國立臺灣大學電機資訊學院電機工程學研究所

碩士論文

Graduate Institute of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

基於機器學習方法之 HTTP 串流速率調節機制

Machine Learning Based Rate Adaptation with Elastic Feature Selection for HTTP-Based Streaming

簡友琳

Yu-Lin Chien

指導教授：陳銘憲博士，林靖茹博士

Advisor: Ming-Syan Chen, Ph.D.

Ching-Ju Lin, Ph.D.

中華民國 104 年 6 月

June, 2015

中文摘要

Dynamic Adaptive Streaming over HTTP (DASH) 於現在已經成為一個越來越重要的應用。影響 HTTP 上串流影音品質最重要的關鍵，就在於如何選擇適當的影片速率調節機制。之前的一些相關論文提出一些可以根據目前網路狀態的變化，來動態調整下載影片速率的方法；但是會影響到影片速率選擇的因素有許多種，而這些方法一般都只考慮其中少數的幾個重要因素，像是預測的頻寬或是目前緩衝影片的長度。但是頻寬預測不僅相當困難，同時容易有很大誤差可能，而這導致了其可能嚴重影響到速率選擇的效果。為了解決這個問題，我們提出了於 HTTP 上基於機器學習的速率調節機制（MLASH）。利用 classification 的方法，MLASH 不僅可以有彈性的將所有可能影響到速率調節的因素都考慮進來，還可以避開頻寬預測的困難。同時，MLASH 還可以與之前的其他速率調節方法進行整合，並且利用大數據的特性，來進一步提升速率調節之效果。我們根據原始資料來進行模擬實驗，以證明我們的方法不僅效果良好，同時於不同的使用者體驗衡量標準上，表現也比之前其他的速率調節方法更加優秀。

關鍵字：HTTP 串流，速率調節，機器學習

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.

Key words: HTTP Streaming, Rate Adaptation, Machine Learning

Contents

List of Figures

Chapter 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 encoding 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 a 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

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

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 using 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 chapter 2. chapter 3 introduces the system architecture and design of MLASH, and chapter 4 evaluates its performance. Finally, chapter 5 concludes this work.

Chapter 2

Related Work and Background

2.1 QoE Metrics and Rate adaptation algorithm

There are lots of previous works about rate adaptation algorithm. Their approaches focus on selecting the most suitable quality level for users during video playback. Yet, traditional metrics such as Peak Signal-to-Noise Ratio (PSNR) won't be important anymore in DASH scenario [1]. Metrics that capture delivery-related effects such as video bitrate, rebuffer rate, video switching rate and join time are much more important. However, 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.

Due to the reason above, we can't design an optimal rate adaptation algorithm which optimizes all the metrics at the same time. Previous rate adaptation algorithms usually focus on a few metrics according to their design principle. [2] [6] try to design rate adaptation algorithms that guarantee higher average bitrate and low rebuffer rate at the same time. [3] [4] propose rate adaptation approaches which focus on minimizing video switching rate. Huang *et al.* argue in [7] [8] [9] 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.

There are other works designing their rate adaptation algorithms according to their requirement or system architecture [10] [11] [12] [13] [14]. Besides these works, there are some works centralized manage all clients and perform rate adaptation on the server side [15] [16], and some works [6, 17–19] investigating video rate adaptation in cellular systems. The works [17] [6] allocate bandwidth resources to different streaming clients in order to optimize their video rates.

Another work [18] further takes wireless link quality into account in its joint rate scheduling and resource allocation algorithm. Cicalo *et al.* [19] then propose to deliver fair video quality to multiple clients.

2.2 TCP throughput / bandwidth prediction

Lots of previous works try to predict TCP throughput. There are two kinds of TCP throughput prediction methods: (i) Formula-based (FB) approach, (ii) History-based (HB) approach. Formula-based approaches try to construct mathematical models. They express TCP throughput as a function of the characteristics of the network path [20] [21] [22]. However, FB prediction does not perform well in some cases because it constructs the model according to priori characteristics before the flow starts. If there are significant changes of the characteristics during flow transfer, the prediction error could be unacceptably large. In addition, the network delays and losses are hard to be predicted accurately, which also increase the prediction error.

In contrast, History-based approaches use history of previous TCP transaction logs to predict TCP throughput on the same path [23] [24] [25]. It is proved to be more accurate than FB approaches in most cases, even when there is only a few previous transaction logs given [26]. However, HB approaches are highly path-dependent. That is, the load on the path and the degree of multiplexing will influence the accuracy of the prediction very much.

