differences in quality for different QP steps increase. When we encode video at higher bitrate, QPs get decreased for the same amount across all frames. However quality index increases more for the frames that had higher QP. In other words quality index is increased more for the frames that had lower original quality index. This is why QP information is sufficient to detect candidate regions of video sequence (or, in our case, segments) for which we can lower bitrate and get smallest decrease in quality, compared to model where this regions are random and selected only based on bandwidth conditions of network link.

To further establish clear relationship we sorted segments in increasing order according to average QPs. Then we sorted them in increasing order according to average SSIM index difference between all segments with same time stamp across offered bitrate levels. Results are shown in Table 1. This can be observed in Figures 4 and 5 as well. Among first 18 segments that are best candidates for switching based on QP there are 17 matches in list of segments that produce lowest quality loss according to SSIM index. This confirms our hypothesis that QP information obtained from compressed domain can indeed be used for quality aware model of adaptive switching. In next section we will present results of experiments that are conducted using test sequences and network simulation for two cases: first where quality parameter is available in original content and second case where there is no quality parameter and information has to be obtained by means of QP extraction presented in this section.

Table 1. Segments from selected sequence sorted in ascending order by average QP and average SSIM difference.

Criteria	First 18 segment indexes sorted by criteria
Average QP	6, 4, 2, 16, 12, 11, 17, 7, 20, 5, 3, 23, 15, 18, 9, 22, 19, 21
Average SSIM difference	6, 4, 17, 7, 2, 20, 16, 18, 11, 21, 12, 19, 22, 5, 23, 3, 15, 8

4. EXPERIMENTS AND RESULTS

As we mentioned, the goal of using quality aware model for adaptive streaming is to allow for optimal bandwidth consumption. Bandwidth reduction can be caused by variable network state or enforced artificially by either service provider or client. Service provider can be forced to use reduction in busy hours when bandwidth consumption peaks. It can also decide to enforce reduction for users that spend certain amount of agreed bandwidth. This is common practice in today's agreements between users and service providers both for home and mobile broadband connections. User can decide to reduce bitrate in order to save bandwidth consumption that is usually limited to some value on a monthly basis. Only by using quality aware optimization it can be guaranteed that the least amount of quality is lost by reducing specified bitrate. To confirm this statement we study two use cases. First one is optimal case in which content creator is provides a quality metric for each segment. This quality metric (Q) can be calculated based on SSIM index or some other appropriate quality metric. Content creator can also use adjustment coefficient c in order to express subjective suggestion about importance of some interval in the sequence. Quality parameter is calculated for every segment as:

$$Q = c * averageSSIM \tag{1}$$

This parameter can be incorporated either inside manifest file or by supplying additional file. In current DASH specification the parameter can be added as attribute on segment level and specified in schema as:

```
<xs:attribute name=”Q” type=”xs:double”>
```

Example of portion of manifest file using this schema addition is shown in following listing:

```
<Representation group="3" bandwidth="249967">
  <SegmentInfo duration="PT10.00S">
    <InitialisationSegmentURL sourceURL="http://url.com/init.ts"/>
    <Url sourceURL="http://example.com/seg1.ts" Q="0.9556" />
    <Url sourceURL="http://example.com/seg2.ts" Q="0.9560" />
    <Url sourceURL="http://example.com/seg3.ts" Q="0.9447" />
    <Url sourceURL="http://example.com/seg4.ts" Q="0.9544" />
    <!-- etc. -->
  </SegmentInfo>
</Representation>
```

Adaptation logic on client side would include parsing of additional parameter *Q*. Client can decide on switching up or down based on achieved bitrate savings and resulting quality loss. Adaptation logic can construct optimal adaptation path across segments using simple algorithm:

```
for i:1 to numOfSegments
  if higher deltaQ / deltaB > thresholdUp
     switchUp(Bitrate)
  else if lower deltaQ / deltaB < thresholdDown
      switchDown(Bitrate)
  endif
next i
```

Here deltaQ (ΔQ) represents difference between *Q* of a segment encoded at higher bitrate and the one encoded at bitrate one level below. Thresholds for Qup and Qdown are selected according to user preferences and can be dynamically changed during adaptation in order to match quality as perceived by user. We can characterize decision process by using:

$$K = \Delta Q / \Delta B \tag{2}$$

where ΔB is bandwidth consumption difference between segments encoded at two consecutive bitrate levels. Optimization for both switching up and down can be expressed by *K* in following manner: if we are switching up we should take decision that maximizes *K*, when switching down we should be guided by minimized *K*. In other words, when we decide to switch up we should do it so that quality is noticeably increased by spending as little extra bandwidth as possible. When switching down ideal scenario is to lose little quality but save as much bandwidth as possible.

