Use of machine learning for rate adaptation in MPEG-DASH for quality of experience improvement

Alzahrani, Ibrahim; Ramzan, Naeem; Katsigiannis, Stamos; Amira, Abbes

Published in:
5th International Symposium on Data Mining Applications (SDMA 2018)

DOI:
10.1007/978-3-319-78753-4_1

E-pub ahead of print: 29/03/2018

Document Version
Peer reviewed version

Use of Machine Learning for Rate Adaptation in MPEG-DASH for Quality of Experience Improvement

Ibrahim Rizqallah Alzahrani, Naeem Ramzan, Stamos Katsigiannis, and Abbes Amira

School of Engineering and Computing, University of the West of Scotland
Paisley, PA1 2BE, United Kingdom
{Ibrahim.Alzahrani, Naeem.Ramzan, Stamos.Katsigiannis, Abbes.Amira}@uws.ac.uk

Abstract. Dynamic adaptive video streaming over HTTP (DASH) has been developed as one of the most suitable technologies for the transmission of live and on-demand audio and video content over any IP network. In this work, we propose a machine learning-based method for selecting the optimal target quality, in terms of bitrate, for video streaming through an MPEG-DASH server. The proposed method takes into consideration both the bandwidth availability and the client's buffer state, as well as the bitrate of each video segment, in order to choose the best available quality/bitrate. The primary purpose of using machine learning for the adaptation is to let clients know/learn about the environment in a supervised manner. By doing this, the efficiency of the rate adaptation can be improved, thus leading to better requests for video representations. Run-time complexity would be minimized, thus improving QoE. The experimental evaluation of the proposed approach showed that the optimal target quality could be predicted with an accuracy of 79 %, demonstrating its potential.

Keywords: MPEG-DASH, machine learning, rate adaptation, QoE, video streaming

1 Introduction

Video has become one of the most important media elements across the entertainment and communications industries. It is expected that in 2021, over than 3/4 of Internet data will be video [1]. As a result, extensive research is being conducted in order to better understand and sort out the complexities of time variations, unstable bandwidth, and end-to-end or startup delays during transmission.

The current video streaming market is dominated by three primary providers: Microsoft (smooth-streaming) [2], Adobe (HTTP-dynamic streaming) [3], and Apple (HTTP-live streaming) [4]. Each of these companies has its own content format, encoding, and protocol that are firmly tied to that specific vendor, a

fact that often leads to video content being prepared for delivery in these three different formats. Thus, many challenges are inherent in producing, encoding, and delivering video streams, but a solution must be provided for all vendors in the market because multimedia streaming is rapidly becoming the dominant media form [1]. Technology such as MPEG-DASH [5] provides a viable solution to the above issue by allowing inter-operability between various electronic devices and servers. The MPEG-DASH client accesses services from the web/media server via the framework described in [6]. At the server, encoded versions of the video are produced and then chunked into smaller segments that are all the same size. These segments are requested by the client via HTTP GET or partial GET requests, but show up as of the same video despite differing methods of transmission, ensuring an acceptable Quality-of-Experience (QoE).

In this work, we propose a machine learning-based method for selecting the optimal target quality, in terms of bitrate, for video streaming through an MPEG-DASH server. The proposed method takes into consideration both the bandwidth availability and the client's buffer state, as well as the bitrate of each video segment, in order to choose the best available quality/bitrate. Choosing the optimal target bitrate will allow the client to make better rate adaptation choices, thus enhancing QoE on the client's end.

The rest of this paper is organised in four sections. Section 2 provides an outline of the related published research, while the proposed method is described in Section 3. The results and discussion of the experimental evaluation are provided in Section 4 and conclusions are drawn in Section 5.

2 Related work

The rate adaptation of Dynamic Adaptive video Streaming over HTTP (DASH) is usually based on the available bandwidth, the client's buffer, or some combination of the items above.

