MACHINE LEARNING BASED RATE ADAPTATION WITH ELASTIC FEATURE SELECTION FOR HTTP-BASED STREAMING

Yu-Lin Chien[†], *Kate Ching-Ju Lin*[‡] *and Ming-Syan Chen*[†]

[†]Department of Electrical Engineering, National Taiwan University
[‡]Research Center for Information Technology Innovation, Academia Sinica
r01921101@ntu.edu.tw, katelin@citi.sinica.edu.tw, mschen@cc.ee.ntu.edu.tw

ABSTRACT

Dynamic Adaptive Streaming over HTTP (DASH) has become an emerging application nowadays. Video rate adaptation is a key to determine the video quality of HTTP-based media streaming. Recent works have proposed several algorithms that allow a DASH client to adapt its video encoding rate to network dynamics. While network conditions are typically affected by many different factors, these algorithms however usually consider only a few representative information, e.g., predicted available bandwidth or fullness of its playback buffer. In addition, the error in bandwidth estimation could significantly degrade their performance. Therefore, this paper presents Machine Learning-based Adaptive Streaming over HTTP (MLASH), an elastic framework that exploits a wide range of useful network-related features to train a rate classification model. The distinct properties of MLASH are that its machine learning-based framework can be incorporated with any existing adaptation algorithm and utilize big data characteristics to improve prediction accuracy. We show via trace-based simulations that machine learning-based adaptation can achieve a better performance than traditional adaptation algorithms in terms of their target quality of experience (QoE) metrics.

Index Terms— HTTP Streaming, Rate Adaptation, Machine Learning

1. INTRODUCTION

Dynamic Adaptive Streaming over HTTP (DASH) is an adaptive bitrate streaming technique that enables media streaming over the Internet delivered from the HTTP web servers. In DASH, a media content object is partitioned into a sequence of small file segments. This partition allows a client to adaptively change the video encoding rate of each segment according to dynamic network conditions. Such an adaptive design can hence handle varying bandwidth conditions and provide smooth streaming during a streaming session.

While DASH provides the flexibility of video rate adaptation, how to select an appropriate rate for each video segment is still a challenging problem to ensure a good quality of experience. The problem is especially difficult because the video quality experienced by a client is determined by many conflicting performance metrics [1], such as the video rate, the rebuffer rate[1] or video rate smoothness. As a result, it is unlikely to find an universal adaptation algorithm that can optimize all the performance metrics. Thus, several recent works [2–5] have been proposed to either optimize certain metrics or make a trade-off between those diverse objectives.

Even regardless of the difficulty of optimizing different metrics, existing rate adaptation algorithms still experience some common deficiencies. First, most of the previous works rely on the information about the available bandwidth. The efficiency of these algorithms is closely determined by the accuracy of bandwidth estimation. However, it is inherently hard to forecast the future bandwidth based on the historical bandwidth measurements. Second, in general, network conditions can be evaluated by many different factors, such as current bandwidth, average bandwidth, latency, variation, playback buffer size, etc. However, existing algorithms usually select the video rate based on some conditional expressions. Since these expressions are derived according to high-level intuition, they do not take all those comprehensive factors into account, yet only use one or a few information as their input.

To cope with the above two issues, this paper introduces Machine Learning-based Adaptive Streaming over HTTP (MLASH), a flexible learning-based rate adaptation framework. Instead of proposing a new adaptation algorithm, our design principle is to exploit the machine learning technique to train a rate classification model for any existing rate adaptation algorithm. By elastically taking a wide range of useful information as features, we can build a classifier that helps improve prediction accuracy of the incorporated adaptation algorithm. In addition, the classification model is trained us-

The work is partially supported by the Ministry of Science and Technology of R.O.C. under contract No. 102-2221-E-001-012-MY2 and No. 103-2221-E-001-017-MY2 and by Academia Sinica Thematic Research Program under contract No. AS-104-TP-A05.

