QUALITY-AWARE HTTP ADAPTIVE STREAMING

Ali C. Begen

Cisco, Canada

ABSTRACT

In conventional HTTP adaptive streaming (HAS), a video content is encoded into multiple representations, and each of these representations is temporally segmented into small pieces. A streaming client makes its segment selections among the available representations mostly according to the measured network bandwidth and buffer fullness. This greatly simplifies the adaptation algorithm design, however, it does not optimize the viewer quality experience. Through experiments, we show that quality fluctuation is a common problem in HAS systems. Frequent and substantial fluctuations in quality are undesired and cause dissatisfaction, leading to revenue loss in both managed and unmanaged video services.

In this paper, we argue that the impact of such quality fluctuations can be reduced if the natural variability of video content is taken into consideration. First, we examine the problem in detail and lay out a number of requirements to make such system work in practice. Then, we study the design of a client rate adaptation algorithm that yields consistent video quality even under varying network conditions. We show several results from experiments with a prototype solution.

Our approach is an important step towards quality-aware HAS systems. A production-grade solution, however, needs better quality models for adaptive streaming. From this viewpoint, our study also brings up important questions for the community.

INTRODUCTION

In an HAS system, a single master high-quality video source is transcoded into several streams, each with a different bitrate and/or resolution. These streams are called representations, and each of the representations is temporally chopped into short chunks of a few seconds, where each chunk is generally independently decodable. These content chunks are called segments. The MPEG's Dynamic Adaptive Streaming over HTTP (DASH) standard [1] supports both physical and virtual segmentation of the representations. The generated segments and the manifest file that provides detailed information about the content features, representations and segments are posted on an HTTP server. The streaming clients then use the HTTP GET requests to fetch the manifest file and the segments from the server.

Segmentation is essential to support live (linear) content delivery, as HTTP is an object delivery protocol and the use of segmentation allows us to generate the segments (i.e., objects) in real time. If the content is not segmented, it cannot be delivered linearly. In this case, the content can only be delivered after it is finalized, and one can use the traditional progressive download method to download the entire content with only one HTTP GET. When segmentation is done for multiple representations, the streaming client also gets the capability of adapting. At segment boundaries, the client can decide to stick with the same representation, fetch a segment from a higher quality representation, or fetch a segment from a lower quality representation. The decision depends on a number of factors including, but are not limited to, server/network performance, client conditions and the amount of data available in client's buffer. Typically, tens of seconds of content is buffered at the client to accommodate unpredictability of the network conditions.

A common practice among the streaming providers has been to run the transcoders in the constant bitrate (CBR) mode, resulting in more or less equal segment sizes and durations for a given representation. That is, all the equal-duration segments produced out of a single representation would have the same size in terms of bytes. This simplified the rate adaptation algorithms for the client developers and helped them minimize the chance of a buffer underrun. However, most content types are not suitable for CBR encoding since they exhibit a large variation in complexity in the temporal domain. Thus, during a session, even if the network conditions do not change, a naïve client that streams CBR-encoded segments may easily experience quality variation when switching between a high-motion or high-complexity (e.g., explosion) scene and a low-motion or low-complexity (e.g., static) scene.

For example, in Figure 1, we show two screenshots from a decoded video of an HAS session streamed over a constant-bandwidth link. The first screenshot is from the preview title, which is static and has low complexity. The second one is from a fairly complex and dynamic scene. Not surprisingly, CBR-encoded segments yield a much lower visual quality in the bottom screenshot than the above one.