Petrangeli et al. [7] proposed a framework based on OpenFlow, that prioritises the delivery of specific video segments to avoid overfilling clients' buffers, a problem that may lead to the video freezing. They based their method on a machine learning approach, which relies on the RUSBoost algorithm (which is used for detecting whether a client is about to encounter a freeze) and fuzzy logic (which decides whether the circumstances of the prioritisation queue are proper to efficiently prioritise the segment). Petrangeli et al.'s proposal [7] would notice if a client is about to encounter a freeze, and drive the network prioritisation to overcome it. They carried out an extensive performance comparison of various video streaming scenarios with other HAS solutions (i.e. heuristics FINEAS and MSS). Their conclusion showed that the proposed method could decrease the video freezes by about 65% and freeze time by about 45%. However, their proposed method is based on the collected network measurements, and ignores both the information about the streamed videos and the clients' system characteristics.

Chan et al. [8] proposed a rate algorithm to improve the efficiency of bandwidth utilisation based on the transport-layer information, which is explored as a means of estimating the buffer size on the clients end. The proposed algorithm showed better results compared to Apple HLS [4] and Microsoft smooth streaming [2]. However, the proposed algorithm lacks the accuracy and up-to-date details of the adaptation process that are available with additional HTTP requests. Schemes based on application layer are techniques used for more than just the prediction of the possible throughput in addition, a number of attempts were proposed on cross-layer throughput estimation. A solution based on machine learning (Support Vector Regression [9]) developed based on [10] is used for the prediction of the possible throughput in [11]. This method trains the throughput prediction via information from the network layer, such as packet loss, delay, and RTT.

Xiong et al. [12] made another argument about the clients' overall perceptions when video quality is changed. They note that the perceived video quality is not easy to describe in an accurate language, or even in other mathematical models that depend on an exact input and output definition. Because of this, they propose using fuzzy logic (called Network-Bandwidth-Aware Streaming Version Switcher), which is divided into three components, namely sensor, controller, and actuator. The sensor works like an estimation resource module, the controller is responsible for the adaptation, and the actuator monitors the controller's decisions. Xiong et al. [12] concluded that the technique was responsive to changes in the network. In [13], fuzzy logic was applied to adapt the video rate to changes in the network's conditions. The proposed algorithm aims to avoid buffer overflow and unnecessary fluctuations in video quality. However, it also suffers from a large variation during changes in video quality. In order to overcome this limitation, Sobhani et al. [14] proposed an AIMD-like fuzzy controller. Their proposed method focuses on both the buffer occupancy and the estimated throughput. Nevertheless, fuzzy logic would require domain expert knowledge, which can be difficult to acquire.

Chien et al. [15] proposed the use of a decision tree in order to map network-related features onto the video rate. Instead of introducing a new algorithm, a scheme was chosen to improve the accuracy of existing adaptation algorithms. In short, these researchers trained the model via a given dataset [16]. The classifier then predicts the current/future requests. Hence, the training process can be applied either on-line or off-line, and the results showed improved prediction accuracy.

Bhat and Bhadu [17] proposed a machine learning approach for video streaming with DASH to help clients adapt to changes in the streaming environment. Their objective was to allow clients to learn about the environment in an unsupervised manner, since they assumed that by doing this, the redundancy of adaptation would be eliminated. They concluded that they could improve QoE and efficiency of bandwidth utilization by up to 68.5%. Van der Hooft et al. [18] used the mean and the average absolute difference in bandwidth, whereas Claeys et al. in [19] and [20] used the changes information about both the avail-

Table 1. DASH-server and DASH-client hardware specification

	DASH-server	**DASH-client**
Processor model	Intel Xeon E3-1245	Intel Core i5-4570
Processor speed	3.40 GHz (8 cores)	3.20 GHZ (4 cores)
RAM	7.5 GB	4 GB
Operating system	Linux Ubuntu-14.04 LTS 64-bit	MS Windows 8.1- 64-bit

able bandwidth and the buffer state. Then, an agent works by changing the video rate to gradually increase its reward, such as enhancing the Mean Opinion Score (MOS) and decreasing the re-buffering [18]. Martín et al. [21] presented an adaptation algorithm based on the Q-Learning method, called DASH-QL. This approach is based on reinforcement learning [22], which allows clients to learn via experience. This method was able to perform an ideal control like other approaches such as DASH-SDP, while still maintaining adaptivity [21].