Except the FB and HB approaches, Mirza et al. [27] proposes a machine learning approach to predict bandwidth. They apply support vector regression (SVR) method, using available bandwidth, queuing delays and packet loss as features, to perform prediction. However, they prove that simple HB approaches can achieve better performance than complicate SVR approach in their case.

2.3 Resource allocation

There are some works discussing centralized resource allocation for streaming video. [28] tries to optimize prefetching policies by using parallel TCP connections to achieve effective buffer management of prefetched data, while [29] proposes a download scheduling algorithm based on crowd-sourced video viewing statistics to perform the right amount of prefetching. Besides, [30] proposes a Quality-of-Service (QoS)-aware video streaming architecture in order to improve user experience while reducing the waste of network resources.

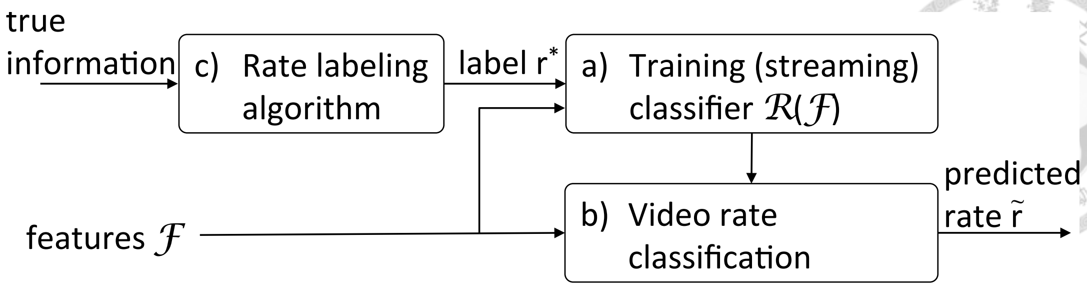
Chapter 3

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 3.1. Unlike previous algorithms that utilize the *estimated* information, e.g., estimated bandwidth, to find the video rate, our machine-learning-based framework instead exploits the *true* information to train a rate classification model. In particular, we partition a video into several segments, each with an interval of T seconds. We use 1) a set of historical network-related information measured from the first $(k-1)$ segments as the feature set, denoted by $\mathcal{F}_{k-1}$, and 2) the *true* best video rate r^*_k of the k-th segment as the corresponding label to train a classification model $\mathcal{R}(\mathcal{F})$, as shown in Figure 3.2. The label of a training data, i.e., *true* best rate, is identified based on the *true* information observed in the k-th segment, e.g., true bandwidth. As a result, the trained classification model $\mathcal{R}()$ can find the explicit relationship between the measured features $\mathcal{F}$ and their corresponding best rate r^*.

More specifically, once a service provider collects a number of requests for different segments of various videos from the clients, it can use their feature sets and the corresponding labels to train a classification model $\mathcal{R}(\mathcal{F})$. We then use this classifier $\mathcal{R}(\mathcal{F})$ to predict the best rate $\tilde{r}$ of any new video request based on its feature set $\mathcal{F}$. 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. Second, as we mentioned in



true information
c) Rate labeling algorithm
label r*
a) Training (streaming) classifier $\mathcal{R}(\mathcal{F})$
b) Video rate classification
features $\mathcal{F}$
predicted rate $\tilde{r}$

Figure 3.1: System Architecture of MLASH

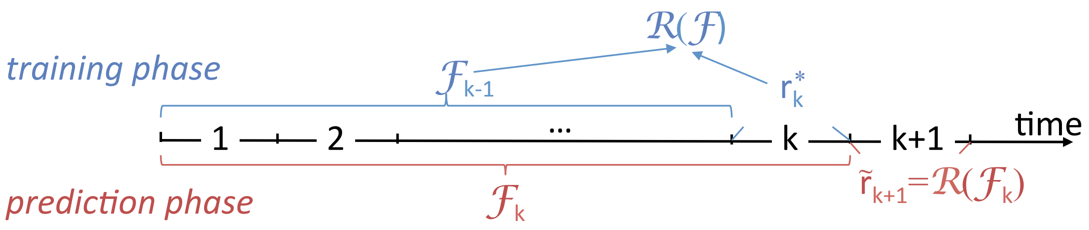


Figure 3.2: Procedure of training the streaming classifier

chapter 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 determine the label (namely its expected best rate) based on its preferable rate labeling algorithm, i.e., component *c)* in Figure 3.1.

When a client requests for the $(k+1)$-th segment of a video, it also sends the measured historical information in order for the server to extract the feature set $\mathcal{F}_k$. The server then feeds $\mathcal{F}_k$ into its classifier, predicts the best rate $\tilde{r}_{k+1} = \mathcal{R}(\mathcal{F}_k)$, and forwards the video segment of the selected rate $\tilde{r}_{k+1}$ to that client. If the server enables streaming classification, it further asks the client to report the label, i.e., *true* best rate, of the k-th video segment, r_k^*. Thus, the server can use this label r_k^* and the feature set $\mathcal{F}_{k-1}$ reported in the last request to update the model, and improve prediction accuracy over time.

