IRate: Initial Video Bitrate Selection System for HTTP Streaming

Ricky K. P. Mok, *Student Member, IEEE*, Weichao Li, *Student Member, IEEE*, and Rocky K. C. Chang, *Member, IEEE*

***Abstract*— Many HTTP streaming video systems have been developed and widely deployed in recent years. Previous efforts were mainly spent on improving the caching of videos or proposing mid-stream measurement methods to update the best bitrate. However, since the video length is often short, the mid-stream measurement may not even converge to the best bitrate due to insufficient bandwidth estimates. On the other hand, because of diversified Web infrastructure, estimating the actual network quality at the pre-stream stage is increasingly challenging for video service providers. In this paper, we propose IRate, which enables video service providers to proactively profile clients' streaming performance by carrying out pre-stream measurement in the Content Delivery Network (CDN). With the measurement results, the video stream can start at the best video quality at the onset of streaming. This is especially beneficial to short video clips, which are very popular in the Internet today. IRate is composed of a probe kit and a quality oracle. The probe kit utilizes the pre-stream time window (e.g., user's think time and pre-roll advertisement) for measuring network quality by running a lightweight measurement script on the Web page to induce probe packets from the IRate middlebox on the server side. With the measurement results, the quality oracle estimates the clients' streaming performance by determining the highest initial bitrate with a pre-trained decision tree. Our testbed results show that IRate is able to achieve 80% accuracy in determining the bitrate within 10s. By having a better estimate of the best initial bitrate, the buffering time and rebuffering events are significantly reduced in HTTP streaming. Furthermore, the stability and the efficiency in dynamic adaptive streaming over HTTP streaming are also improved by about 40% and 36%, respectively. Our user quality of experience (QoE) experiment further validates that IRate can improve the QoE by more than 6% and the perceived quality of initial quality by 24% in the actual Internet environment.**

***Index Terms*— DASH, HTTP streaming, initial video bitrate, QoE.**

I. Introduction

ONLINE video streaming is unarguably one of the most popular web applications nowadays. By leveraging the web infrastructure, the video streaming service is often delivered through HTTP. An entire video may be downloaded using a single HTTP request or in "chunks" through multiple HTTP range requests from video caches [1]. The initial video bitrate/quality level is usually set to some default values. For example, the default initial bitrates for YouTube are set according to the size of the video player [2], while other non-US providers only use some predefined quality based on the available quality of the requested video.

Dynamic Adaptive Streaming over HTTP (DASH), a recent advance in the HTTP streaming, supports adaptive bitrate streaming that allows a video player to switch among several bitrates/quality levels during a video viewing session. A common implementation is to encode an entire video into chunks each of which consists of a few seconds of the video using different bitrates. If bitrate switching is required, the corresponding chunk will be requested. Different methods have been proposed for determining the best bitrate [3]–[5]. They are often based on *mid-stream network measurements* to estimate the available bandwidth during the playback of the stream. A quality adaptation algorithm then computes the bitrate corresponding to the measurement results, e.g., [6], [7].

An important problem, but receiving little attention, is to determine the best *initial* bitrate (BIBR), because the default settings employed by the current system cannot easily accommodate diverse end-to-end network quality (e.g., fixed network vs. wireless network) and end-systems (e.g., handheld device vs. large-screen HDTV). Without any network measurement to base on, a "safe" approach is to choose a conservative bitrate to start with. This approach, however, results in suboptimal quality of experience (QoE), especially for short video clips, because of the following two reasons.

1) Even though the quality adaptation algorithm in DASH can ramp up the video bitrate and find the best rate quickly on the client side, the video player cannot discard the low-bitrate video segments already downloaded during this process. The user therefore has to suffer from low initial video quality for a period of time. This start-up phase can be longer than 150s of video [8]. Our subjective tests (cf. §II-C) show that using a low initial bitrate can degrade the overall QoE.
2) Starting from a low video bitrate also induces unnecessary quality switching during the adaptation process. It has been shown that a frequent switching of bitrates can hurt user engagement [9] and also the QoE [10]–[12].

In this paper, we propose to introduce *pre-stream network measurement* into existing video streaming systems for

Manuscript received September 11, 2015; revised January 20, 2016; accepted February 15, 2016. Date of publication April 27, 2016; date of current version June 6, 2016. This work is partially supported by an ITSP Tier-2 project grant (ref. no. GHP/027/11) from the Innovation Technology Fund in Hong Kong and a research grant from the Joint Universities Computer Center of Hong Kong (ref. no. H-ZL17). *(Corresponding author: Rocky K. C. Chang.)*

The authors are with The Hong Kong Polytechnic University, Hong Kong (e-mail: cs.rickymok@connect.polyu.hk; csweicli@comp.polyu.edu.hk; csrchang@comp.polyu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2016.2559078

TABLE I
STREAMING VIDEO SERVICES PROVIDED BY MAJOR VIDEO SERVICE PROVIDERS

Providers	Streaming type	No. of bitrates/quality levels supported	Initial bitrate/quality	Advertisement
YouTube	DASH-enabled	2-6	Player size [2]	Non-skippable/Skippable after 5s
Netflix	DASH-enabled	12-14 [13]	Country [14]	No
Hulu	DASH-enabled	3	N/A	No
Vimeo	Static	2	Predefined (HD)	No
Dailymotion	Static*	2-5	Predefined (HQ) [15]	Skippable after 4s
Youku	Static*	3	Predefined ($\sim$200 Kbps)	Non-skippable
Tudou	Static*	3-4	Predefined ($\sim$500 Kbps)	Non-skippable

*These results are based on our own tests as we cannot obtain the data from public sources.

determining the BIBR with a reasonable accuracy before the video playback starts. Integrating such measurement is, however, challenging. There are three main challenges. First, we need to identify a suitable time frame for carrying out the measurement, such that the measurement results can be ready before the onset of video. On the other hand, to avoid measurement inaccuracy at the browser level, we migrate the measurement core to the server side. At the same time, we need to consider whether the measurement can be seamlessly integrated into the existing web architecture and provide accurate path quality metrics for estimating the BIBR.

In this paper, we present a practical system to determine the Initial bitRate (abbv. *IRate*). Our user-behavior driven design helps explore possible time frames for pre-stream measurements. Moreover, we exploit a number of tactics in the TCP/IP layer and web technology to implement the measurement component. The measurement core of IRate is designed as a measurement box, which can be easily deployed along with the existing video caches and servers to collect accurate network path quality data at the server side. The measurement core can accommodate packet-pair based bandwidth estimation algorithms to measure the network quality at the server side. By imbedding a script in web pages, clients can measure dedicated IRate-enabled video caches through the browser. Based on the measurement results, IRate profiles clients by determining their BIBR to video caches. The web server utilizes this information and redirects users to a better video cache which could serve the highest BIBR for the best QoE.

We have conducted extensive testbed experiments to evaluate IRate's accuracy of estimating the BIBR and its robustness when operating under diverse network conditions. Our results show that using IRate can achieve a 80% accuracy using only 10s of measurement data. It also achieves better stability and higher efficiency than DASH with the default setting. We further conducted user assessments to compare the performance of IRate and the predefined approach in terms of QoE in actual Internet environment. Our results show that IRate can increase the QoE by more than 6% and the perceived quality of initial quality by 24%.

In §II, we first survey the default initial bitrate settings used in several popular video streaming services and a subject assessment on different initial bitrate settings. We present our user-behavior driven approach for estimating the BIBR in §III and then describe IRate, our implementation based on this approach, in §IV. §V and §VI present our testbed and user experiment results for evaluating the accuracy and robustness of IRate. We then discuss some limitations of IRate in §VII. After highlighting related works in §VIII, we conclude this paper in §IX.

II. BACKGROUND

A. Default Settings for Initial Video Quality

Video streaming providers generally offer several quality levels or video bitrates for each video. However, whether they support adaptive bitrate streaming or not, they set their default initial bitrates oblivious to the performance of the end-to-end path between the client and video server. We have surveyed several video content providers on their default settings, and summarized in Table I. Since the default settings are not always stated clearly in public documentations, we have performed testings to uncover their initial bitrate settings. However, we cannot perform the tests for Hulu, because its service are not available locally. We also show in the table whether they support adaptive streaming, the number of supported bitrates, and whether they have any pre-roll advertisement.