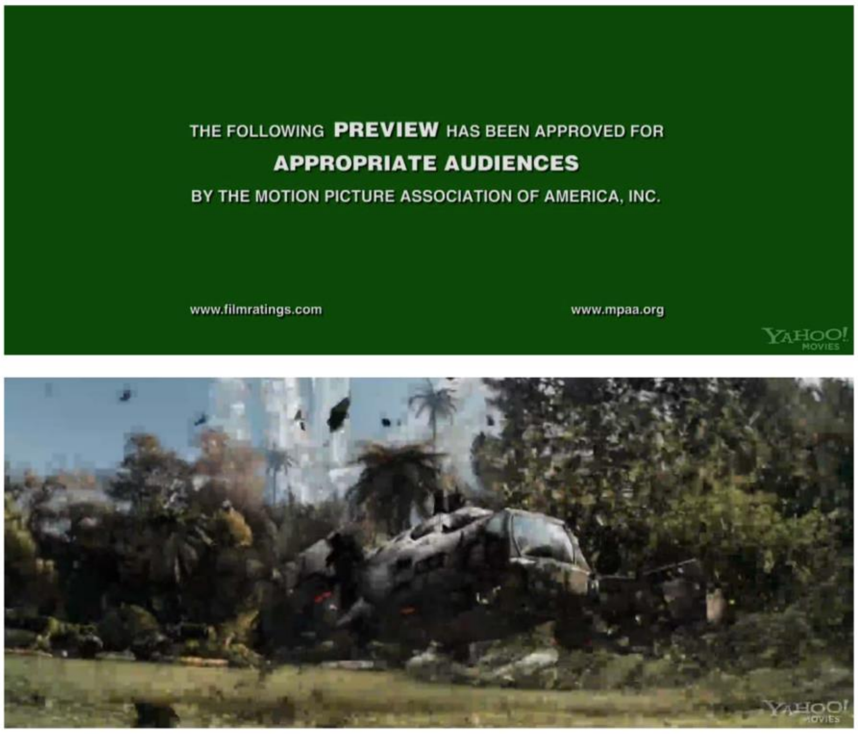
THE FOLLOWING PREVIEW HAS BEEN APPROVED FOR
APPROPRIATE AUDIENCES
BY THE MOTION PICTURE ASSOCIATION OF AMERICA, INC.
www.filmratings.com
www.mpaa.org
YAHOO! MOVIES
YAHOO! MOVIES

Figure 1 – Two screenshots from the decoded video of an HAS session using CBR-encoded segments.

With the introduction of rate adaptation algorithms that were capable of dealing with varying segment encoding bitrates (and sizes), some providers started encoding their videos in the variable bitrate (VBR) mode. This helped alleviate the problem of wild quality fluctuations to some extent, thus, improved the viewer experience. However, more advanced rate adaptation algorithms have to be developed to achieve the full potential of quality-aware adaptive streaming. Recently, we introduced a new concept, called consistent-quality streaming [2], which formulated quality-aware adaptive streaming as an optimization problem that temporally allocates bits among the video segments to yield an optimal overall quality. We laid out a mathematical framework to find a practical solution to this complex optimization problem. A prototype was developed and initial results were encouraging.

In this paper, we briefly explain the concept of consistent-quality streaming and provide results. More importantly, we discuss the current challenges and provide an overview of the open issues.

WHAT DOES IT MEAN TO BE QUALITY-AWARE?

Suppose that we generate the video segments using VBR encoding. Strictly speaking, it has to be a capped VBR encoding, which means that there is a limit in practice to how many bits the encoder can steal from simple-scene segments and use them in complex-scene segments while trying to maintain a more stable encoding quality. Also suppose that we have a representative metric that can capture the presentation quality of a given segment and this metric is computed for every segment of each representation. If the collective quality information is conveyed to the streaming client (either through in the manifest file or in an out-of-band manner), a quality-aware client implementation can take this information into account and make better decisions in rate adaptation. At a decision epoch, the client essentially has to pick a particular segment from the available representations such that the rendered quality will be the highest based on the adopted temporal pooling model for the video quality.

Temporal pooling is a process that attempts to measure the overall quality of a content piece that consists of a sequence of segments. While there is no one common temporal pooling model that will work for every viewer or any type of video content, a model that works for most of the scenarios should be used. For example, overall impression of a viewer towards a video has been shown to be greatly influenced by the single most severe event while the duration is neglected [3]. This is likely to be the case for most viewers, but not necessarily for all viewers.

There is a time-varying bandwidth constraint in any real-time streaming application. Yet, in quality-aware streaming, we have two more constraints:

- First, the optimization is myopic – it does not know the available bandwidth in the future. Furthermore, in the case of live streaming, the client can have visibility only into a few upcoming video segments (including both bitrate and quality information) to preserve the liveness of the content.
- Second, we make use of the client-side video buffer as a breathing room for video bitrate variability, in a way that the buffer should neither be completely drained nor filled above a threshold. If the buffer is completely drained, the playout will stall, which is probably the worst event for the viewer experience. Also, to accommodate bandwidth variability, the client buffer size should be bounded above some

minimum level (for example, several segments). On the other hand, due to the desire to keep the end-to-end latency in live streaming small, or device memory limit, or simply economic reasons, the buffer size should also be bounded below some maximum level.

Under these constraints, [2] developed a solution that combines an online algorithm with dynamic programming. The online algorithm adapts the video bitrate step-by-step, and at each step it uses dynamic programming to solve a constrained optimization sub-problem within a sliding window. The dynamic programming solution allows us to turn a combinatorial problem into something solvable in polynomial time. To our advantage is that the available bitrate is a discrete value in HAS applications, so, it well fits into the dynamic programming framework.