Due to long delays and fluctuations in bandwidth, the user's QoE needs to be enhanced constantly. One possible solution involves adaptation logic improvements, which would decrease re-buffering events when transmitting higher video qualities. Bezerra et al. [23] presented a control system (DBuffer) to improve the QoE while a streaming session simultaneously occurs in mobile networks. Their proposed method approximates the most appropriate video quality for specific requests based on the clients' buffer state, which is the main component of their proposed method that controls the adaptation logic [23]. They concluded that their proposed method showed the lowest stall events when comparing the conventional and PANDA adaptation approaches.

3 Proposed method

This section proposes a new approach that utilises machine learning algorithms for determining the best possible quality (in terms of bitrate) that a video streaming client could enjoy when video content is streamed from a DASH server. Current bitrate, buffer state, and the available bandwidth are used in order to determine the optimal target quality of the server, thus ensuring the best possible playback at the client and enhancing the QoE.

3.1 Experimental Setup & Data acquisition

A DASH-server and a DASH-client were used in order to stream video sequences and acquire network and quality statistics. The hardware specifications of the server and the client system used in this study are summarised in Table 1. An Apache HTTP server was used for the DASH-server system, while MP4Client, an open-source multimedia player from GPAC [24], was used as the DASH-client

Table 2. Video sequences used

Sequence name	Big_Buck_Bunny, Elephants_Dream, Site_sings_the_blues
Resolution	1920×1080 pixels
Frame rate	24 fps
Duration	5.45 min (345 sec)
Segment size (*sec*)	10, 15
Number of segments	23 (15 *sec* segments), 35 (10 *sec* segments)
Bitrate (kbps)	125, 250, 375, 500, 750, 1000, 1500, 2000

at the client system, since it allows clients to retrieve the most appropriate video content based on the given configurations. Furthermore, the bandwidth was capped at 2.5Mbps using the NetEm emulator [25] in order to create a realistic test environment for the DASH streaming system.

Raw video encoding is achieved by using the FFmpeg command line tool to produce different versions of each video sequence encoded with the H.265/HEVC codec [26] at the following eight average bitrates: 125, 250 375, 500, 750, 1000, 1500, and 2000 kbps, keeping the original resolution and frame rate. Then, the encoded videos are chunked into smaller segments of a pre-set duration using MP4Box [27]. In this study, segments of 10 and 15 seconds were created for each of the encoded video sequences. The produced segments are kept on the DASH-server for future retrieval by the DASH-clients. After dividing the encoded video sequences into segments, the Media Presentation Description (MPD) is created automatically for future use by the system. Every available representation of each video sequence (different bitrates, frame rates, resolutions, segments, codecs, etc.) is recorded on the MPD file. The DASH-client then uses the MPD file to choose the most appropriate video representation through HTTP-GET or Partial GET requests, based on the bandwidth conditions available during streaming and the DASH-client's buffer state. An overview of the MPEG-DASH system used in this study is provided in Fig. 1.

Three raw high definition (1080p at 24 fps) video sequences [28] with no copyright restrictions were used in this study. The duration of each video sequence was 5.45 minutes (345 *sec*) and each sequence was encoded and segmented as previously described. Details about the video sequences used are provided in Table 2. Then, simulating a Video on Demand (VoD) scenario, the examined video sequences were streamed to the DASH-client using the DASH-server, targeting the maximum possible quality.