As shown in Table I, there are three types of initial bitrate settings. YouTube sets the initial bitrate to the highest quality based on the video player size and the quality of original uploaded video [2]. We also observe from our tests that the corresponding quality levels for "small", "large", and "full screen" are 360p, 480p, and 1080p, respectively. Netflix, on the other hand, determines the default video quality settings by countries. For example, for Canada and Brazil users, the quality level is set to "good", and other users are set to "best" [14]. We are, however, unable to determine the setting for Hulu. As for the other three, our tests show that they choose predefined quality for the initial bitrates. For example, Youku streams video clips in a standard definition. Dailymotion selects "HQ" as the default bitrate [15], where we observe that the resolution can be 360p or 480p.

We also perform supplementary tests to verify whether the default settings for new users will be changed according to different network conditions. We run a Windows 7 virtual machine as the client. We then configured the capacity of the virtual machine's network interface in the `VMware` to 10Kbps, 200Kbps, and unlimited. The downstream link for receiving video is therefore bandwidth throttled. After setting the network, we arbitrarily select at least three video clips from the front page to stream from the YouTube, Vimeo, Dailymotion, Youku, and Tudou websites with Internet Explorer. To emulate

TABLE II
A COMPARISON OF THE DEFAULT INITIAL BITRATES AND THE BIBRS FOR SAMPLE VIDEOS

Providers	Possible bitrates of the sample video	Default initial bitrates	Source server	Average throughput (BIBR)	
				Overall	Start-up
YouTube	144p (82Kbps), 240p (226Kbps), 360p (546Kbps), 480p (756Kbps), 720p (1577Kbps), 1080p (3364Kbps)	Based on the player size	202.40.221.15	9.22Mbps (1080p)	3.63Mbps (1080p)
Dailymotion	Low (246Kbps), Medium (459Kbps), Standard (829Kbps), HD (1618Kbps)	Standard	188.65.126.17	0.66Mbps (Medium)	1.23Mbps (Standard)
Youku/ Tudou✠	Standard (285Kbps), High (547Kbps), Super High (1197Kbps)	Standard	23.236.118.48	1.08Mbps (High)	1.36Mbps (Super High)

✠ In our experiment, we found that Youku and Tudou streamed the video from the same video server.

new clients with no prior knowledge of the throughput, the *InPrivate* mode is used to avoid local caching or any tracking of user preferences. We believe the videos shown on the front page are popular videos which have been cached nearby CDN caches. All the tests show that their default values do not change for the three different downlink bandwidth scenarios, thus further supporting that their initial bitrates are oblivious to the path performance.

B. Impacts of the Default Settings

To understand the impact of the default settings, we employ a two-step approach. The first step is to measure the degree of discrepancy between the default setting and the BIBR. Then, we present a subjective experiment to show how the default settings impact on the user perceived quality.

1) Discrepancy Between Default Settings and the BIBR: We performed throughput measurements for YouTube, Dailymotion, Youku, and Tudou. Our methodology is to arbitrary select a video from the front page of each provider. We captured the traffic using `wireshark` while playing the video clip. Then we computed the average throughput of the whole video stream. Furthermore, to capture the potential influences from the TCP slow-start, we also compute the average throughput of downloading the first 250KBytes of video data as the start-up throughput. For each video, we repeated the download for five times. Finally, the BIBR is determined by mapping to bitrate closest to the measured throughput.

Table II shows the results of comparing the default initial bitrates and the BIBRs based on the throughput measurement. Except for YouTube, the initial bitrates set by the other three providers do not agree with the BIBRs. Both Youku/Tudou can provide a higher quality than the default settings, whereas Dailymotion's exceeds the actual throughput provision. YouTube, on the other hand, can provide the highest quality for the BIBR. Surprisingly, the start-up throughput in Youku/Tudou and Dailymotion is higher than the average throughput of the whole video stream. The start-up throughput for both providers can support one quality level higher than the overall one. Even though YouTube shows a lower start-up throughput, the start-up throughput is still sufficient for supporting the same (highest) video bitrate. Since their default settings for all player sizes are less than the BIBR, theirs are sufficient for guaranteeing the best QoE at the onset of the viewing. YouTube's exceptional throughput performance is due to their local cache servers [16], our laboratory's high-speed network (100Mbps), and their aggressive buffering strategies [1]. For the other three providers, we performed `traceroute` with TCP SYN packets and confirmed that the video servers are not located locally. It is believed that the throughput is limited by the overseas link.

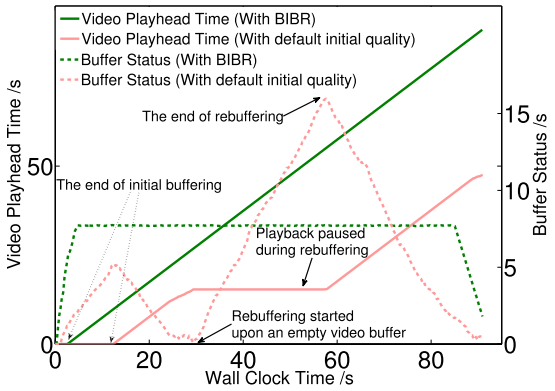


Fig. 1. Video playhead times and buffer statuses for default initial bitrate and BIBR in classic HTTP streaming under a lossy network condition.

We next present two testbed traces to illustrate the impact of suboptimal initial bitrates (the full set will be presented in §V). The first case is classic HTTP streaming, and there are packet losses on the path. We plot in Fig. 1 the video playhead times (the solid lines) and buffer status (the dotted lines) for two cases: a default initial bitrate of level 3 in a scale of 0 to 4 (the light red) and the BIBR (the dark green). Notice that the default bitrate is too high for the network condition. Hence, the playback suffers from a long initial buffering time ($>$10s). It then pauses at around 30s for rebuffering and resumes at 60s. The case using the BIBR, on the other hand, chooses the lowest bitrate, thus yielding a smooth playback.

The second case shows the first 15s of the traces for DASH streaming with a default initial bitrate and the BIBR. The network condition is better than the first case, which can support level 4, the highest quality. Fig. 2 shows the quality levels against the video playhead time. The BIBR is used throughout the entire period, whereas the default case is started with the lowest quality (level 0) and reaches the final quality level after two consecutive up-switchings at 4s and 8s. The shaded region therefore refers to the amount of under-utilized bandwidth for the default case. Depending on

TABLE III
SUMMARY AND THE MOS OF DIFFERENT QUALITY TRANSITION SCHEMES

ID	Transition	MOS	$\overline{E_{init}}$	$\overline{E_{overall}}$	N	ID	Transition	MOS	$\overline{E_{init}}$	$\overline{E_{overall}}$	N
R_{-2}	l_0	3.21	3.07	3.09	129	R_7	$l_1 \rightarrow l_2 \rightarrow l_4$	4.00	3.83	4.11	35
R_{-1}	l_4	4.16	4.23	3.99	90	R_8	$l_1 \rightarrow l_3 \rightarrow l_4$	4.26	4.03	4.13	31
R_0	$l_0 \rightarrow l_4$	3.92	3.05	3.65	130	R_9	$l_2 \rightarrow l_3 \rightarrow l_4$	4.13	3.63	4.03	40
R_1	$l_1 \rightarrow l_4$	4.06	3.86	4.06	35	R_{10}	$l_0 \rightarrow l_1 \rightarrow l_2 \rightarrow l_4$	3.78	3.24	3.49	37
R_2	$l_2 \rightarrow l_4$	4.15	3.94	3.88	33	R_{11}	$l_0 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4$	4.00	3.50	3.87	30
R_3	$l_3 \rightarrow l_4$	4.13	3.98	3.90	40	R_{12}	$l_0 \rightarrow l_1 \rightarrow l_3 \rightarrow l_4$	3.63	3.47	3.63	30
R_4	$l_0 \rightarrow l_1 \rightarrow l_4$	3.55	3.00	3.50	38	R_{13}	$l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4$	4.23	3.58	4.02	43
R_5	$l_0 \rightarrow l_2 \rightarrow l_4$	3.90	3.00	3.57	30	R_{14}	$l_0 \rightarrow l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4$	3.77	3.42	3.74	31
R_6	$l_0 \rightarrow l_3 \rightarrow l_4$	3.79	2.79	3.29	34						

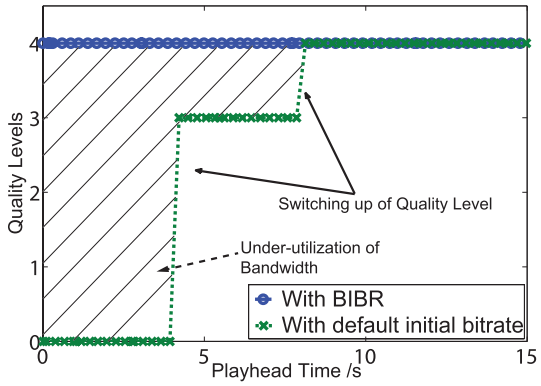


Fig. 2. Quality levels vs. video playhead times for default initial bitrate and BIBR under DASH streaming in a good network condition.

the video segment length, buffer size and the aggressiveness of the quality adaptation algorithm, users could experience an unstable video quality for more than 150s [8].