It is worth noting that, this solution is a component inside the rate adaptation algorithm at the client side. It is fully orthogonal to video encoding. For example, in principle, it works with either CBR or VBR-encoded video. The only needed architectural change is to convey the video quality information to the client in some way.

A Simple Demonstration

Assume that one-second video content has already been downloaded and buffered at a client. The client is now trying to decide which video segment to fetch next. It knows the quality and bitrate information of the video segments for the current step and one step ahead, and there are two representations to choose from. It also has the information of the current available bandwidth. Assuming that the bandwidth will not change in the near future, the client can precisely calculate the evolution of the buffer at the end of each step for any specific segment that is fetched.

At the current step, the client has two choices: if downloading the low-quality segment, the buffer gain is half a second and the resulting segment quality is 1; if downloading the high-quality segment, the buffer loss is half a second and the quality is 2. Similarly, at the next step, downloading the low-quality segment would result in buffer gain of 0.4 seconds and segment quality of 2, and downloading the high-quality segment would result in buffer loss of 0.7 seconds and segment quality of 4. Figure 2 illustrates all the possible selections and the resulting size of the client buffer at the end of each step (in orange boxes).

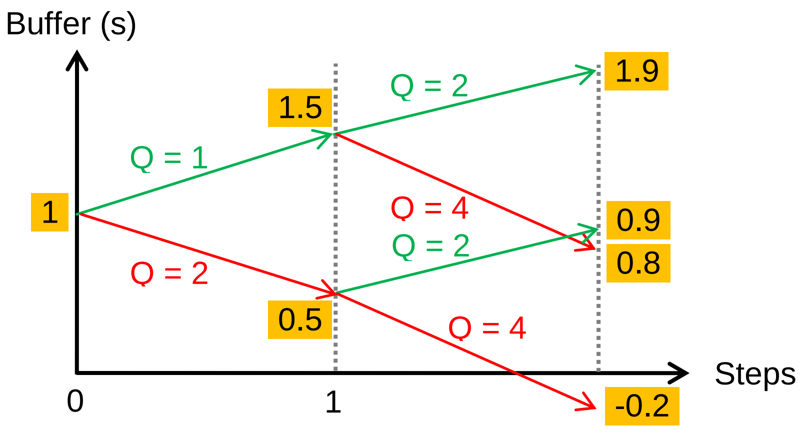

Figure 2 – The two-step decision tree example for a quality-aware streaming client. At each step, the client can choose either the low or the high-quality segment.

Assume that at the end of the second step, we do not want the buffer to fully drain. Thus, selecting the high-quality segment option in both steps is not viable. Out of three remaining possible decisions:

1. If the objective is to maximize the minimum quality out of the two segments, the client should select the high-quality and low-quality segments in the first and second steps, respectively, yielding best minimum quality of 2.

2. If the objective is to maximize the total quality of the two segments, the client should select the low-quality and high-quality segments in the first and second steps, respectively, yielding best total quality of 5.

3. If the objective is to increase the buffer size, the client should select the low-quality segments in both steps.

While the client makes a decision for multiple steps at every decision epoch, it has to recalculate its decision at the next step with any new bandwidth and quality information. Readers interested in the mathematical formulation of consistent-quality streaming and its solution should refer to [2] for details.

RESULTS

In our earlier work we designed a prototype and tested it under various conditions and with different variations of the objective functions. Here, we would like to provide a few experimental results to demonstrate how much improvement can be achieved over conventional quality-unaware streaming.

Test Setup

We used two video sources for our evaluation. The first one is a two-minute long 720p Elysium trailer crawled from YouTube [4]. The second one is a twelve-minute long 1080p clip extracted from the movie Avatar. The segment duration was two seconds. The Elysium trailer was encoded into seven representations at bitrates of ranging from 400 to 3200 Kbps, and the Avatar clip into 11 representations at bitrates of ranging from 400 to 9000 Kbps. The segments were generated using the CBR mode of the encoder to illustrate that quality-aware streaming did not require the video to be VBR-encoded, and also to conduct a fair comparison with the conventional adaptation schemes. Using VBR encoding would further enhance the visual quality.

To measure the video quality, we used the negation of mean-squared error (MSE) value for each segment for its simplicity. In the plots below, the MSE values are converted to peak signal-to-noise ratio (PSNR) for better display. These plots compare the following three algorithms:

1. Unaware: This algorithm is quality-unaware and considers only the bitrate information in its decisions.

2. Max-min: This algorithm tries to maximize the minimal quality.

3. Max-mean: This algorithm tries to maximize the mean quality (equivalently, the total quality).

Comparisons

Figure 3 and Figure 4 show the traces of the quality and the bitrate of the fetched segments for the three algorithms for the Elysium trailer and Avatar clip, respectively. From the quality trace, we observe that the two algorithms trying to optimize the quality yield much better quality than the quality-unaware algorithm, both in terms of mean quality and minimal quality.