[1]The rebuffer rate is defined as the ratio of the number of rebuffer events to the total number of video segments.

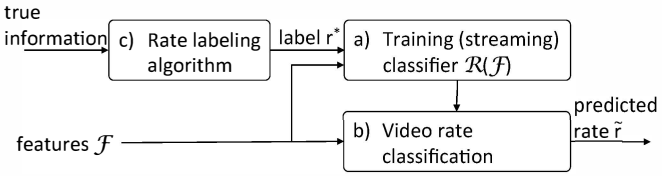

Fig. 1: System Architecture of MLASH

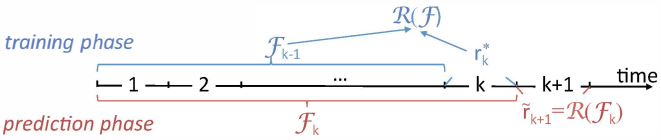

Fig. 2: Procedure of training the streaming classifier

ing a large number of transaction logs, and hence also benefits from big data characteristics. To the best of our knowledge, this work is the first to apply the machine learning approach to perform video rate adaptation. The trace-based simulations show that MLASH improves the performance of different incorporated algorithms in terms of their target quality of experience metrics.

The rest of paper is organized as follows. We give some related works in Section 2. Section 3 introduces the system architecture and design of MLASH, and Section 4 evaluates its performance. Finally, Section 5 concludes this work.

2. RELATED WORK

Previous work [1] has verified that the quality of experience of a streaming service is affected by many performance metrics, such as the video bit-rate, the rebuffer rate, and the frequency of video rate switching. Those metrics are usually conflicting with each other, and are hardly optimized at the same time. For example, in order to avoid rebuffering, which is referred to as the problem of draining out the buffer before new video segments arrive, a client should select a more conservative video rate, which is usually lower than the actual available bandwidth and thus underutilizes the bandwidth.

Therefore, several recent rate adaptation algorithms have been proposed to optimize one or a few specific performance metrics. The adaptation algorithm proposed in [2] tries to take the best trade-off between maximizing the video rate and minimizing the rebuffer rate. The works [3] [4] then focus on improving rate smoothness of video streaming by reducing the number of rate switching. Huang *et al.* argue in [6] [7] that the root of inefficiency of previous rate adaptation algorithms is the difficulty of bandwidth estimation, and hence propose to avoid rebuffer events and rate switching only based on the buffer size. A utility function that jointly considers diverse metrics is then proposed in [5], which aims at enhancing the overall quality of experience of a client. Our MLASH differs from the above ones in that it exploits the machine learning technique to utilize the comprehensive network-related features and improve prediction accuracy.

3. MLASH DESIGN

3.1. System Architecture

MLASH's design consists of two main components: *a) model training* and *b) video rate prediction*, as shown in Figure 1. We consider HTTP-based streaming, where a video is partitioned into several segments, each with an interval of T seconds. Unlike previous rate adaptation algorithms that utilize the *estimated* information, e.g., the average historical bandwidth or the bandwidth of the previous video segment, to predict the best video rate of the next video segment, our machine learning-based framework instead exploits the *true* information to train a rate classification model. In particular, say that a client has already received k video segments. We use 1) a set of historical network-related information measured from the first $(k-1)$ received segments as the feature set, denoted by $\mathcal{F}_{k-1}$, and 2) the *true* best video rate r_k^* of the k-th received segment as the corresponding label to train a classification model $\mathcal{R}(\mathcal{F})$, as shown in Figure 2. This model can then be used to predict the best video rate of the $(k+1)$-th segment using the historical information retrieved from the k received segments. Since the label of a training data, i.e., *true* best rate, is identified based on the *true* bandwidth observed in the k-th received segment, the trained classification model $\mathcal{R}()$ can find the explicit relationship between the measured features $\mathcal{F}$ and their corresponding best video rate r^*.