C. The Impact to the QoE

The default bitrate can definitely impact to the QoE. If the bitrate is too high, the playback will be stalled by rebuffering events. Previous works had shown that these events can degrade the QoE [17] and also the user engagement [9]. However, we found that using a conservative choice cannot meet user expectations and impacts to the overall QoE. To quantify the impact of low default quality to the QoE, we carry out subject assessments to measure the user perceived quality under different initial settings and transition schemes. Our assessment emulates users streaming videos with DASH using a high-speed network connection, which can support the highest bitrate of the video without any rebuffering. But, the initial bitrate is set to a suboptimal value and then switches up until the highest bitrate.

To emulate different quality adaptation methods, we compose 17 (15+2) cases, denoted by R_{id}, to switch up the quality levels to the BIBR as listed in Table III. The transitions are represented by $\rightarrow$. For example, $l_x \rightarrow l_y$ means the quality levels switched from l_x to l_y. Each level is played for two segments before the next transitions. R_{-2} and R_{-1} are control cases emulating the worst and the best overall picture quality, respectively. These two cases do not have any quality change in the entire video playback. For other cases, the quality levels will monotonically switch up to l_4 and keep steady until the end. The video is streamed through a Flash-based customized video player, which is implemented on top of the Strobe Media Playback, and allows us to override the internal quality adaptation algorithm and switch the quality levels according to the preset rules.

We download four different kinds of High Definition video clips from the YouTube (including News, sports video, music video, and movie trailer) to randomize the effect from the video content. Then, the video clips are trimmed to 1 minute and encoded into five quality levels (denoted by l_0 to l_4) according to Adobe's encoding recommendation [18] (first five levels of Variant 3), where l_0 is the lowest quality. The video segment length is 4s, which is the default value for the Adobe HTTP Dynamic Streaming's file packager. We argue that the video length in our subjective test is sufficient and realistic. Measurement studies of YouTube [19], [20] and Akamai [21] show that short video clips are still very popular in today's Internet.

We employ Amazon Mechanical Turk (MTurk) and CrowdFlower to carry out subjective tests using crowdtesting approach [22]. We implement a web-based crowdtesting platform similar to [23]. The workers are instructed to visit our test system, which is a website hosted on an Amazon EC2 instance. Because this assessment focuses on the effect of picture quality instead of the playback smoothness, we use CloudFlare CDN to cache the video data to mitigate any rebuffering events due to insufficient network throughput at the server side. From the log returned by the video player, we confirm that no rebuffering event is observed. Each worker is asked to watch and rate his/her perceived quality of the four video clips. Among the four video clips to be watched by the worker, one of them is a control case (either R_{-2} or R_{-1}) and others are randomly selected from the 15 cases with quality transitions. After finished playing each video, a questionnaire immediately shows up and requires the worker to rate the overall perceived quality in a 5-point Likert scale (1: Bad - 5: Excellent) [24]. Furthermore, we also measure their expectation by asking them whether the initial and overall picture quality meets their expectations in a 5-point Likert scale (1: Strongly disagree - 5: Strongly agree), denoted by E_{init} and $E_{overall}$, respectively. Each worker is awarded from USD \$0.5 to \$1. We successfully recruited 209 subjects to perform this assessment after screening our low-quality workers using the methodology in [25].

The columns MOS, $\overline{E_{init}}$, $\overline{E_{overall}}$, and N in Table III show the mean opinion score, the mean value of the expectation rating of initial, the overall picture quality, and the number

of samples, respectively. All the cases are evaluated by at least 30 subjects. The cases with darker background color have more video quality transitions. We can see that all the transition schemes (R_0, R_4, R_5, R_6, R_{10}, R_{11}, R_{12}, R_{14}) started from the lowest quality level, l_0, have a lower MOS, E_{init}, and $E_{overall}$ than the similar schemes (R_1, R_7, R_8, R_{13}) started from l_1. We further compute the correlation coefficient among these three metrics. The MOS values show significant positive correlation with both E_{init} ($r = 0.42$, $p < 0.001$) and $E_{overall}$ ($r = 0.71$, $p < 0.001$). Although the correlation coefficient for E_{init} is smaller than $E_{overall}$, we can conclude that the initial quality shows significant influence on the overall QoE. Furthermore, R_{14} shows a lower MOS than other schemes with less number of transitions (e.g., R_5, R_6, R_{11}). This reveals that a longer duration of the transition phase can also hurt the QoE.

III. User-Behavior Driven Design

A. Design Goals

We design IRate by considering three aspects—users, video service providers, and system accuracy. Our objective is to derive the BIBR from pre-stream measurement results. Moreover, the current infrastructure and user habit can be preserved. From users' point of view, the pre-stream measurement cannot disturb the normal user behavior, especially blocking the onset of the streaming after they selected the video or generating excessive amount of measurement traffic. On the other hand, the design of IRate has to be practical for service providers to deploy it to the current distributed video delivery infrastructure. More importantly, IRate needs to collect accurate enough path performance data, in a lightweight way, to determine the BIBR.

These issues are not independent of each other, because the measurement accuracy also depends on the measurement method and measurement duration. The amount of time allocated to the pre-stream measurement is clearly very limited. Moreover, in order not to block the video streaming, the pre-stream measurement must be conducted and completed before the user selection of the video. As a result, the measurement must be conducted in the "background" when the user is making his video selection. As the measurement is now run in parallel with the download of web objects of the current page, the choice of measurement method has to be lightweight. Furthermore, a practical design should also be able to accommodate the distributed architecture of large-scale video delivery systems.

B. Pre-Stream Measurement Window

Our approach to designing IRate is to first study a user's typical behavior before selecting a video to watch. The user behavior will inform us the constraints, as well as the opportunities, for conducting pre-stream measurement to determine the BIBR. The five dark-bordered boxes in Fig. 3 show the main actions performed by a user and his browser:

(1) The user visits the front page of a video streaming website, such as YouTube, which usually displays a video catalog.

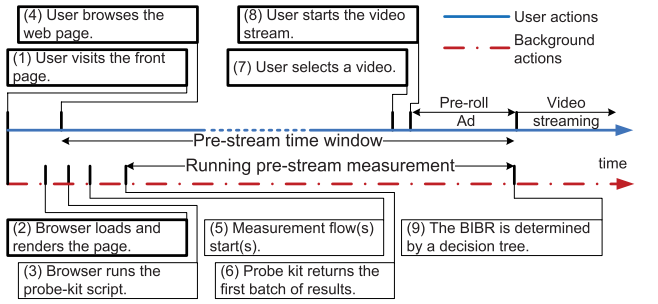


Fig. 3. Typical user actions and background actions before starting a streaming video. The dark-bordered boxes are the actions for typical video streaming, whereas the light-bordered boxes are added for IRate-enabled video streaming.

(2) The browser starts downloading and rendering the web objects in the front page.

(4) When the page is partially parsed and rendered, the user may browse the page and acquire the video clips information from the page.

(7) After selecting a video to watch, the user clicks on the video's hyperlink/thumbnail to view the video page.

(8) The user may need to explicitly start the video streaming. A pre-roll video advertisement may be shown before streaming the requested video.

We refer the period between action (4) and the onset of the video streaming to as a *pre-stream time window*. In this period, the user may engage in different types of activities. One of them is considering which page to visit next, which can be modeled as user think time. A measurement study [26] shows that the user think time for YouTube is about 30s, which is longer than traditional web transactions. Another typical event occurring in the pre-stream time window is showing short video advertisements (Ads), also known as pre-roll Ads, (typically 15-30s) [27], [28]. The Ads in some sites are not skippable. Even though YouTube's TrueView advertising package [29] and Dailymotion provide a "skip" option for users to jump over the Ad, users have to wait for at least 4s. As a result, the pre-stream time window will provide a window of about 34(30+4)s for conducting the pre-stream measurement.