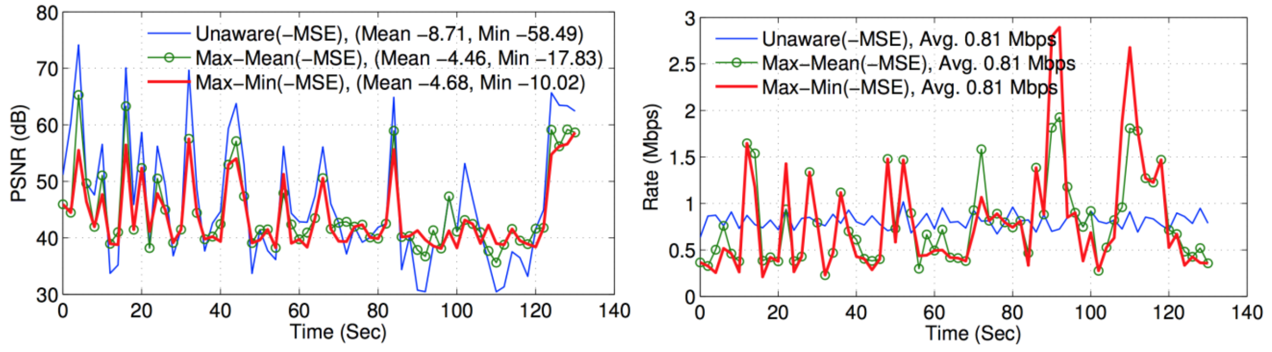

Figure 3 – Comparing the traces of three algorithms in terms of quality (on the left) and bitrate (on the right) for the Elysium trailer.

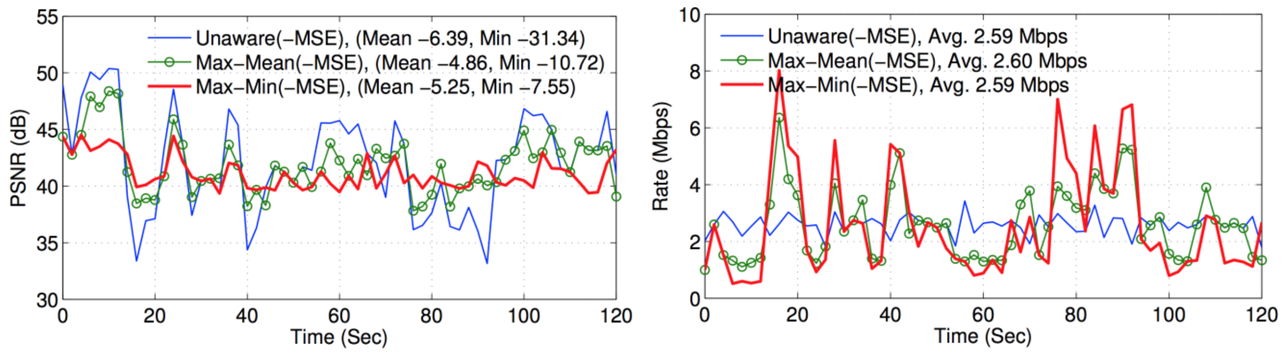

Figure 4 – Comparing the traces of three algorithms in terms of quality (on the left) and bitrate (on the right) for the Avatar clip.

The outputs from the Elysium trailer experiments are posted on a public Web site for visual inspection. Refer to [4] for details.

CHALLENGES AND OPEN ISSUES

In this section, we discuss some of the current challenges and provide an overview of the open issues. It is important for the research community and the video industry to tackle with these issues to successfully deploy quality-aware adaptive streaming systems.

Development of Quality Metrics and Temporal Pooling Models

One can solve an optimization problem better if its objective function is well defined. This implies that if we can use a better metric for measuring the quality and a better temporal pooling model for computing the visual experience over a number of segments whose qualities are known, we can improve the solution to our optimization problem, leading to a better viewer experience. Much effort has been spent in researching and standardizing a video quality metric for streaming applications, and there are still efforts underway. Metrics such as MSE/PSNR, Sarnoff's Picture Quality Metrics (PQR), Multiscale Structural Similarity (MS-SSIM), Video Quality Model (VQM) and Spatio-temporal Reduced Reference Entropic Differences (STRRED) have been examined in detail (References for these metrics can be found in [2]). However, there is still no wide agreement on a common metric that is suitable for a variety of content types.