The accuracy of the classification model, however, relies on how many training data used to learn the classification model. Therefore, instead of asking every client to learn its own model, we let a service provider build the model based on a large amount of measurements reported from its served clients. More specifically, when a client requests for the $(k+1)$-th segment of a video, it also reports its measured features and the label to the service provider. Once the service provider collects a number of requests for different segments of various videos from all its clients, it extracts the reported feature sets $\mathcal{F}_{k-1}$ and the corresponding labels r_k^* to train a classification model $\mathcal{R}(\mathcal{F})$. This classifier $\mathcal{R}(\mathcal{F})$ can be used to predict the best rate $\tilde{r}$ of the current request and any future video requests. After model training, the server feeds the reported features $\mathcal{F}_k$ into the model, predicts the best rate of the requested segment, i.e., $\tilde{r}_{k+1} = \mathcal{R}(\mathcal{F}_k)$, and forwards the video segment of the selected rate $\tilde{r}_{k+1}$ to the client.

Two things are worth noting here. First, this classification model can be offline trained using a number of previous requests, or online updated using streaming data to improve prediction accuracy over time. The model hence benefits from big data characteristics. Second, as we mentioned in Section 2, since the quality of experience for video streaming depends on many different metrics, it is hard to have a universal algorithm that identifies the "optimal" video rate, even when all the true information is exactly known. Hence, we

keep our design flexible, and enable any service provider to use its preferable rate labeling algorithm, i.e., component *c)* in Figure 1, to determine the label (namely its expected best rate).

3.2. Feature Selection

Existing algorithms usually use a series of conditional expressions to predict the video rate. Since those expressions are found according to some high-level intuition, they can only take one or a few features as the input. However, in general, network performance can be evaluated from many different perspectives. The goal of MLASH is hence to leverage all these features to find a more comprehensive model. The features considered in our model are categorized as follows.

a) Bandwidth: Available bandwidth is the most common factor used for rate prediction. However, there exist many different ways to measure the bandwidth of a client. Most expression-based algorithms, e.g., [2, 5], however can only take one of these different measures as the input of the expressions. This is also why the recent work [6] adopts a pure buffer-based algorithm, without worrying about the difficulty of bandwidth estimation. Different from the above two types of solutions, we instead take all different forms of bandwidth estimates as the features, including

- *Last segment bandwidth (LSB),* which measures the available bandwidth of the last video segment.
- *Session average bandwidth (SAB),* which measures the available bandwidth of the whole video session.
- *Moving window average bandwidth (WAB),* which measures the average bandwidth of the last k segments.
- *Variation,* which represents the variation of the bandwidth measured in different segments.

b) Buffer size: The buffer size is another important factor that affects the rebuffer rate. We hence further consider the following features.

- *Current buffer length,* which is the length of video segments (in seconds) currently cached in the buffer.
- *Maximal buffer length,* which equals $L * r_{\max}$ seconds, where L is the buffer size (in bits) maintained by a client and $r_{\max}$ is the maximal video rate.

c) Round-trip time (RTT), which is the total time for a request to be sent to and responded from the server.

d) Current video rate, which is the video rate currently watched by the client. This feature might affect the decision of the model as we consider video rate smoothness.

Limited by the data traces we have, we only use the above features. However, more features that could potentially affect video rate selection, such as the packet loss rate and the latency, can be incorporated in our system.

3.3. Model Training

It is fairly difficult to find expressions that represent the complex relationship between the wide range of features and the corresponding best rate. Therefore, we build our rate adaptation model using a *decision-tree-based random forest classification* [8] approach, which combines multiple decision tree classifiers to create an improved composite classification model that maps a given set of features to an output label. Our random forest model was implemented using the Scikit-learn package [9]. We set the number of trees in the forest as 200 and maximum depth of the trees as 50. Note that a content provider could store their video objects in different content delivery networks (CDNs), which might experience different network conditions. Hence, to improve model accuracy, we train a distinct model for those video servers with the same IP prefix.