C. Measuring Network Path-Quality Metrics

Using a more conservative estimate, the window for pre-stream measurement is about 10s. Within such a small time window, it is very challenging to collect accurate enough measurement data for determining the BIBR. A general approach is to estimate the network throughput and then select the video bitrate that is closest to the throughput. There are a number of methods/tools for measuring the (available) throughput. Besides flooding based method, packet pairs or packet trains can be used to measure the available bandwidth. Pathload [30], pathChirp [31], and PTR [32] employ probe rate model, which measures the network by self-induced congestion. Spruce [33] and IGI [32] are based on the probe gap model, which relies on the timing information carried in the time gap between a pair of probe packets after traversing

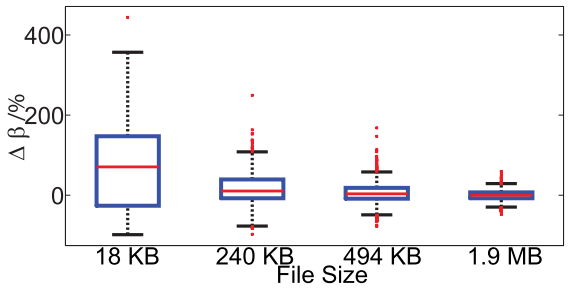


Fig. 4. The percentage difference of average throughput against different web object sizes.

the network. However, these tools are not designed for web clients, because they often need raw socket to send packets in a particular pattern.

To enable web clients to perform network measurement, a number of browser-based measurement tools have been developed in recent years. They are based on Adobe Flash [34], Java applet [35], and JavaScript [36]. Browsers allow these tools to elicit HTTP requests or initiate connections to servers. However, browsers are run on the application layer, which has limited access to the information from the network layer. It is hard to capture timestamps of specific probe packets and to send probe packets in a particular pattern. Besides, these tools may suffer a higher delay as they are running on the application layer [37]. Hence, they, such as speedtest [34], often measure the throughput by flooding, which incurs high overhead in order to obtain reasonably accurate results.

We investigate the accuracy of flooding based method by downloading five objects of different size (cf. §V-A.2 for details). We compare the average TCP throughput obtained between the largest object (4.3MBytes) and those obtained from the smaller objects by computing the percentage differences, $\Delta\beta_s$, by (1).

$$\Delta\beta_s = \frac{\beta_s - \beta_{4.3\text{MB}}}{\beta_{4.3\text{MB}}} \times 100\%, \tag{1}$$

where β_s is the throughput measured by using one of the smaller objects, and $\beta_{4.3\text{MB}}$ is the throughput measured by the 4.3-MB object.

Fig. 4 shows a box-and-whisker plot of $\Delta\beta$ of the four small objects. The lower and upper edge of box gives the 25^{th} and the 75^{th} percentile, respectively, and the central line inside is the median. The lower and upper whiskers respectively are the minimum and maximum, after excluding the outliers. Outliers are defined as the data points exceeding 1.5 of the upper quartile, and those below the minimum are less than 1.5 of the lower quartile, which are marked as dots outsides the whiskers. From the figure, we can see that using small objects (18KBytes and 240KBytes) show a large disagreement and variance to $\beta_{4.3\text{MB}}$. Some cases overestimate the throughput by 200%. The difference reduces as the file size increased to 1.9MBytes. The inter-quartile range of $\Delta\beta_{1.9\text{MB}}$ is about 14.8%. Hence, high overhead of this kind of speed measurement is unavoidable.

Another important consideration for measuring the network path quality is which side (user or video server) to conduct the measurement. Recall that the pre-stream measurement is to be performed before selecting the video to download. However, the measurement may have to terminate once users make the selection. Client-side tools may not be able to feedback the results timely to the server when the measurement is still in progress.

This constraint therefore motivates us to employ a server-side active measurement paradigm for IRate. In this paradigm, the server side masters the measurement process, while the browser running at the client side is required only to send some dummy data to the server. This paradigm has several advantages:

1. It can provide more accurate network-layer measurement by optimizing the implementation and unifying the measuring agent. For example, a hardware-assisted measurement middlebox could be implemented based on Endace DAG card [38] or NetFPGA for high performance and accuracy.
2. The implementation can be incorporated into other server-side in-line network appliances (e.g., firewall, IPS), shaper [39], or measurement middleboxes (e.g., QDASH [7]). These middleboxes are very common in today's enterprise network [40].
3. The browser does not need to install extra plug-ins or tools to cooperate with the measurement. In IRate, a probe-kit script is written in commonly used Web technology (e.g., a Flash object or HTML5 script), and is embedded in a web page to induce data for measurement.
4. The server-side measurement facilitates the BIBR estimation, because all the data are collected on the server side and are readily available for BIBR estimation.

Our design does not restrict the probing method. But, in particular, we use a server-side version of TRIO [41] for network measurement. TRIO is a light-weight, non-cooperative packet-pair based measurement method. Fig. 5 shows a box-and-whisker plot of the amount of data used in 10-second IRate measurement across different network conditions against the RTTs. The dotted blue and orange lines, respectively, show the minimum number of data used in downloading a 494-KB object and a 1.9-MB object in Speedtest Mini package [42] as reference points. The median number of data used in IRate is less than that of downloading the 494-KB object. Although IRate consumes more data in some low-RTT cases, all of them are much less intrusive than downloading the 1.9-MB object. Hence, IRate is more light-weight than the flooding based measurement tools.

D. Methods for Estimating the BIBR

We use the set of path-quality metrics collected in the pre-stream measurement to estimate of the BIBR. The metrics includes the delay, delay jitter, loss rate, reordering rate, and capacity for two paths: video server→user and user→video server. A major concern of estimating the BIBR is the computational speed, because the web server has to wait for the estimated BIBR for generating the parameters on the video

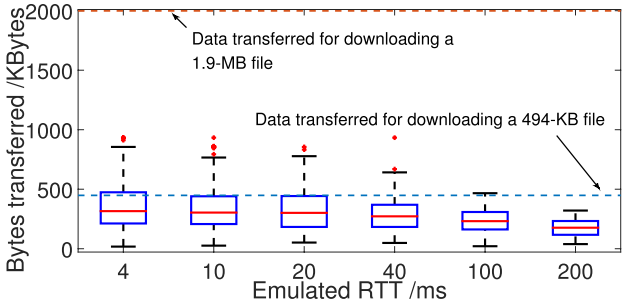


Fig. 5. The number of Bytes transferred in 10-second IRate measurement against RTTs.

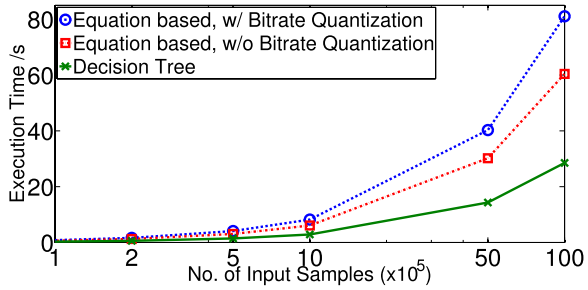


Fig. 6. An execution-time comparison of the equation-based and decision tree methods for estimating the BIBR.

page to control the initial bitrate. Otherwise, the IRate could become a bottleneck of the video delivery system. Instead, we can tolerate larger errors as the difference of bitrates among quality levels are often large.

We therefore consider a much faster and lightweight approach that assumes a pre-computed throughput model. Two examples are equation-based [43], [44] and decision tree. The equation-based method mathematically models the steady-state throughput of a TCP flow using round-trip delay, packet loss rate, and bottleneck capacity. The decision tree, on the other hand, is based on a set of training data to construct a decision tree for determining the BIBR directly.

It is not our goal to compare the two methods' accuracy in this paper. Instead, we focus on their computational efficiency and scalability, because of the small time window for measurement and large number of clients. Fig. 6 shows a comparison of execution time between Padhye's model [43] for TCP Reno and the decision tree (cf. §IV-B for details). We implemented both methods with `perl` and randomly generated 100K to 10M sets of network path metrics as the input to predict the quality levels. For the equation-based approach, we use our bitrate quantization in (2) (in §IV-B) to convert the estimated throughput to a video quality level. We ran both methods on a Dell R210 Rack Mount server and used `time` command in Linux to record the execution times.