In this work, we mainly focus on rate adaptation for constant bitrate video, and leave addressing the issue of variable bitrate video as our future work. We will describe in the next two sections how we train our rate classification model.

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 [7] 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.

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* [31] approach, which maps a given set of features to an output label. 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., true bandwidth or the event of rebuffering and rate switching. However, it is hard to distinguish which rate is the best one, even given those 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 true information, e.g., 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 chapter 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

$$r^* = \arg\max_{r \in V_r} r \leq bw, \tag{3.1}$$

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

$$r^* = \arg\max_{r \in V_r} r \leq bw', \text{where}$$
$$bw' = \begin{cases} bw * 0.3, & \text{if } 0.00 \leq bl < 0.15 \\ bw * 0.5, & \text{if } 0.15 \leq bl < 0.35 \\ bw, & \text{if } 0.35 \leq bl < 0.50 \\ bw * (1 + 0.5 * bl), & \text{otherwise,} \end{cases} \tag{3.2}$$

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.

Chapter 4

Trace-based Evaluation

We use the trace of HTTP streaming provided by [32] 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 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:

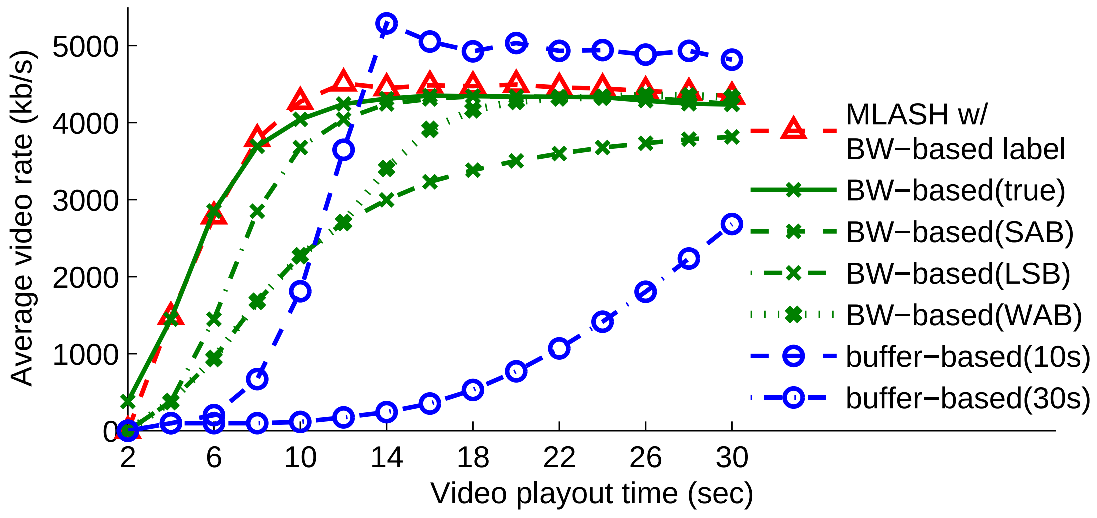
- *Bandwidth-based adaptation*, which selects the video rate based on Equation 3.1.
- *Buffer-considered adaptation* [2], which selects the video rate based on Equation 3.2.
- *Rate smoothness-based adaptation* [3], which tries to improve video rate smoothness.
- *Buffer-based adaptation (d sec)* [7], 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 4.1(a) 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