Though our model can be built based on the existing decision tree learning algorithms, there however still exists a challenging problem unsolved. That is, given a set of features $\mathcal{F}$, how can we find its corresponding label for model training? Ideally, the label of a training data should be the expected best video rate, which could be found based on the true information, e.g., the actual throughput of video downloading or the real events of rebuffering or rate switching. However, it is hard to distinguish which rate is the best one, even given the true information. For example, the video rate that is closest to the true available bandwidth might not cause rebuffering, but might be different from the currently-used one, resulting in rate switching.

Fortunately, many previous works have proposed different rate adaptation algorithms, each of which targets to optimize one or a few performance metrics. These algorithms might not be efficient due to bandwidth estimation errors, but should fundamentally work if the real bandwidth is perfectly known. Therefore, instead of using those algorithms to predict the rate, we alternatively incorporate them with our machine learning-based framework and utilize them to "*label the best rate*" of a feature set $\mathcal{F}$. In other words, our MLASH leaves the labeling procedure flexible, and enables the service provider to choose a labeling algorithm that optimizes its favorable performance metrics. We will show in Section 4 that combining those algorithms with MLASH can improve their performance because the trained classifier can filter out the effect of estimation errors. We give some example algorithms as follows, and use them as the labeling algorithms in our evaluation.

a) Bandwidth-based rate adaptation: This naïve algorithm is used to maximize the video rate. It always picks a video rate that is closest to but not above the actual available bandwidth, which can be expressed by

$$r^* = \arg\max_{r \in V_r} r \leq bw, \tag{1}$$

where V_r is the set of all available video rates and bw is the

true available bandwidth.

b) Buffer-considered rate adaptation [2]: This algorithm is designed to avoid rebuffering by taking both the bandwidth and current buffer length into account. It picks an aggressive rate when the buffer is nearly full, while picking a conservative rate when the buffer is nearly empty. The selected rate is given by

$$r^* = \arg\max_{r \in V_r} r \leq bw', \text{where}$$

$$bw' = \begin{cases} bw * 0.3, & \text{if } 0.00 \leq bl < 0.15 \\ bw * 0.5, & \text{if } 0.15 \leq bl < 0.35 \\ bw, & \text{if } 0.35 \leq bl < 0.50 \\ bw * (1 + 0.5 * bl), & \text{otherwise,} \end{cases} \quad (2)$$

and bw is the true bandwidth and bl is the ratio of the current buffer length to the maximal buffer length.

c) Smooth rate adaptation [3]: This algorithm tries to minimize the number of video rate switching and ensure video playback smoothness at the same time. It chooses the rate with consideration of the currently-used video rate, and avoids rate oscillation when the bandwidth fluctuates. To ensure playback smoothness, when the bandwidth is decreasing, it immediately decreases the video rate to fit the current bandwidth. The algorithm introduces a knob m to tune the smoothness. Thus, we also use this knob m as the feature in our model training. We refer the readers to [3] for the detailed algorithm.

4. TRACE-BASED EVALUATION

We use the trace of HTTP streaming provided by [10] to check the performance of MLASH. The trace records the results of 10,000 tests everyday. The requests are from more than 1,000 hosts (IP addresses), which locate in about 100 different countries and 1,000 autonomous systems. Each test lasts for 30 seconds, in which the client periodically requests for a video segment with a constant video bitrate of two seconds. The clients in the trace simply apply the bandwidth-based rate adaptation based on the measure of last segment bandwidth (LSB). The trace logs the information, including the video rate of each segment, time required for downloading a segment, and monitored RTT. Since the trace does not give any information about the buffer, unless otherwise specified, we set the default maximal buffer length to 10 seconds in our simulations.