The results show that the decision tree (marked with crosses) is much faster than the equation-based method (marked with circles) by nearly three times. The execution time could be made shorter if we skip the bitrate quantization step (marked with squares). However, the equation-based method is still two times slower than the decision tree. The reason, we believe, is that the CPU requires more clock cycles for

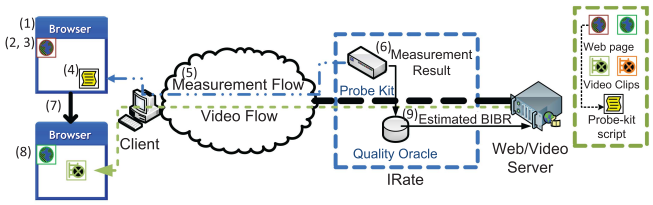


Fig. 7. The main steps in IRate-enabled video streaming.

computing floating points than determining cases in decision tree. Hence, we adopt the decision tree approach in IRate as it can scale to a large number of users. We also note that we have considered other machine learning approaches, such as [45]–[47], but they are not suitable for IRate for various reasons. For example, Mirza's work [46] uses SVR to estimate TCP throughput, but the input metrics must be obtained from a cooperative measurement which is hard to deploy in browsers. Nunes et al. [45] combine a number of machine learning algorithms for predicting RTTs, but they cannot predict packet loss, which is a key metric for estimating TCP throughput.

With IRate-enabled video streaming, four more light-bordered boxes ((3), (5), (6), and (9)) are added to Fig. 3 for background actions. Besides, a probe-kit script is embedded to the front page which is accessed by users in (1).

(3) The browser runs the probe-kit script.
(5) The probe-kit script starts establishing TCP measurement flow(s) with the video server.
(6) Pre-stream measurement begins when receiving the first batch of measurement results.
(9) Based on the network measurement data, a decision tree method is used to determine the BIBR for the video streaming.

IV. An IRate Implementation

We have implemented a prototype called IRate to estimate the BIBR based on the methods discussed in the last section. The square box in the middle of Fig. 7 depicts IRate which is located before the streaming video website and video servers. To simplify the ensuing discussion, both the website which is first contacted by a user and the video server are assumed located in the same domain.

IRate consists of two building blocks: a *probe kit* and a *quality oracle*. The probe kit (cf. §IV-A) is responsible for measuring the network performance during the pre-stream time window. It is transparent to both browser and video server, as it intercepts the measurement packets destined to the video server. The quality oracle (cf. §IV-B) maintains a decision tree for determining the BIBR with the network performance data as inputs. Note that the numbers inside parentheses correspond to those in Fig. 3.

The design of IRate has considered the deployment in the large-scale video delivery infrastructure. Fig. 8 shows the way of deploying IRate when the front-end and video servers are under different domains. We assume the front-end web server is videoweb.com for hosting the web pages of the video site. There are two video caches at different locations and domains to the web server, namely v1.cache.com and v2.cache.com.

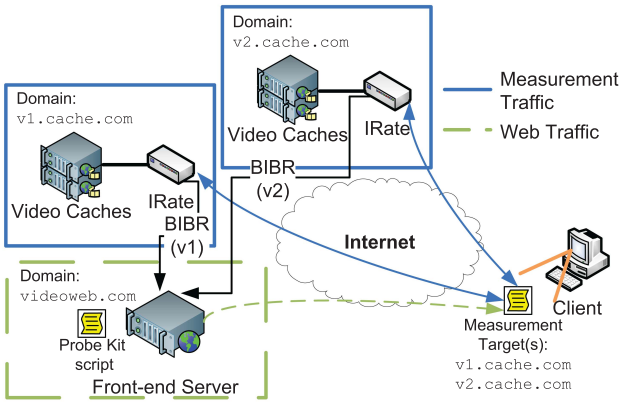


Fig. 8. Deploying IRate in a large-scale video site.

The key is that the probe kit script and the IRate middlebox can be separately located at the front-end server and the video caches, respectively. When a client reaches the front-end server, the server can assign one or more IRate-enabled video cache(s) as the measurement target in the probe kit script. By utilizing the cross domain policy in Adobe Flash and WebSocket, the client can measure the two video caches at different domains to the web server. Then, the front-end server can query the respective IRate middlebox for the BIBR of the client at the end of pre-stream timing window.

A. Probe Kit

The probe kit offers a *probe-kit script* and a *network measurement middlebox*. The measurement middlebox is located on the incoming path to a video server, and the probe-kit script is run at the user's browser. Together, they allow the middlebox to measure the quality of the network path between the middlebox and the user without the user's and video server's intervention. As pointed out in the last section, this server-side measurement paradigm alleviates the user from installing any measurement tool (e.g., Wireshark) and browser plugin (e.g., Fathom [48]).

1) Probe-Kit Script: The probe-kit script, hosted at the web server, requires high compatibility with various kinds of clients. Therefore, in our implementation, we prepare two versions of the probe-kit script using Adobe Flash and HTML5 WebSocket. Adobe Flash is still a de facto standard in Windows clients, while WebSocket is supported by latest version of Apple Safari browser and mobile clients. To facilitate our testbed experiments, we have also implemented a text-based version of probe-kit script which can be run on Linux clients, including PlanetLab nodes.

When the script is executed at the browser, it calls `Socket` in Flash ActionScript or the `WebSocket` to establish at least one TCP connection to the video server as measurement flows. It then prepares and sends a long dummy string to the measurement flow's socket buffer. As the data has been sent to the network stack, the delay overhead at the application layer can be mitigated. The script can also receive command from the middlebox to close the measurement flows.

2) Measurement Middlebox: We have implemented a prototype in a Linux box. The middlebox acts as a measuring node, while the user's machine as a remote node. Fig. 9 shows the details of a measurement flow. The probe-kit script establishes at least one TCP connection with the web server through the browser. The middlebox also records the TCP states kept by both sides by examining the packets exchanged. Moreover, when the middlebox detects the IRate URI in the HTTP `POST` message used for measurement flow, it hijacks the flow by terminating the connection to the web server. We use the `NFQUEUE` library to intercept packets passing through the middlebox and raw socket to send out measurement probes.

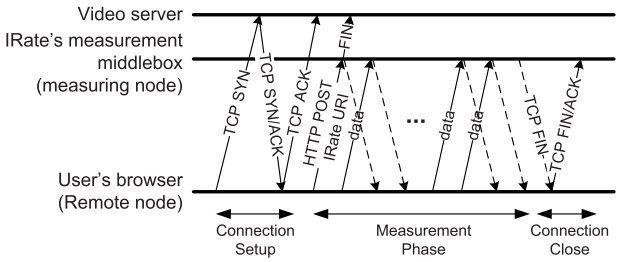


Fig. 9. Probe kit's measurement flow between a user's browser and the measurement middlebox.

After successfully hijacking the measurement flow, the middlebox continues to send probe packets (according to the TRIO probes [41]) to elicit more data from the user's browser for network measurement. The content of the probe packets is a legitimate HTTP response message, emulating a complete HTTP transaction in the measurement flow. This can effectively prevent raising the alarm of firewalls. When the response data prepared by the script are used up, the middlebox terminates the connection. According to [49], the middlebox can measure the round-trip time, and detect packet loss and reordering events on both unidirectional paths based on the response packets. In addition, TRIO cleverly exploits two types of probes to obtain three minimum RTTs to compute both forward and reverse capacities, and another minimum RTT for measurement validation.

B. Quality Oracle

The quality oracle returns an estimated BIBR or initial quality level when given the path measurement data from the probe kit. Due to the storage space consideration, only four to five quality levels are usually available for each video [50]. The quality oracle determines the BIBR or quality level based on a decision tree constructed from a set of training data. As the computation details of decision tree is out of the scope of this paper, we mainly describe the method on how to prepare the inputs and build the decision tree. The single decision tree may not be able to capture the characteristic of different clients. The server may manually build multiple trees with their own historical data to better capture the characteristics of different sets of clients, such as in certain ISPs, ASes, connection methods, or geographical locations [51], [52].

The training data for the decision tree generation is composed of a set of metrics characterizing the performance

TABLE IV
INPUT ATTRIBUTES AND CLASS VARIABLE FOR DECISION TREE BUILDING

Notation	Description
$\bar{d}$	Mean round-trip time, RTT (ms)
$\tilde{d}$	Median round-trip time, RTT (ms)
j_d	Delay jitter (ms)
ι_f	Forward Packet Loss Rate (%)
ι_r	Reverse Packet Loss Rate (%)
c_f	Forward Capacity (Kbps)
$\mathbb{L}^*$	BIBR

Note: *: Class variable

of a network path and the actual BIBR. Table IV summarizes the six network performance metrics considered here. The forward/reverse direction is referenced from the middlebox, because it acts as a measuring node. We do not include the packet reordering rates and reverse capacity, because our experience shows that they are not important for determining the BIBR. On the other hand, the actual BIBR is usually based on the actual throughput measurement. When the quality level is used (instead of the bitrate), the throughput measurement is converted to the number of levels using (2). In our implementation, we employ C4.5 [53] to generate the decision tree **D**. The quality oracle then uses **D** to obtain a BIBR estimate $\widehat{\mathbb{L}}$ for a set of network performance data given by the probe kit.