We used 10 test sequences with following characteristics: resolution 720x480p, format YUV 4:2:0, duration 360 seconds, framerate 25 fps. All sequences are clips from movies. We used x264 encoder[15] to produce H.264 AVC baseline and main profile bit streams that are further segmented into TS files of 2 and 10 second durations. Selected bitrates for representations are 500, 600, 800, 1100 and 1500 kbps. We chose these values according to Verizon's report of bandwidth available for 3G users across US. We implemented server/client model according to DASH specification. Segments are requested by one user at a time. We ran two sets of experiments – in first we used manifest file with additional attribute *Q* and second was implemented by using only QP information extracted from compressed sequences. In all experiments available bandwidth is always above highest requested bitrate in order to simplify the implementation.

4.1 Results for first set of experiments

In a first set of experiments client adaptation logic parsed manifest attribute *Q* and decided on switching according to threshold values that are set by user. We ran experiments for 4 settings:

Setting 1 – medium value for thresholdUp , low value for thresholdDown

Setting 2 – high value for thresholdUp, low value for thresholdDown

Setting 3 – medium value for thresholdUp , medium value for thresholdDown

Setting 4 – high value for thresholdUp, medium value for thresholdDown

Specific values for low, medium and high are not normative and can be adjusted according to user preferences. For ΔQ values we selected: low = 0.0025, medium = 0.005 and high = 0.0075. Values of ΔB are calculated based on bitrate differences of 100, 200, 300 and 400 kbps.

Results of experiments are shown in Table 2. Those are aggregated results for all 4 settings for all test sequences. Results are shown as comparison to selection in which user always downloads the highest bitrate segment. Bandwidth savings are presented as average percentage of savings across all sequences and quality loss is presented as average SSIM index value that is lost per switched segment.

We can see that significant bandwidth savings can be achieved by losing relatively small amount of quality and that this can be done by user preference. Same algorithm can be implemented on the proxy server by network provider in order to limit the bandwidth in quality aware manner. This way provider ensures that bandwidth savings are enforced exactly on segments that affect overall quality in the smallest possible amount.

This method of quality representation (inside original manifest file with specified quality parameter) is a preferable one as it allows for fine optimization on the user end. It produces very small overhead both in terms of additional bits for manifest file and encoder implementation, since automatic calculation of objective quality metrics is included in a workflow of almost every modern encoder. However, if this information is not present (or we are accessing the content that is prepared by older recommendations) we have to rely on QP extraction on the server side.

Table 1. Bandwidth savings and average quality loss across test sequences for specified settings.

Setting	Bandwidth savings	Average quality loss
1	15%	0.0021
2	18%	0.0025
3	22%	0.0031
4	25%	0.0037

4.2 Results for second set of experiments

In the case where quality parameter is not calculated by original content creator, there is no way to obtain reliable and precise quality metric because the reference sequence is not available. That's why we have to rely on compressed domain information extraction. As we already shown in section 3, there is strong correlation between average QPs on segment level and average quality differences between segments at discrete bitrate levels. We tested same set of sequences in different scenario – we used QP information from compressed sequence in order to select the candidate segments for bitrate reduction. We considered only the case where user wants to reduce the bitrate for specified number of segments in order to save bandwidth. In our scenario user decides to reduce bitrate for half of the segments in every sequence. We conducted experiments where 18 out of 36 segments are selected by our (QP) method and other approach which selects 18 segments at random. Manifest file and original sequences are unaltered in this case. Process of QP extraction can be done on a proxy server by network provider.

We ran three different experiments that reflect scenarios: experiment 1 – user decides to reduce segment bitrate always by only one level, experiment 2 – user decides to reduce segment bitrate always by two levels and experiment 3 – user uses both one and two levels for bitrate reduction.

Results are shown in Figure 7. Graph shows comparison between average quality losses in our (optimized) switching model and randomized switching which is a close approximation of network scenario when quality aware optimization is not implemented. In both cases, approximately same amount of bandwidth is reduced. This shows that additional bandwidth can be saved at a same quality cost if our model is applied.