3.2 Feature extraction & Classification

Information about the streamed content was retrieved by the DASH-client for all the available representations of the three examined video sequences (average

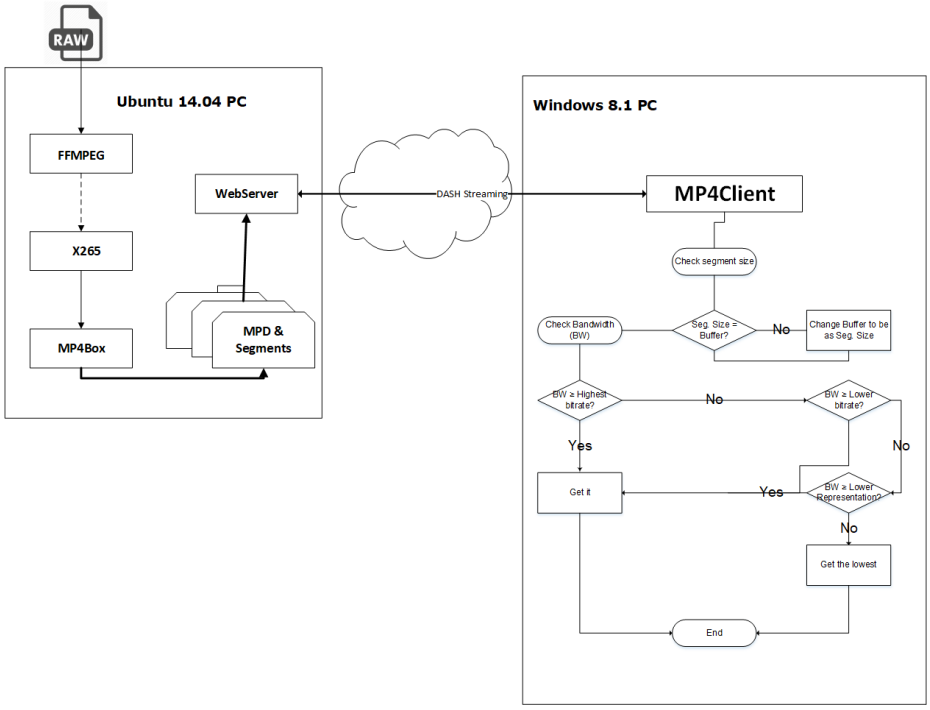

Fig. 1. Diagram of the examined MPEG-DASH system

bitrate and segment size). For each segment of each representation, the following information was recorded: the bitrate of played video in *bps* (not the original bitrate but the one served to the client), the actual available bandwidth in *bps*, and the DASH-Clients buffer size in *bytes*. Each triplet (bitrate, bandwidth, buffer size) was then used as a feature vector labelled with the original average bitrate of the examined segment (one of 125, 250 375, 500, 750, 1000, 1500, and 2000). This process led to 280 feature vectors (samples) for each video sequence when segmented with a segment duration of 10 *sec*, and to 184 feature vectors for each video sequence segmented with a segment duration of 15 *sec*. Finally, two set were created, one referring to 10 *sec* segments that had 840 samples and one referring to 15 *sec* segments that had 552 samples, with each sample belonging to one of the eight quality (bitrate) categories. Linear and non-linear classifiers were then used in order to create a machine learning model that can predict the quality class. The motive behind this approach is to assist the DASH-client in targeting the optimal quality level depending on the available resources of the client.