(2) converts throughput measurement, β, to the BIBR in quality levels, denoted by $\mathbb{L}$. A speed factor, f_{speed}, is multiplied with the average video bitrate of different levels to reduce the influence caused by the bitrate fluctuation for video clips encoded with VBR (Variable Bitrate). f_{speed} is set to 1.25, which is adopted by bandwidth throttling strategy in YouTube [54].

$$\mathbb{L} = \begin{cases} l_{min}, & \text{if } \beta \leq (b(l_{min}) \times f_{speed}), \\ l_{max}, & \text{if } \beta \geq (b(l_{max}) \times f_{speed}), \\ l_i, & \text{otherwise}, \end{cases} \quad (2)$$

where $b(\cdot)$ is a function to map the quality level to its video bitrate. i is an integer such that $b(l_{i-1}) \leq \beta / f_{speed} \leq b(l_i)$. l_{min} and l_{max} represent the minimum and maximum quality levels, respectively.

V. EVALUATION

In this section, we mainly present testbed results to demonstrate how we train the decision tree in the quality oracle for profiling clients as an application of the network performance data we collected. Besides, we evaluate the accuracy of the estimated BIBR by comparing the actual streaming speed.

A. Testbed Experiments

1) Testbed Setup: We setup a testbed to generate data for decision tree building and evaluate IRate. Fig. 10 shows the testbed topology. The web server and the IRate middlebox are directly connected. The middlebox is connected to a result database through another internal network, so that the database traffic will not interfere with the network measurement.

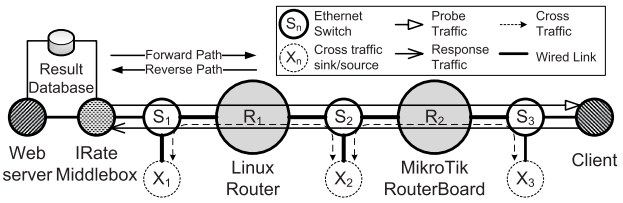


Fig. 10. The testbed topology.

TABLE V
NETWORK PATH PARAMETERS USED FOR GENERATING THE TRAINING DATA

Emulated Path Metrics	Devices	Values
Round-trip Time	R_1	4, 10, 20, 40, 70, 100, 150, 200 (ms)
Forward Packet Loss	R_1	0, 2, 4, 6 (%)
Reverse Packet Loss	R_1	0, 2, 4, 6 (%)
Bottleneck Capacity {Forward,Reverse}	R_2	{6,0.64},{8,0.8},{30,10} (Mbps)

S_1, S_2, and S_3 are Gigabit Ethernet switches. R_1, a Linux router installed with Ubuntu 12.04 LTS, emulates different sets of network path quality with `tc`. The link capacity of R_1 is set to 1Gbps for emulating large capacity links in the Internet core, but it often times incurs a higher delay and packet loss. R_2 is a MikroTik 750G RouterBoard, emulating a home network by limiting the bottleneck link capacity. We choose three asymmetric capacity profiles which are commonly found in ADSL or VDSL users. Moreover, R_2 is configured to use a 50-packet FIFO queue. The network path is loaded with 20% of cross traffic generated by three Linux hosts (X_1, X_2, and X_3) using D-ITG [55]. The cross-traffic packets are UDP packets, being generated according to Pareto inter-arrivals with a shape parameter $\alpha = 1.9$ [56].

The client is installed with Ubuntu 10.10 with a Firefox browser and Flash player 11, while the web server is installed Ubuntu 12.04 LTS, Apache 2.2.22, and Adobe F4F Apache Module 4.5.1 for supporting Adobe HDS. The IRate middlebox is also a Linux server installed with the software bridge (`bridge-utils` and `brctl`) for bridging two network interfaces. In this paper, the direction of forward and reverse path are referred to the uplink and downlink of the server, respectively.

2) Preparing Training Data and the Decision Tree: To generate useful training data, we use a wide range of network path parameters to emulate different network connections (e.g., local access vs. access from another country). Table V summarizes the network parameters. We consider all the possible combinations of the parameters. That is, we have a total of 384 ($8 \times 4 \times 4 \times 3$) connection settings. For each setting, we repeatedly perform both IRate pre-stream measurement and TCP bulk download for three times. We then use these data to build a decision tree using C4.5.

For the IRate pre-stream measurement, we first let the testbed run for two seconds to allow the cross traffic to

reach a steady state. We then run the text-based probe-kit script to launch the pre-stream measurement for 60s. Three measurement flows are established between the client and the IRate middlebox. To mitigate possible effects from periodic events on the network path and the client, the second and the third measurement flows are initiated at a random time between [0.5, 1]s after the start of first and the second measurement flow, respectively. Furthermore, we employ asymmetric IP packet sizes of 1500Bytes and 120Bytes for measuring the forward and reverse path, respectively. This packet size combination can mitigate the network congestion caused by the reverse-path bottleneck, which has slight impact on the video streaming performance.

For the bulk download, we measure the TCP throughput by initiating an HTTP bulk download of a 4.3-MB file using `wget`, and `tcpdump` is run at the background to capture the traffic on the client side. The size of the file is approximately equal to a one-minute 500-Kbps video clip. Hence, the throughput of downloading that file is similar to that in streaming a short video clip. To speed up the experiments, the download tests lasted for at most 60s, which is long enough to leave the slow-start phase and capture the average TCP throughput.

We analyze the packet traces using `tshark` to extract the packet timestamps and compute the average throughput β. We then convert β to the BIBR using (2). We adopt the five video quality levels, denoted by l_i , $i = 0, 1, 2, 3, 4$, used in our subjective assessment described in §II-C. Their bitrates are 300, 500, 1000, 1700, and 2500Kbps, respectively. Two additional levels $i = -1, 5$ are introduced to label the cases having a throughput much lower than the lowest and higher than the highest bitrates. Therefore, the number of samples in each level is more evenly distributed, which can alleviate the data overfitting problem.

We use C4.5 classifier in Weka [57] to generate a decision tree with the inputs of the network path parameters and β. We set the confidence factor to 0.25 for tree pruning. To prevent outliers from increasing the height of the tree, we also require the number of records at each leaf to be at least 10% of the total number of test records. The resultant decision determines the BIBR mainly by the median RTT and forward packet loss rate.

Besides, measuring the asymmetric packet loss rate is very important for estimating the BIBR. As the forward loss rate has to be inspected in all the cases in the decision tree, while only four cases need the reverse loss rate. This is because the forward path is used for streaming the video data, but the reverse path is for TCP ACKs, which have only a slight impact on the throughput. If only round-trip loss rate is used, the BIBR will be under-estimated when the actual packet loss rates are asymmetric.

3) Accuracy of Estimating the BIBR: We next use the 60-second training data to evaluate IRate's accuracy of estimating the BIBR. We slice the 60-second measurement data to shorter measurement durations from the beginning of the measurement to the required duration for emulating a shorter time window for pre-stream measurement. If the predicted quality level is one of the two additional levels $i = \{-1, 5\}$, it is mapped to the quality levels $\{0, 4\}$, respectively. Fig. 11 shows the accuracy of IRate's prediction across the entire pre-stream time window, from 1s to 30s. We only show the accuracy up to 30s, because we find that the accuracy is converged after 30s. We compute $\Delta\mathbb{L} = \widehat{\mathbb{L}} - \mathbb{L}$, where $\mathbb{L}$ is the actual BIBR computed from the throughput measurement. The green (bottom), white (middle), and red (top) portions of the bars show the percentage of samples that IRate has under-, correctly-, and over-estimated the BIBR (i.e., $\Delta\mathbb{L} < 0$, $\Delta\mathbb{L} = 0$, and $\Delta\mathbb{L} > 0$), respectively.