(a) Video rate over time

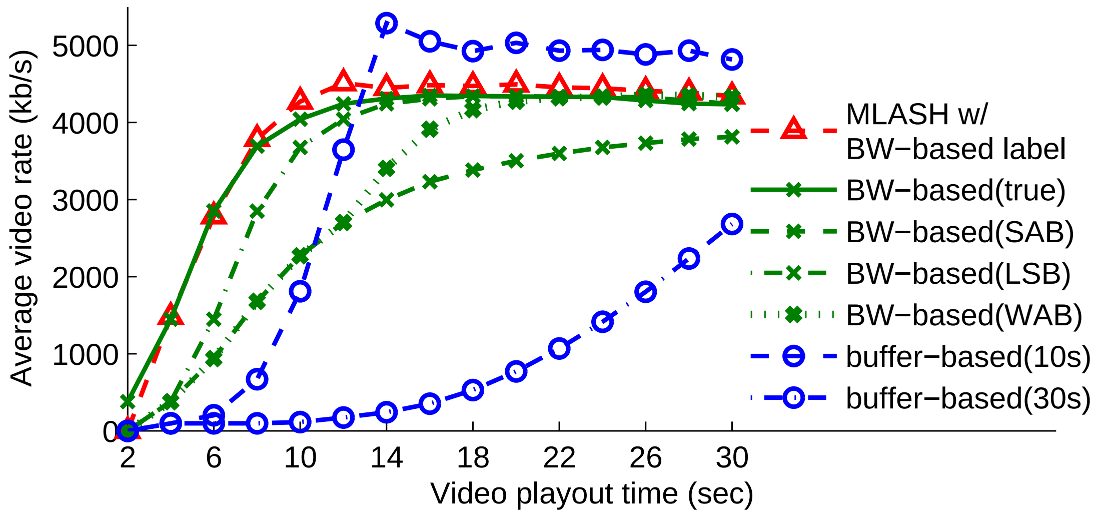
Algorithm	**BW-based (SAB)**	**BW-based (LSB)**	**BW-based (WAB)**	**Buffer-based(10)**	**Learning-based**	**BW-based (true)**
Average Rate(kb/s)	2760	3554	3157	3560	3945	3845
Average Error(kb/s)	1417	731	1160	1600	560	0
Rebuffer Rate(%)	8.9	9.1	9.2	8.6	12.8	6.1
OverEst. Rate(%)	4.2	4.2	4.5	3.7	8.4	0

(b) Prediction errors and other performance

Figure 4.1: Performance of MLASH with bandwidth-based labeling

bandwidth. This means that our machine-learning-based adaptation can effectively avoid performance degradation due to bandwidth estimation errors, and hence improve network utilization.

We also show in Figure 4.1(b) the prediction error, which 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. Figure 4.1(b) also summarizes the rebuffer rate of the comparison schemes, which is defined as the number of segments ex-



(a) Video rate over time

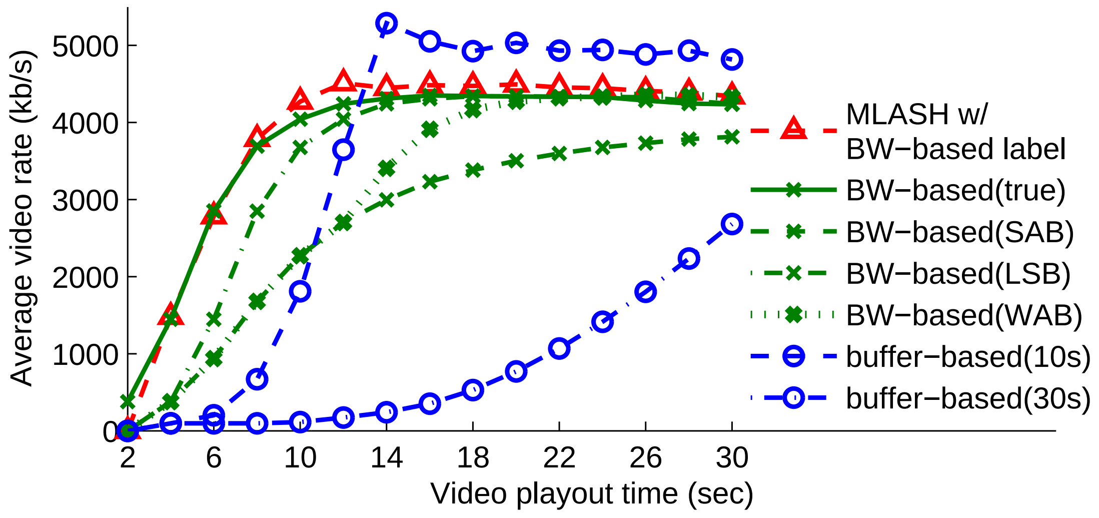
Algorithm	**Buffer +SAB**	**Buffer +LSB**	**Buffer +WAB**	**Buffer-based(10)**	**Learning-based**	**Buffer +true**
Average Rate(kb/s)	3012	3827	3427	3560	4138	4014
Average Error(kb/s)	1438	688	1155	1332	532	0
Rebuffer Rate(%)	4.9	4.5	5.1	4.3	4.5	3.8
OverEst. Rate(%)	1.24	0.78	1.38	0.57	1.17	0