The nature of adaptive streaming applications is quite different than the traditional broadcast TV. Thus, even if we develop a good quality metric, it is a whole new problem to develop a good temporal pooling model that will reliably work for different content types, viewer profiles, devices and networks. Substantial further investigations are needed in this area.

Integration into Popular Streaming Client Implementations

The proliferation of different schemes for adaptive streaming is a big concern to content and service providers who would like to support all popular platforms in their services. Different mobile, computing, connected TV and gaming ecosystems support different features and mostly require unique implementations tailored for them. While developing a new quality-aware streaming client is already a serious effort, integrating that implementation into different platforms and achieving a consistent experience across all the platforms is a way more challenging undertaking.

As the unifying standard, MPEG DASH is foreseen as the key to solving these integration issues, however, closed environments where the developers are bound to certain system-dependent libraries and requirements (such as Apple's iOS) will likely continue to pose difficulties in the short term.

Development of Metadata Standards

Producing the quality information for a large number of segments in real time is a challenge. Conveying this information to a large number of streaming clients in a timely and scalable manner is an equally important challenge. MPEG is currently developing a new specification that describes how to carry timed metadata in the ISO base media file format [5]. This work, which is planned to be completed in late 2015 or early 2016, can be used to distribute the quality information in a scalable way to a large number of clients.

Expansion to Multi-Client Scenarios

In this paper, we focused on the single-client optimization where the client tried to optimize its own quality. However, prior research and experience from deployed systems have shown that when two or more adaptive streaming clients compete for the bottleneck bandwidth, the selfish nature of adaptive streaming leads to unwanted mutual interactions between the clients, and the clients suffer from bitrate oscillations [6]. There have been several proposals in the past few years that defined fairness in terms of having a fair share of the network bandwidth and solved the oscillation problem based on this specific fairness definition. Yet, fairness should better be defined as "quality fairness" not "bitrate fairness." For example, one of the two clients sharing the same access network is streaming a live soccer game, while the other one is streaming a newscast with a talking head only. A prudent strategy here is to allocate more bandwidth to the first client as the soccer game needs more bits for encoding to achieve the same level of quality with the newscast. Naturally, determining the right strategy even in such a simple scenario is hard. What if the second client is streaming the newscast onto a 60" TV screen where 10 viewers are watching it and the first client is streaming the game onto a tiny mobile phone screen only for individual consumption. The optimization across a number of streaming clients has to be done based on the utilities of the streamed videos, which depend on several factors such as the content type, content features, rendering device, audience profile and size. By the help of the ongoing standardization of Server and Network-assisted DASH (SAND) [7], controlled unfairness may be possible to deploy and then we may be able to increase the scope of quality-aware optimization from a single client to multiple clients.

CONCLUSION

In this paper, we explained the concept of consistent-quality streaming and provided encouraging results. We also discussed some of the current challenges and presented the open issues for the research community and the video industry to tackle with. We hope that this paper will be successful in raising the community and industry interest in quality-aware streaming.

REFERENCES

[1] MPEG, Dynamic Adaptive Streaming over HTTP (DASH), ISO/IEC 23009-1, 2012

[2] Zhi Li, Ali C. Begen, Joshua Gahm, Yufeng Shan, Bruce Osler and David Oran, "Streaming video over HTTP with consistent quality," in Proc. ACM Multimedia Systems Conf. (MMSys), Singapore, Mar. 2014

[3] J. Park, K. Seshadrinathan, S. Lee, and A.C. Bovik, "Video quality pooling adaptive to perceptual distortion severity," in IEEE Trans. Image Processing, 22(2):610–620, 2013

[4] Source: https://sites.google.com/site/cqhttpstreaming/

[5] MPEG, Timed Metadata Metrics of Media in the ISO Base Media File Format, ISO/IEC 23001-10, 2015 (Work in progress)

[6] Saamer Akhshabi, Ali C. Begen and Constantine Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," Proceedings of ACM Multimedia Systems Conf. (MMSys), San Jose, CA, Feb. 2011

[7] MPEG, Server and Network-assisted DASH, ISO/IEC 23009-5, 2015 (Work in progress)

ACKNOWLEDGEMENTS

The author would like to thank his co-authors for their contributions to the paper that introduced the concept of consistent-quality streaming, which has appeared in the proceedings of the ACM Multimedia Systems (MMSys) Conference held in Singapore in 2014.