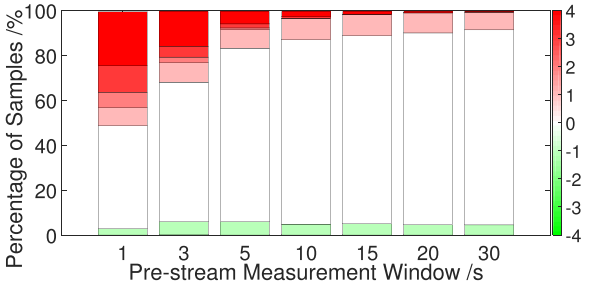


Fig. 11. The accuracy of IRate's estimation of the BIBRs using different pre-stream time windows.

The white portions indicating the correct estimation increase from 45.9% to 86.8% as the pre-stream time window grows from 1s to 30s, because a larger time window can allow the probe kit to obtain more results and mitigate the effect of short-term fluctuations. Although about 4.8% of the samples under-estimates the BIBR by one level, the video playback under these cases can still play smoothly without any rebuffering. By also considering these cases as acceptable, the accuracy is above 80% for a pre-stream time window of 10s.

4) Video Streaming Performance With IRate: This set of experiments is to illustrate the benefit of IRate by comparing the streaming performance against the usage of BIBR to HTTP streaming under an unknown network environment. We have integrated IRate into a small video streaming system and run it on the same testbed. The client in the testbed streams video clips from the server with a customized Flash video player, which is modified from the *Strobe Media Playback* for supporting the similar functions as *FlashTrack* [17] for both HTTP streaming and DASH. The player can report application layer information, such as rebuffering events, the number of bytes downloaded. Particularly for DASH, the BIBR chosen by IRate is only effective to the first video segment. After that, the same quality adaption algorithm will take over the bitrate adaption process.

To emulate a more realistic environment to evaluate the performance of IRate, we generated a set of 200 samples by randomly choosing an RTT from the range of [4,150]ms, forward and reverse packet loss rates from [0,6]%, and one of the bandwidth profile in Table V. For each set of metrics, the client ran the probe-kit script for 10s to determine the BIBR. Besides, the test video clip is a 60-second sports video, which is one of the test video clips used in the subjective assessment.

A Firefox browser is run on the client to load the video page and play the video using HTTP streaming with IRate,

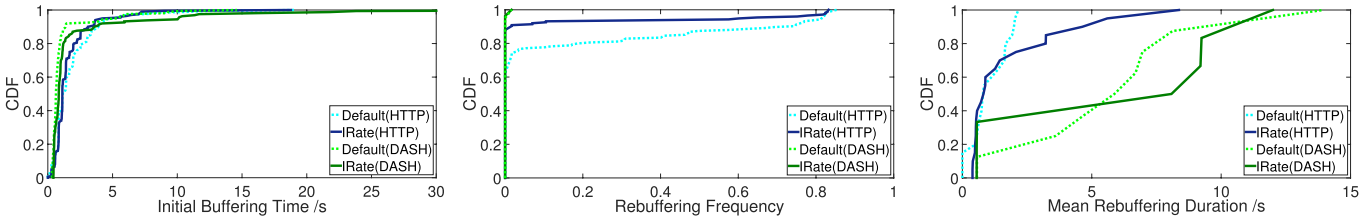


Fig. 12. The CDFs of application performance metrics. (a) Initial buffering time. (b) Rebuffering frequency. (c) Mean rebuffering duration.

HTTP streaming with predefined initial bitrate. We assume that the predefined initial bitrate scheme uses quality level l_2. To speed up the experiment, we only allow the video to be played for 90s. Moreover, HTTP streaming with predefined initial bitrate and DASH without IRate will not be tested if the predicted BIBR is l_2 or l_0, respectively, because the predicted BIBR is the same as the default value.

The accuracy for IRate's estimation of the BIBR under randomly selected network metrics is about 75%. Furthermore, we quantify the improvement in the video streaming performance by using IRate. We analyze the log captured by FlashTrack by using the application performance metrics proposed in [17]: Initial buffering time (T_{init}), rebuffering frequency (f_{rebuf}), and mean rebuffering duration (T_{rebuf}). Fig. 12(a), 12(b), and 12(c) plot the CDFs of T_{init}, f_{rebuf}, and T_{rebuf} for HTTP streaming and DASH, with and without IRate, respectively.

In our analysis, we exclude the cases that the estimated BIBR is equal to the predefined bitrate of the streaming method (i.e., l_2 for HTTP streaming and l_0 for DASH), because both methods are expected to have the same performance. There are {19%, 12.5%} of cases that have the estimated BIBR of $\{l_0, l_2\}$. Fig. 12(a) shows that IRate reduces the initial buffering time in HTTP streaming. About 80% of HTTP streaming with IRate can start playing the video within 2s, which is 6.3% higher than that of without IRate. The native DASH has a shorter T_{init}, because it always starts with the lowest quality. Although DASH needs more time for initial buffering when the predicted BIBR is used, the absolute time is still very short. T_{init} is less than 4s for 90% of the cases, but it also shows a long tail for over-estimated cases.

Rebuffering frequency quantifies how often rebuffering events occur during the video playback. The smaller the value means the playback is smoother, giving a better QoE [9], [17]. $f_{rebuf} = 0$ means no rebuffering throughout the playback. Fig. 12(b) shows that 88% of the HTTP streaming (with IRate) cases encounter no rebuffering events, which is 25.7% more than HTTP streaming (with predefined quality). On the other hand, DASH greatly reduces rebuffering events by adapting the video bitrate, and the performance for DASH with IRate and the native DASH is comparable.

We average the rebuffering duration of all the rebuffering events of each video playback. Fig. 12(c) plots the distributions of T_{rebuf} for those cases with $f_{rebuf} > 0$ in Fig. 12(b). By using IRate, the median of T_{rebuf} is similar in HTTP streaming. However, some cases in HTTP streaming show higher mean rebuffering duration than the default, which can be due to over-estimated BIBR. Similarly, the performance of DASH for both default and IRate is generally comparable. However, we hesitate to draw any general conclusion as the rebuffering events are rare in our dataset.

5) DASH Stability and Efficiency: DASH stability refers to how frequent the bitrate changes during the video playback. Previous studies showed that unstable bitrate can hurt the QoE [12]. We define a stability metric, denoted by τ, in (3) to quantify the improvement of DASH's bitrate stability by IRate. This metric is similar to the stability metric in [58]. However, we consider only the first seven video chunks (∼28s) of the video playback which are more relevant to the choice of initial bitrate.

$$\tau = \frac{\sum_{d=2}^{d\leq 7} |b(l_{d-1}) - b(l_d)|}{\sum_{d=1}^{d\leq 7} b(l_d)}, \tag{3}$$

where l_d is the bitrate level used for streaming d^{th} video chunk.

Fig. 13(a) shows the CDF of DASH stability for the predefined (lowest) quality strategy and IRate. Nearly half of the cases in IRate has no bitrate switching (i.e., $\tau = 0$) in the first seven video chunks. Since we do not show the cases with BIBR equals to l_0, all the cases in the native DASH switch the bitrate at the beginning (i.e., $\tau > 0$). This shows that selecting the BIBR helps improve the stability by reducing quality up-switching at the beginning of video.

DASH efficiency, denoted as ϵ, quantifies the effectiveness of DASH on utilizing the network resources for video streaming. In the ideal case, the BIBR is equal to the network throughput. In other words, the time to download a video chunk is equal to the length of video it contains. In [58], they proposed to use the video bitrate and the average throughput to compute the efficiency. However, these two metrics cannot accommodate the video encoded with the variable bitrate (VBR). Hence, we propose to use the video chunk download time and the length of video (in seconds) contained in a video chunk, denoted by g, to compute the efficiency:

$$\epsilon = \frac{\sum_{i=1}^{i\leq \gamma} \frac{\omega_i - g}{g}}{\gamma}, \tag{4}$$

where γ is the total number of downloaded video chunks, and ω_i is the time spent on downloading i^{th} video chunk.