(b) Prediction errors and other performance

Figure 4.2: Performance of MLASH with buffer-considered labeling

periencing 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



(a) Video rate over time

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

(b) Prediction errors and other performance

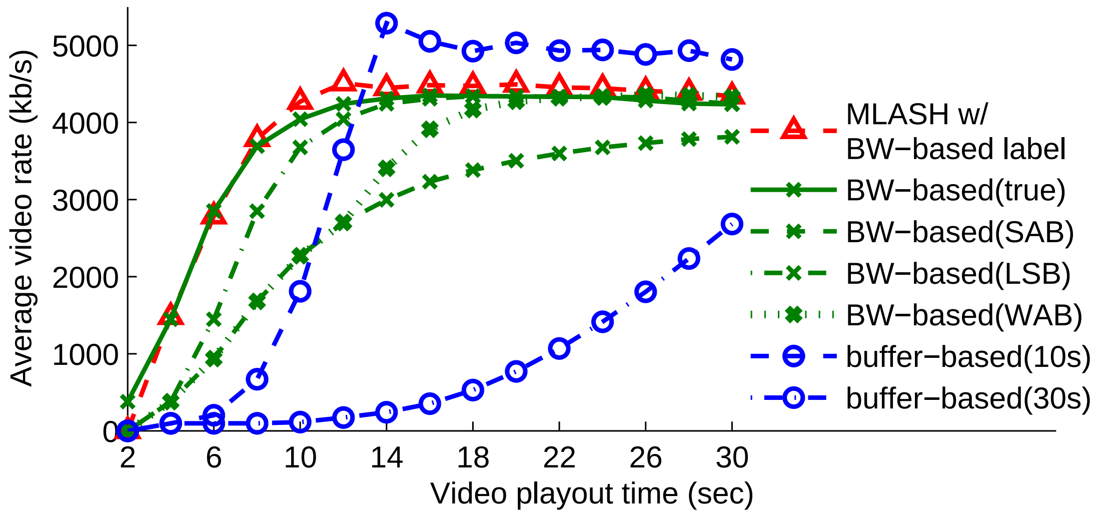
Figure 4.3: Performance of MLASH with smoothness-based labeling

it is incorporated with the buffer-considered adaptation algorithm. Figure 4.2(a) 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 4.1(a). Our approach again can select a video rate close to the optimal rate chosen based on the true bandwidth. Figure 4.2(b) 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 4.3. Since the smoothness-based adaptation algorithm is designed to prevent unexpected rate switching, we hence also show in Figure 4.3(b) 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.

4.2 Variable bitrate scenario

Varaible bitrate (VBR) encoding technique is commonly used on sound or video encoding. Videos encoded in VBR vary the amount of output data per time segment, thus the rate adaptation become more difficult due to the fluctuation of the video segment size. We try to explore the performance of MLASH under VBR scenario. Due to the fact that our dataset only provides constant bitrate encoding, we add fluctuations for the size of each video segment randomly to simulate the varaible bitrate scenario. We use normal distribution to add a multiplier for each segment size of the video, the mean of the normal distribution is set as the current video encoding rate and the variance is set as 0.3 of the current video encoding rate, which is close to video segment size distribution with VBR



(a) Video rate over time

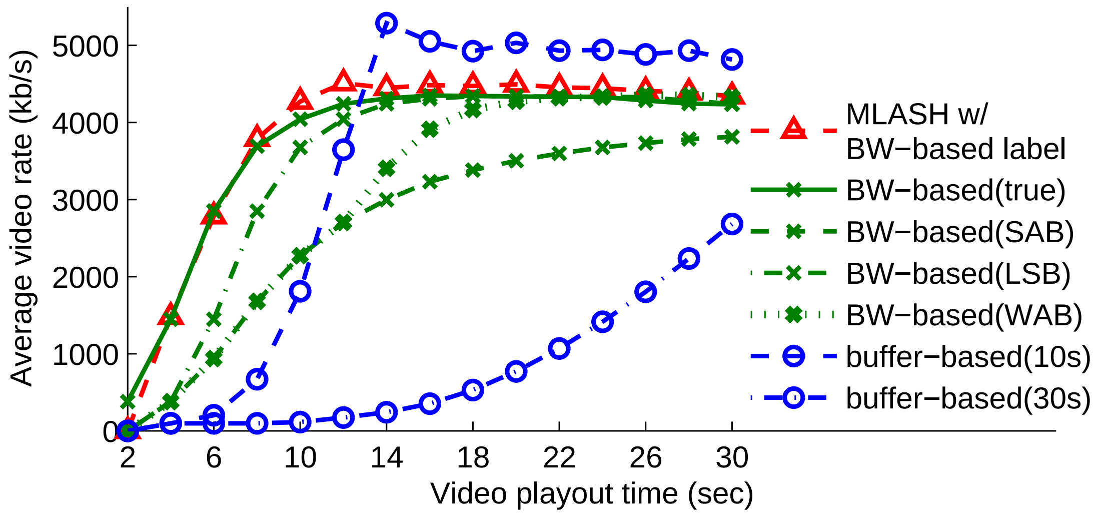
Algorithm	**BW-based (SAB)**	**BW-based (LSB)**	**BW-based (WAB)**	**Buffer-based(10)**	**Learning-based**	**BW-based (true)**
Average Rate(kb/s)	2714	3498	3109	3545	3868	3785
Average Error(kb/s)	1384	716	1138	2022	607	0
Rebuffer Rate(%)	9.1	9.1	9.4	10.8	18.0	6.3
OverEst. Rate(%)	4.4	4.2	4.6	6.0	13.9	0