Table 3. Classification results for each segment size

Classifier	**Accuracy (%)**	
	10 *sec*	**15** *sec*
1-NN	71.2	61.4
3-NN	74.0	62.0
5-NN	75.2	62.3
10-NN	74.3	58.3
SVM (Linear)	76.5	47.6
SVM (RBF)	77.1	63.6
SVM (Quadratic)	**79.0**	**68.1**
Decision Trees	78.6	55.8
LDA	66.1	42.0

4 Experimental results

Multi-class supervised classification experiments were conducted in order to evaluate the performance of the proposed model. The examined classification algorithms were the k-Nearest Neighbour (k-NN) for $k = 1, 3, 5, 10$, the Linear Support Vector Machines (SVM), the SVM with RBF and quadratic kernels, the Decision Trees (DT), and Linear Discriminant Analysis (LDA). The one vs one method was used in order to achieve multi-class classification using the SVM classifiers. Furthermore, features were normalised before the classification experiments due to their difference in range, and 10-fold cross validation was employed in order to avoid over-fitting the examined models. The available MATLAB [29] implementations were used for all the classifiers. Results in terms of classification accuracy for each segment size are shown in Table 3. It is evident that the SVM with the quadratic kernel achieves the highest accuracy for both the examined segment sizes, reaching a 79 % accuracy for video sequences with a 10 *sec* segment size, and 68.1 % for a 15 *sec* segment size.

5 Conclusion

In this work, we proposed and evaluated a machine learning approach for selecting the optimal streaming quality for an MPEG-DASH video streaming server. Client-side features that included the current bitrate, the current bandwidth, and the current buffer size were used in order to create a machine learning model that is able to distinguish the target quality level in terms of the average bitrate used for encoding the video sequence with the H.265/HEVC codec. Supervised classification experiments showed that the SVM classifier with quadratic kernel function provided the highest classification accuracy, which reached 79 % when the video segment size is set to 10 *sec* and 68.1 % when set to 15 *sec*. As a result,

the proposed model could be successfully utilised within an MPEG-DASH client for selecting the optimal quality for each client based on the client's current capabilities, leading to enhanced overall QoE. Future work could evaluate the proposed approach for different segment sizes (e.g. 1, 2, 5 and 20 *sec*), as well as additional features related to the client's capabilities.

References

1. Cisco: Cisco visual networking index: Global mobile data traffic forecast update (2017). URL https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html. White Paper (Accessed 23 January 2018)
2. Microsoft Corporation: [MS-SSTR]: Smooth Streaming Protocol. V8.0 2017/09/15 (2015). URL https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-SSTR/[MS-SSTR].pdf. White Paper. (Accessed 2 October 2017)
3. Adobe Systems Inc.: High-quality, network-efficient HTTP streaming. URL https://www.adobe.com/products/hds-dynamic-streaming.html. (Accessed 21 June 2017)
4. Pantos, R., May, W.: HTTP Live Streaming. RFC 8216, RFC Editor (2017)
5. ISO/IEC: Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats. Standard, International Organization for Standardization (2014)
6. Oyman, O., Singh, S.: Quality of experience for http adaptive streaming services. IEEE Communications Magazine **50**(4), 20–27 (2012). DOI 10.1109/MCOM.2012.6178830
7. Petrangeli, S., Wu, T., Wauters, T., Huysegems, R., Bostoen, T., Turck, F.D.: A machine learning-based framework for preventing video freezes in HTTP adaptive streaming. Journal of Network and Computer Applications **94**, 78 – 92 (2017). DOI 10.1016/j.jnca.2017.07.009
8. Chan, K.M., Lee, J.Y.B.: Improving adaptive HTTP streaming performance with predictive transmission and cross-layer client buffer estimation. Multimedia Tools and Applications **75**(10), 5917–5937 (2016). DOI 10.1007/s11042-015-2556-y
9. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3), 273–297 (1995). DOI 10.1007/BF00994018
10. Mirza, M., Sommers, J., Barford, P., Zhu, X.: A machine learning approach to TCP throughput prediction. IEEE/ACM Transactions on Networking **18**(4), 1026–1039 (2010). DOI 10.1109/TNET.2009.2037812
11. Tian, G., Liu, Y.: Towards agile and smooth video adaptation in HTTP adaptive streaming. IEEE/ACM Transactions on Networking **24**(4), 2386–2399 (2016). DOI 10.1109/TNET.2015.2464700
12. Xiong, P., Shen, J., Wang, Q., Jayasinghe, D., Li, J., Pu, C.: NBS: A network-bandwidth-aware streaming version switcher for mobile streaming applications under fuzzy logic control. In: 2012 IEEE First International Conference on Mobile Services, pp. 48–55 (2012). DOI 10.1109/MobServ.2012.10
13. Vergados, D.J., Michalas, A., Sgora, A., Vergados, D.D.: A control-based algorithm for rate adaption in MPEG-DASH. In: IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications, pp. 438–442 (2014). DOI 10.1109/IISA.2014.6878834