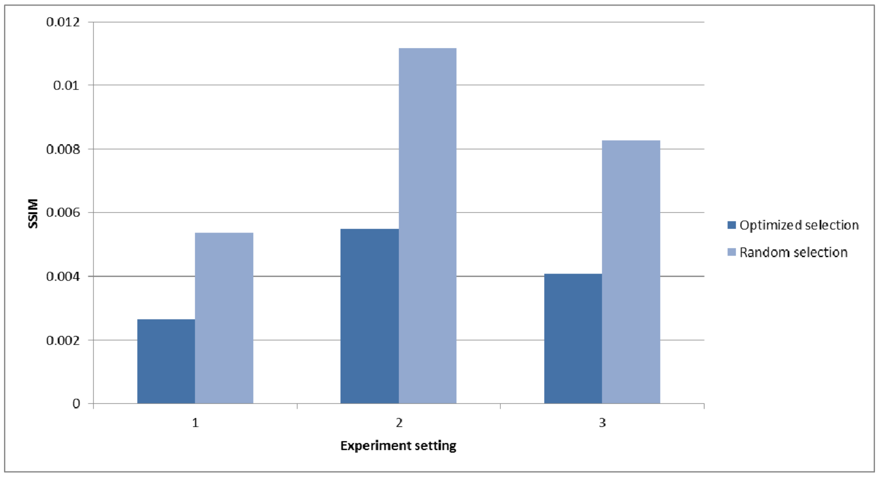

Figure 7. Average SSIM quality loss per switched segments for 3 experiments comparing optimized and random selection.

5. CONCLUSIONS

We presented quality aware model which can be implemented for adaptive streaming over HTTP. Experiments confirmed that using such model can introduce significant bandwidth savings. These savings cannot be overlooked when we have in mind rapid growth of mobile video streaming traffic. Our model also allows users and network providers to enforce bandwidth reduction at minimal quality cost. Model is flexible and can be implemented either as extension of current specifications or can be implemented on a proxy server. It can be applied both on explicitly specified parameters and by extracting QP information from compressed domain. The cost of implementation is insignificant since the encoding/extraction process is done only once per video sequence.

REFERENCES

[1] H1 2011 Allot Mobile Trends Report, http://www.allot.com/MobileTrends_Report_H1_2011.html (2011).
[2] Sandvine Intelligent Broadband Networks, “Internet Phenomena report”, Fall 2010, http://www.sandvine.com (2010).
[3] Zambelli, A. Smooth Streaming Technical Overview. http://learn.iis.net/page.aspx/626/smooth-streaming-technical-overview (2010).
[4] Using Adobe HTTP Dynamic Streaming. http://help.adobe.com/en_US/HTTPStreaming/1.0/Using/index.html (2010).
[5] Pantos, R. (Ed). HTTP Live Streaming draft. http://tools.ietf.org/html/draft-pantos-http-live-streaming-06, (2011).
[6] Stockhammer T., Fröjdh P., Sodagar I., Rhyu S.,(ed.) “Information technology — MPEG systems technologies — Part 6: Dynamic adaptive streaming over HTTP (DASH)”,ISO/IEC, MPEG Draft International Standard (2011).
[7] 3GPP TS 26.234, Transparent end-to-end packet switched streaming service (PSS); Protocols and codecs, http://www.3gpp.org/ftp/Specs/html-info/26234.htm (2011).
[8] Van Deursen, D., Van Lancker, W., Van de Walle, R. , "On media delivery protocols in the Web," Multimedia and Expo (ICME), 2010 IEEE International Conference on , vol., no., pp.1028-1033 (2010).

[9] H. Riiser, P. Halvorsen, C. Griwodz, and D. Johansen, Low overhead container format for adaptive streaming. In Proceedings of the first annual ACM SIGMM conference on Multimedia systems). ACM, New York, NY, USA, 193-198. (2010).

[10] Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj, Rate adaptation for adaptive HTTP streaming. In Proceedings of the second annual ACM conference on Multimedia systems (MMSys '11). ACM, New York, NY, USA, 169-174.(2011).

[11] Yago Sánchez de la Fuente, Thomas Schierl, Cornelius Hellge, Thomas Wiegand, Dohy Hong, Danny De Vleeschauwer, Werner Van Leekwijck, and Yannick Le Louéde, iDASH: improved dynamic adaptive streaming over HTTP using scalable video coding. In Proceedings of the second annual ACM conference on Multimedia systems (MMSys '11). ACM, New York, NY, USA, 257-264.(2011)

[12] Van der Auwera, G.; David, P.; Reisslein, M.; , "Traffic characteristics of H.264/AVC variable bit rate video," Communications Magazine, IEEE , vol.46, no.11, pp.164-174. (2008).

[13] Nicola Cranley, Philip Perry, Liam Murphy, User perception of adapting video quality, International Journal of Human-Computer Studies, Volume 64, Issue 8, Pages 637-647. (2006).

[14] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, (2004).

[15] X264 Encoder, official web page, http://x264.nl