(b) Prediction errors and other performance

Figure 4.4: Performance of MLASH with bandwidth-based labeling under VBR scenario encoding.

Due to the fact that adding a multiplier will influence the selection of optimal video encoding rate, we reselect the optimal video rate according to the new video segment size on each timeslot. In addition, because the muitiplier ratio will influence the selected rate, we take it as an additional feature for each of the rate adaptation approach to incorporate the influence into our model.

We then observe the results comparing with traditional history-based appoaches under the simulated variable bitrate scenario.



(a) Video rate over time

Algorithm	Buffer +SAB	Buffer +LSB	Buffer +WAB	Buffer-based(10)	Learning-based	Buffer +true
Average Rate(kb/s)	3290	3926	3654	3545	4070	3954
Average Error(kb/s)	1076	739	1005	1769	633	0
Rebuffer Rate(%)	5.0	4.6	5.1	4.9	6.8	3.9
OverEst. Rate(%)	1.26	0.78	1.41	1.18	3.44	0

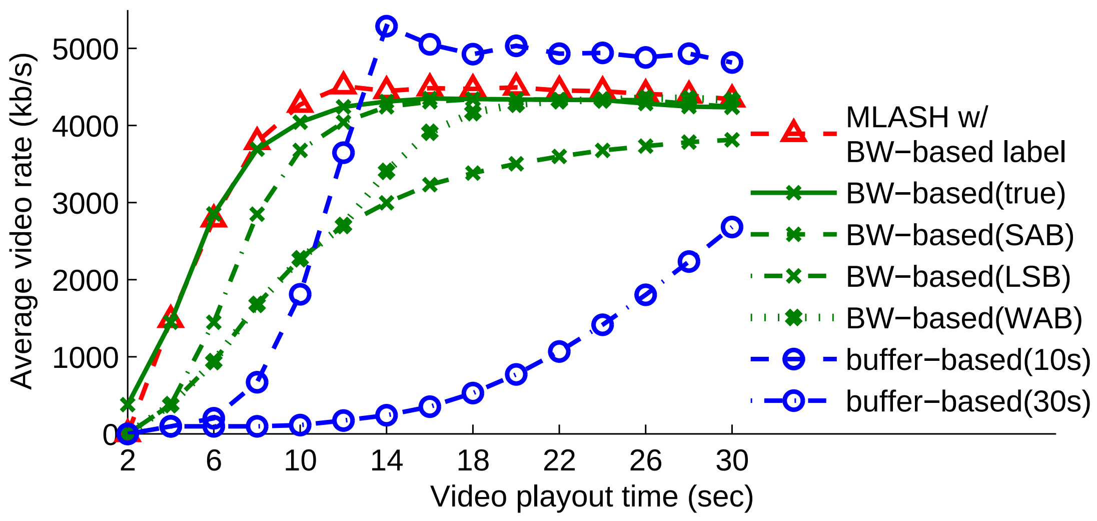
(b) Prediction errors and other performance

Figure 4.5: Performance of MLASH with buffer-considered labeling under VBR scenario

MLASH with bandwidth-based rate labeling under VBR scenario:

We first check whether our machine-learning-based approach can improve the performance of bandwidth-based adaptation under variable bitrate scenario. Figure 4.4(a) plots the selected video rate of different comparison schemes.

We also show in Figure 4.4(b) the prediction error. The results show that under VBR scenario, MLASH can still improve the accuracy of bandwidth estimation, and select a rate close to the actual bandwidth. Figure 4.4(b) also summarizes the rebuffer rate and rate of rebuffer events that are mainly caused by bandwidth overestimation of the comparison



(a) Video rate over time

Algorithm	**Smooth (SAB)**	**Smooth (LSB)**	**Smooth (WAB)**	**Buffer-based(10)**	**Learning-based**	**Smooth (true)**
Average Rate(kb/s)	537	2653	599	3545	2493	2528
Average Error(kb/s)	2166	2122	2139	2192	229	0
Rebuffer Rate(%)	5.8	7.3	5.8	6.0	10.1	4.4
OverEst. Rate(%)	1.65	1.75	1.69	1.80	4.62	0
Switching Rate(%)	14.7	18.8	14.8	65.7	29.4	26.5

(b) Prediction errors and other performance

Figure 4.6: Performance of MLASH with smoothness-based labeling under VBR scenario

schemes. We can see from the statistics that, our classifier still has high network utilization while having higher rebuffer rate.