The true available bandwidth of each segment is calculated by the number of bits of a video segment divided by the required downloading time. Since we apply some other rate adaptation schemes on top of this trace, the selected rate might be different from the one logged in the trace. We can use this true bandwidth to estimate the required downloading time of the newly selected video rate by $2r/bw$, where r is the video rate of a 2-second segment and bw is the true bandwidth. In addition, this true bandwidth is also used to find the label of the training data. We use all the tests logged in December 2013 as our training data, and use data in January 2014 as our testing data. The available video encoding rates used in our simulations include 100, 150, 200, 250, 300, 400, 500, 700, 900, 1200, 1500, 2000, 2500, 3000, 4000, 5000, 6000, 7000, 10000, 20000 kb/s.

We compare our MLASH design with the following schemes:

- *Bandwidth-based adaptation*, which selects the video rate based on Eq. 1.
- *Buffer-considered adaptation* [2], which selects the video rate based on Eq. 2.
- *Rate smoothness-based adaptation* [3], which tries to improve video rate smoothness.
- *Buffer-based adaptation (d sec)* [6], which selects the rate purely based on the current buffer length when the maximal buffer length is set to d seconds.

Since the first three schemes all require the information about estimated bandwidth, we use three different measures, i.e., SAB, LSB and WAB, as the estimate of the bandwidth, and further use the true bandwidth to find the upper bound of those algorithms, i.e., without any estimation error.

4.1. Performance Comparison

We check the performance of MLASH by using the first three algorithms to find the label of training data. We do not use the buffer-based algorithm as the labeling algorithm because the performance of this algorithm is not affected by bandwidth estimation error. Thus, combining the buffer-based algorithm with our machine learning-based framework does not improve its performance.

MLASH with bandwidth-based rate labeling: We first check whether our machine learning-based approach can improve the performance of bandwidth-based adaptation. Figure 3 plots the selected video rate of different comparison schemes. The results show that using estimated average bandwidth to select the video rate cannot adapt to network dynamics and is very likely to pick a more conservative rate. The pure buffer-based algorithm does not consider the bandwidth, and could select a rate either much lower or higher than the available bandwidth. A more important issue is that the performance of the buffer-based algorithm highly depends on the maximal buffer length, and it is quite difficult to determine the optimal buffer length for a video of any length. In contrast, MLASH can predict a video rate that is fairly close to the optimal rate chosen based on the true bandwidth. This means that our machine learning-based adaptation can effectively avoid performance degradation due to bandwidth estimation errors, and hence improve network utilization.

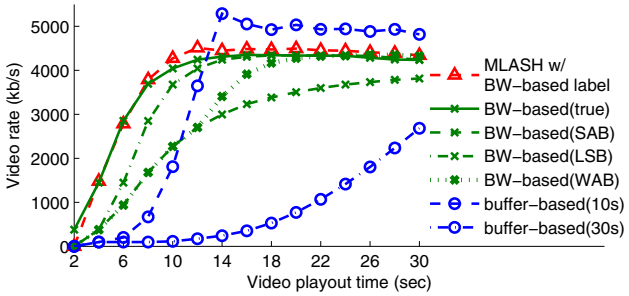

Fig. 3: Video rate over time (bandwidth-based labeling)

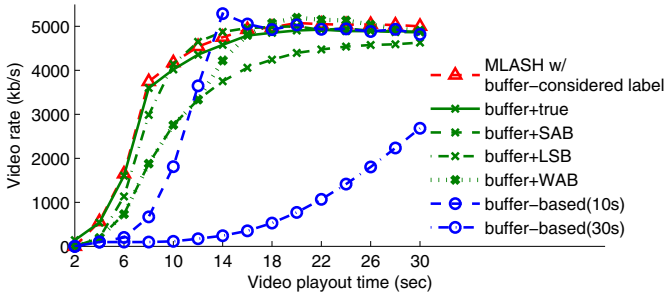

Fig. 4: Video rate over time (buffer-considered labeling)