14. Sobhani, A., Yassine, A., Shirmohammadi, S.: A fuzzy-based rate adaptation controller for DASH. In: Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV '15, pp. 31–36. ACM, New York, NY, USA (2015). DOI 10.1145/2736084.2736090
15. Chien, Y.L., Lin, K.C.J., Chen, M.S.: Machine learning based rate adaptation with elastic feature selection for HTTP-based streaming. In: 2015 IEEE International Conference on Multimedia and Expo (ICME), pp. 1–6 (2015). DOI 10.1109/ICME.2015.7177418
16. Basso, S., Servetti, A., Masala, E., De Martin, J.C.: Measuring DASH streaming performance from the end users perspective using Neubot. In: Proceedings of the 5th ACM Multimedia Systems Conference, MMSys '14, pp. 1–6. ACM, New York, NY, USA (2014). DOI 10.1145/2557642.2563671
17. Bhat, A.R., Bhadu, S.K.: Machine learning based rate adaptation in DASH to improve quality of experience. In: 2017 IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), pp. 82–89 (2017). DOI 10.1109/ICSTM.2017.8089131
18. van der Hooft, J., Petrangeli, S., Claeys, M., Famaey, J., Turck, F.D.: A learning-based algorithm for improved bandwidth-awareness of adaptive streaming clients. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 131–138 (2015). DOI 10.1109/INM.2015.7140285
19. Claeys, M., Latré, S., Famaey, J., Wu, T., Van Leekwijck, W., De Turck, F.: Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming. In: Adaptive and Learning Agents Workshop, part of AAMAS2013, Proceedings, pp. 30–37 (2013)
20. Claeys, M., Latre, S., Famaey, J., Turck, F.D.: Design and evaluation of a self-learning HTTP adaptive video streaming client. IEEE Communications Letters **18**(4), 716–719 (2014). DOI 10.1109/LCOMM.2014.020414.132649
21. Martín, V., Cabrera, J., Garca, N.: Evaluation of Q-learning approach for HTTP adaptive streaming. In: 2016 IEEE International Conference on Consumer Electronics (ICCE), pp. 293–294 (2016). DOI 10.1109/ICCE.2016.7430618
22. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
23. Bezerra, D., Ito, M., Melo, W., Sadok, D., Kelner, J.: DBuffer: A state machine oriented control system for DASH. In: 2016 IEEE Symposium on Computers and Communication (ISCC), pp. 861–867 (2016). DOI 10.1109/ISCC.2016.7543844
24. GPAC: Osmo4 / MP4Client, a powerful multimedia player. URL https://www.gpac-licensing.com/gpac/. (Accessed 3 February 2017)
25. The Linux Foundation: netem wiki. URL https://wiki.linuxfoundation.org/networking/netem. (Accessed 24 April 2017)
26. ITU-T: H.265: High efficiency video coding. ITU-T Rec. H.265 (2016)
27. GPAC: MP4Box General Documentation. URL https://gpac.wp.imt.fr/mp4box/mp4box-documentation/. (Accessed 3 February 2017)
28. Xiph Foundation: Xiph.org video test media [derf's collection]. URL https://media.xiph.org/video/derf/. (Accessed 21 January 2017)
29. Mathworks: Matlab 2016b (2016). URL https://uk.mathworks.com/products/matlab.html