MLASH with buffer-considered rate labeling under VBR scenario: We next check how MLASH performs as it is incorporated with the buffer-considered adaptation algorithm under variable bitrate scenario. Figure 4.5(a) compares the video rate of the traditional buffer-considered algorithm to our design with buffer-considered labeling. We find that comparing to Figure 4.4(a) and Figure 4.2, although MLASH can select a video rate close to the optimal rate chosen based on the true bandwidth, the performance of buffer-considered algorithm with last segment bandwidth is close to MLASH in average rate, and is better than our approach in rebuffer rate. It might due to the reason that the variable bitrate makes the rate selection more difficult, and because MLASH takes a more aggressive strategy compared to history-based bandwidth estimation approaches, it's easier to cause rebuffer event while selecting a relatively high video rate on average.

MLASH with smoothness-based rate labeling under VBR scenario: We next show the performance of MLASH with smoothness-based labeling under varaible bitrate scenario in Figure 4.6. The statistics show that just like Figure 4.5, MLASH can achieve a similar switching rate, as compared to the traditional smoothness-based algorithm using the true bandwidth information, but it still get a very high rebuffer rate compared to other approaches. It's also worth noting that, while traditional smoothness-based algorithms, with session average or moving window average as estimated bandwidth, still have a very low average selected video rate, the average rate of using last segment bandwidth as estimated bandwidth is very close to optimal rate selection. It means that, with a lower rebuffer rate and switching rate compared to MLASH , simple smoothness-based algorithm with last segment bandwidth as input will be a better choice than our MLASH under this scenario.

From the results above, we can get that the performance of MLASH under variable scenario isn't as good as the performance under constant bitrate scenario. It's more difficult for MLASH predictor to choose an aggressive rate, while keeping a low rebuffer rate at the same time. We would try to improve the performance under VBR scenario with a more specified method in our future work.

4.3 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 4.7 plots

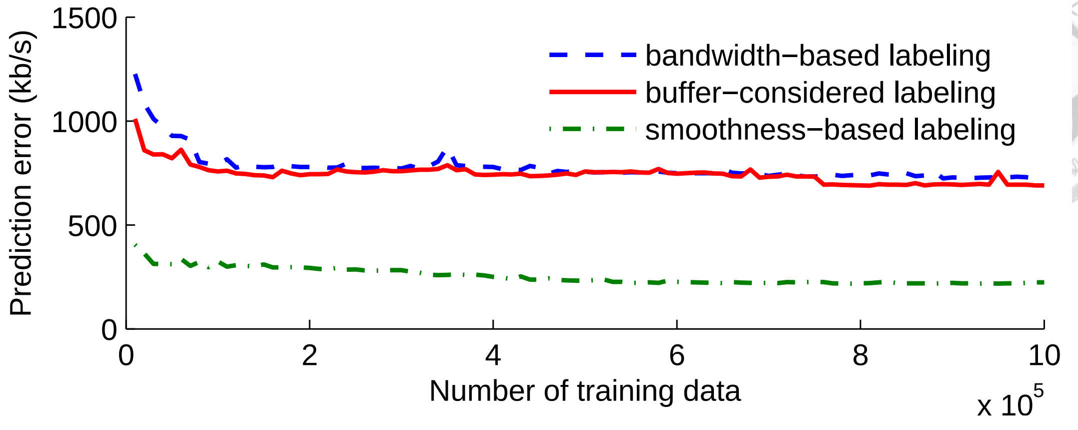


Figure 4.7: Prediction error of the streaming model over time

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.

Chapter 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.

Bibliography

[1] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. Developing a predictive model of quality of experience for internet video. In *Proceedings of the 2013 ACM Conference on SIGCOMM*.

[2] Christopher Müller, Stefan Lederer, and Christian Timmerer. An evaluation of dynamic adaptive streaming over http in vehicular environments. In *Proceedings of the 4th Workshop on Mobile Video*, 2012.

[3] Guibin Tian and Yong Liu. Towards agile and smooth video adaptation in dynamic http streaming. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, 2012.

[4] Ricky K. P. Mok, Xiapu Luo, Edmond W. W. Chan, and Rocky K. C. Chang. Qdash: A qoe-aware dash system. In *Proceedings of the 3rd Multimedia Systems Conference*, 2012.