Algorithm	BW-based (SAB)	BW-based (LSB)	BW-based (WAB)	Buffer-based(10)	Learning-based	BW-based (true)
Average Rate(kb/s)	2760	3554	3157	3560	3945	3845
Average Error(kb/s)	1417	731	1160	1600	560	0
Rebuffer Rate(%)	8.9	9.1	9.2	8.6	12.8	6.1
OverEst. Rate(%)	4.2	4.2	4.5	3.7	8.4	0

Table 1: Other performance (bandwidth-based labeling)

Algorithm	Buffer +SAB	Buffer +LSB	Buffer +WAB	Buffer-based(10)	Learning-based	Buffer +true
Average Rate(kb/s)	3012	3827	3427	3560	4138	4014
Average Error(kb/s)	1438	688	1155	1332	532	0
Rebuffer Rate(%)	4.9	4.5	5.1	4.3	4.5	3.8
OverEst. Rate(%)	1.24	0.78	1.38	0.57	1.17	0

Table 2: Other performance (buffer-considered labeling)

We also show in Table 1 the average video rate and average prediction error for each algorithm. The average video rate is the average video bitrate for the segments of all the 30-second-long videos, and the prediction error is defined as $|\tilde{r} - r_{\texttt{true}}|$, where $\tilde{r}$ is the predicted video rate and $r_{\texttt{true}}$ is the rate selected based on the true bandwidth. The results verify that MLASH can improve the accuracy of bandwidth estimation, and hence select a rate close to the actual bandwidth. Table 1 also summarizes the rebuffer rate of the comparison schemes, which is defined as the number of segments experiencing a rebuffer event divided by the total number of video segments. We note that some rebuffer events are not caused by bandwidth overestimation, yet are inevitable when the available bandwidth is lower than the lowest video rate. We hence also show the rate of rebuffer events that are mainly caused by bandwidth overestimation, i.e., $\tilde{r} > r_{\texttt{true}}$. We can see from the statistics that, since, as incorporated with bandwidth-based labeling, our classifier aims at maximizing network utilization without considering the rebuffer issue, the expected price it has to pay is a slightly higher rebuffer rate.

MLASH with buffer-considered rate labeling: We next check how MLASH performs as it is incorporated with the buffer-considered adaptation algorithm. Figure 4 compares the video rate of the traditional buffer-considered algorithm to our design with buffer-considered labeling. The results follow a trend similar to that shown in Figure 3. Our approach again can select a video rate close to the optimal rate chosen based on the true bandwidth. Table 2 summarizes the prediction errors and the rebuffer rate. Interestingly, we find that, as incorporated with buffer-considered labeling, our machine learning-based adaptation has a pretty low rebuffer rate, while achieving a much higher video rate than other schemes. The rebuffer rate of our design is close to that of the buffer-based algorithm, which is especially designed to eliminate rebuffering. This implies that, with a proper labeling scheme, MLASH can efficiently utilize the available bandwidth, while avoiding rebuffering at the same time.

MLASH with smoothness-based rate labeling: We next show the performance of MLASH with smoothness-based labeling in Figure 5. Since the smoothness-based adaptation algorithm is designed to prevent unexpected rate switching, we hence also show in Table 3 the switching rate, which is defined as the ratio of the number of switching events to the total number of video segments. The statistics show that MLASH can achieve a similar switching rate, as compared to the traditional smoothness-based algorithm using the true bandwidth information. The traditional smoothness-based algorithm has a much smaller switching rate when it uses the estimated bandwidth. This is because the estimated bandwidth is the average bandwidth of a duration, and is naturally smoother than the true bandwidth. We can however observe that the video rate selected based on estimated bandwidth is much lower than the ideal video rate. On the other hand, the performance of the pure buffer-based algorithm is fairly sensitive to the buffer length. Therefore, an improper buffer length could lead to frequent rate switching.

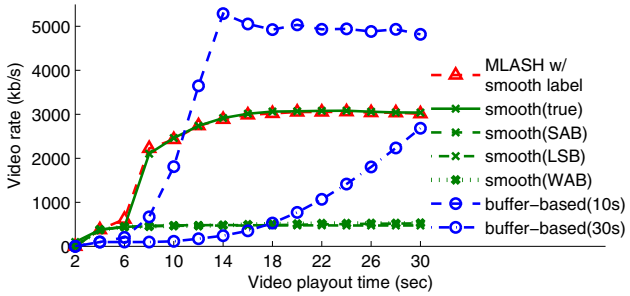

Fig. 5: Video rate over time (smoothness-based labeling)

Algorithm	Smooth (SAB)	Smooth (LSB)	Smooth (WAB)	Buffer-based(10)	Learning-based	Smooth (true)
Average Rate(kb/s)	470	459	479	3560	2472	2468
Average Error(kb/s)	2167	2160	2161	2028	160	0
Rebuffer Rate(%)	5.4	5.5	5.4	4.3	5.4	4.2
OverEst. Rate(%)	1.39	1.45	1.39	0.57	1.42	0
Switching Rate(%)	13.2	13.1	13.2	57.9	26.4	23.1

Table 3: Other performance (Smoothness-based labeling)

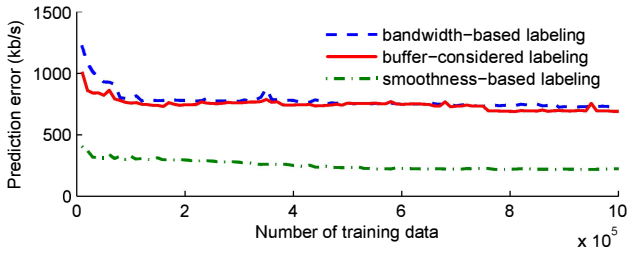

Fig. 6: Prediction error of the streaming model over time

4.2. Convergence of Model Training

Our MLASH can use either the offline classification model or streaming classification model. The streaming model can not only use accumulated transaction logs to improve prediction accuracy, but also adapt to network dynamics. We hence check how many training data are required to obtain a stable and reliable classification model. Figure 6 plots the prediction error of the streaming classification model for different labeling algorithms as the number of training data grows over time. The figure shows that the accuracy becomes quite stable when the classifier is trained by using about 100,000 records of training data.

5. CONCLUSION

This paper presents a machine learning-based adaptive streaming framework over HTTP (MLASH). Instead of designing a completely new adaptation algorithm, our goal is to combine the machine learning technique with different existing rate adaptation algorithms designed for optimizing different QoE metrics. We train a classification model to describe the explicit relationships between a wide range of network-related features and the label found by any preferable rate adaptation algorithm based on *true* information. This machine learning-based approach can hence elastically utilize comprehensive features, and, more importantly, avoids the difficulty of bandwidth estimation faced by many existing adaptation algorithms. We demonstrate via trace-based simulations that, by leveraging existing adaptation algorithms as the labeling scheme, our MLASH can improve prediction accuracy of those algorithms, and hence their target performance metrics.

6. REFERENCES

[1] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," in *ACM SIGCOMM*, 2013.

[2] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments," in *Workshop on Mobile Video*, 2012.

[3] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic HTTP streaming," in *International Conference on Emerging Networking Experiments and Technologies*, 2012.

[4] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, "QDASH: A QoE-aware DASH system," in *ACM Multimedia Systems Conference (MMSys)*, 2012.

[5] V. Joseph and G. de Veciana, "NOVA: QoE-driven optimization of DASH-based video delivery in networks," in *IEEE INFOCOM*, 2014.

[6] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *ACM SIGCOMM*, 2014.

[7] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: Picking a video streaming rate is hard," in *ACM Conference on Internet Measurement Conference (IMC)*, 2012.

[8] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[10] S. Basso, A. Servetti, E. Masala, and J. C. De Martin, "Measuring DASH streaming performance from the end users perspective using neubot," in *ACM Multimedia Systems Conference (MMSys)*, 2014.