[5] V. Joseph and G. de Veciana. Nova: Qoe-driven optimization of dash-based video delivery in networks. In *INFOCOM, 2014 Proceedings IEEE*.

[6] Thomas Wirth, Yago Sánchez, Bernd Holfeld, and Thomas Schierl. Advanced downlink lte radio resource management for http-streaming. In *Proceedings of the 20th ACM International Conference on Multimedia*, 2012.

[7] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM Conference on SIGCOMM*.

[8] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. Confused, timid, and unstable: Picking a video streaming rate is hard. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, 2012.

[9] Te-Yuan Huang, Ramesh Johari, and Nick McKeown. Downton abbey without the hiccups: Buffer-based rate adaptation for http video streaming. In *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*, 2013.

[10] Ahmed Hamza and Mohamed Hefeeda. A dash-based free viewpoint video streaming system. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, 2014.

[11] Zied Aouini, Mamadou Tourad Diallo, Ali Gouta, Anne-Marie Kermarrec, and Yannick Lelouedec. Improving caching efficiency and quality of experience with cf-dash. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, 2014.

[12] Danny H. Lee, Constantine Dovrolis, and Ali C. Begen. Caching in http adaptive streaming: Friend or foe? In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, 2014.

[13] Ali C. Begen Saamer Akhshabi, Lakshmi Anantakrishnan and Constantine Dovrolis. What happens when http adaptive streaming players compete for bandwidth? In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, 2012.

[14] Sajid Nazir, Ziaul Hossain, Raffaello Secchi, Matthew Broadbent, Andreas Petlund, and Gorry Fairhurst. Performance evaluation of congestion window validation for dash transport. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, 2014.

[15] Feng Wang, Jiangchuan Liu, and Minghua Chen. Calms: Cloud-assisted live media streaming for globalized demands with time/region diversities. In *INFOCOM, 2012 Proceedings IEEE*, 2012.

[16] Cong Wang and Michael Zink. On the feasibility of dash streaming in the cloud. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, 2014.

[17] D. De Vleeschauwer, H. Viswanathan, A. Beck, S. Benno, Gang Li, and R. Miller. Optimization of http adaptive streaming over mobile cellular networks. In *INFOCOM, 2013 Proceedings IEEE*.

[18] R. Bhatia, T.V. Lakshman, A. Netravali, and K. Sabnani. Improving mobile video streaming with link aware scheduling and client caches. In *INFOCOM, 2014 Proceedings IEEE*.

[19] S. Cicalo, N. Changuel, R. Miller, B. Sayadi, and V. Tralli. Quality-fair http adaptive streaming over lte network. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*.

[20] Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer. Equation-based congestion control for unicast applications. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2000.

[21] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *SIGCOMM Comput. Commun. Rev.*, 27(3):67–82, 1997.

[22] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling tcp throughput: A simple model and its empirical validation. In *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1998.

[23] Martin Swany and Rich Wolski. Multivariate resource performance forecasting in the network weather service. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, 2002.

[24] S. Vazhkudai, J.M. Schopf, and I. Foster. Predicting the performance of wide area data transfers. In *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, 2002.

[25] Yin Zhang and Nick Duffield. On the constancy of internet path properties. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, 2001.

[26] Qi He, Constantine Dovrolis, and Mostafa Ammar. On the predictability of large transfer tcp throughput. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*.

[27] M. Mirza, J. Sommers, P. Barford, and Xiaojin Zhu. A machine learning approach to tcp throughput prediction. *Networking, IEEE/ACM Transactions on*, 18(4):1026–1039, 2010.

[28] Vengatanathan Krishnamoorthi, Niklas Carlsson, Derek Eager, Anirban Mahanti, and Nahid Shahmehri. Quality-adaptive prefetching for interactive branched video using http-based adaptive streaming. In *Proceedings of the ACM International Conference on Multimedia*, 2014.

[29] Mohammad Ashraful Hoque, Matti Siekkinen, and Jukka K. Nurminen. Using crowd-sourced viewing statistics to save energy in wireless video streaming. In *Proceedings of the 19th Annual International Conference on Mobile Computing &#38; Networking*, 2013.

[30] Hyunwoo Nam, Kyung Hwa Kim, Bong Ho Kim, D. Calin, and H. Schulzrinne. Towards dynamic qos-aware over-the-top video streaming. In *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on*.

[31] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[32] Simone Basso, Antonio Servetti, Enrico Masala, and Juan Carlos De Martin. Measuring dash streaming performance from the end users perspective using neubot. In *Proceedings of the 5th ACM Multimedia Systems Conference*, 2014.