If DASH is completely efficient, the download time of a video chunk will be equal to the number of seconds of video

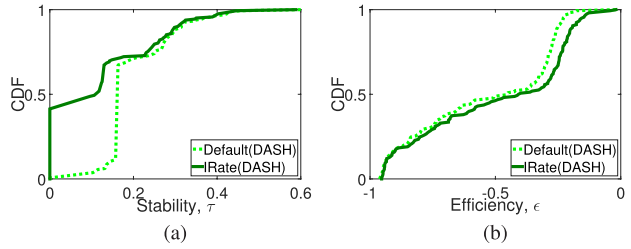


Fig. 13. The CDFs of DASH efficiency and stability metrics. (a) DASH stability. (b) DASH stability.

contained in the chunk, (i.e., $\epsilon = 0$). When DASH cannot completely utilize the network bandwidth, less time is used to download the same video chunk, resulting in $\epsilon < 0$. Fig. 13(b) shows the CDF of the efficiency metric. For $\epsilon \leq 0$, DASH with IRate obtains a closer-to-zero value, meaning a higher efficiency. The DASH with IRate has a median value 36% larger than the predefined quality case. This is because DASH with IRate can avoid ramping up from the lowest bitrate, therefore better utilizing the network bandwidth.

VI. User QoE Experiments

To further compare the performance in terms of QoE, we conduct subjective assessments similar to the one described in §II-C. The main difference is that the video quality levels in this set of experiments are adapted according to the real Internet performance, instead of an emulated transition scheme. The goal of this assessment is to compare the difference in QoE between the default (lowest) initial quality and the BIBR estimated by IRate.

In this assessment, we create the pre-stream time window by requiring the subjects to fill a short survey, because we do not have considerable amount of content for subjects to choose from to generate the user think time. The pre-stream measurement is conducted at the background to measure the network path quality between the clients and the IRate-enabled video server connected to our campus network until the completion of the survey. After the survey, each subject is required to watch two video clips randomly chosen from the four video clips used in §II-C. The approach (default or IRate) for selecting the initial bitrate in the video clip is randomized and unknown to the subjects. After that, the video player adapts the video quality according to the network throughput.

We have successfully collected the results from 22 volunteers. Instead of inviting them to the laboratory, we deliver the assessment site through email. Therefore, the subjects can stream the video clips using the real Internet networks. Table VI shows the network types used by the subjects according to the IP records. Most of the subjects access the experiment using local residential broadband network. One of the participants is from the US. The predicted BIBRs from IRate show that one of local subject's network quality cannot support the highest quality level. The player logs record only one rebuffering event in one of the playback started with the default approach. However, we cannot observe any rebuffering event for all video playbacks using IRate. Therefore, the accuracy of IRate can be considered as 100%.

TABLE VI
Distribution of Network Types

No. of subjects	Network type	Location
10	Residential broadband (HKBN)	Hong Kong
6	Residential broadband (PCCW)	Hong Kong
2	University dormitory WiFi	Hong Kong
2	University campus network	Hong Kong
1	Residential broadband (Hutchison)	Hong Kong
1	Overseas academic network	US

Similar to the study presented in §II-C, we compare the MOS, $\overline{E_{init}}$, and $\overline{E_{overall}}$ between the video clips started with the estimated BIBR and the predefined quality level. We find that the MOS is increased from 3.82 to 4.09 (6.67%) when the estimated BIBR from IRate is applied. Furthermore, IRate's average rating on the initial video quality $\overline{E_{init}}$ and the overall picture quality $\overline{E_{overall}}$ are higher by 24.4% and 11.8%, respectively. Moreover, the differences in rating for $\overline{E_{init}}$ and $\overline{E_{overall}}$ are significant (p <0.05) in two-sample t-test.

VII. Discussion

In this section, we discuss the limitations and issues of IRate.

A. Accessing Videos From Third-Party Video Sites

In some circumstances, video pages are accessed directly without first visiting the video website. Users could be redirected from 1) embedded video objects in third party websites, 2) sharing forums in social networks, or 3) recommendations in search engines. IRate can still be used for the first case. The embedded video objects provided by the video sites for sharing are implemented as an `iframe`, which then loads a small page from the video website. In this case, the probe-kit script can be inserted to the page and executed by users. For the second and third cases, pre-stream measurement cannot be carried out as the social networks or the search engines only provide a link to the video website. Therefore, the probe-kit script cannot be inserted. Without any reliable measurement results, IRate can simply fallback to the default bitrate scheme. A better solution requires the co-operation between these websites and the video service provider in that the probe-kit script can also be inserted into the web pages of these websites and executed when the web pages are rendered. Another solution is to perform the measurement during the pre-roll advertisement, but more cross traffic can be incurred by the download of the Ad stream.

B. Scalability and Security Issues of IRate

The server-side design may suffer from scalability issue for large-scale video websites. Our current IRate prototype measures every client accessing the video website. However, measuring all clients may not be necessary. As the network condition may be stable within a short time [59], [60] and the average throughput can be better known after the first video is watched, the web server could also utilize these historical data to improve the accuracy of the BIBR. Moreover, IRate can

reduce the measurement traffic by closing the measurement flows of some clients for which sufficient data have been collected for estimating their BIBRs. The IRate middlebox can send messages to the probe-kit script, so that the script will not establish any new measurement flow.

We believe that the IRate middlebox is not vulnerable to DDoS attacks, because the IRate middlebox is IP-less, therefore transparent to clients. Moreover, the middlebox only handles the TCP connections successfully established by web server. Malicious clients can inject specially crafted HTTP requests to be used by the IRate into a network flow and confuse the middlebox to mistaken the flow as a measurement flow. However, the middlebox can close the existing connections and refuse any new measurement connections from the clients whenever the middlebox collects enough data or maintains sufficient measurement flows to the same client.

C. Short vs. Long Video Clips

IRate is obviously most beneficial for short video clips, which are still very popular in today's Internet. On the other hand, for long videos or movies, the benefit to the overall QoE rating will be decreased as users may forget the experience at the beginning [10]. In a recent study, Staelens et al. [61] evaluated the QoE of long videos on tablets using Single Stimulus Continuous Quality Evaluation (SSCQE) [62], which continuously measures the QoE. Their results showed that large-range bitrate switchings at the beginning of the video, which can be mitigated by IRate, have significant impact to the QoE instantaneously. Further investigation on quantifying the effect of the video length will be our future work.

VIII. Related Works

Hsu [63] invented a system for determining the startup bitrate in adaptive bitrate streaming. His system stores the measured throughput in previous video streaming in the browser's cookie. At the next video playback, the startup bitrate is estimated by using the stored historical throughput. However, the startup bitrate cannot be determined at the first visit, or after the cookie is cleared. Netflix [51] and Google [52] archived the throughput data from video streaming clients by countries, ISPs, and access methods, but these data will not be used in deciding the initial video bitrate. Furthermore, more fundamentally, the historical data may not correctly reflect the latest network condition.

Liu et al. [64] proposed a coordinated video control plane to optimize the video delivery by a global view of clients and their network. However, the summarized global view may not be able to react to short-term performance degradation promptly for a small number of clients. Fardous and Kanhere [65] utilized the geographic location to estimate the bandwidth for mobile users' next locations, but the method is location-specific, and the training cost is expensive.

IX. Conclusion

In this paper, we presented a server-side pre-stream measurement system, IRate, which is compatible with existing video delivery infrastructure. IRate exploits the pre-stream time window to perform active measurement to the actual video cache. The performance data collected from IRate could help profile the client by estimating the best initial bitrate (BIBR) for HTTP streaming. Our testbed results showed that IRate can acquire enough path quality data for estimating the BIBR with 80% accuracy in 10s, and it can help improve the QoE of HTTP streaming. Our user experiment further validated that IRate can improve the QoE by more than 6% in the actual Internet environment.

In the future, we will investigate incorporating historical data to further improve the accuracy of IRate. It is common to record the throughput and video streaming performance in the video player. These collected data can validate the BIBR predicted by IRate. Similar to MLASH [66], the corresponding IRate measurement results can be used as a feedback to the model, such that the quality oracle can learn the characteristics of the clients and improve the prediction accuracy.

Acknowledgements

The authors thank the three anonymous reviewers for their valuable comments.

References

[1] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "YouTube everywhere: Impact of device and infrastructure synergies on user experience," in *Proc. ACM/USENIX IMC*, 2011, pp. 345–360.
[2] YouTube. *Video Quality*, accessed on Feb. 1, 2016. [Online]. Available: https://support.google.com/youtube/answer/91449?hl=en
[3] L. De Cicco, S. Mascolo, and V. Palmisano, "Feedback control for adaptive live video streaming," in *Proc. ACM MMSys*, 2011, pp. 145–156.
[4] Z. Li *et al.*, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 719–733, Apr. 2014.
[5] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic HTTP streaming," in *Proc. ACM CoNEXT*, 2012, pp. 109–120.
[6] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proc. ACM MMSys*, 2011, pp. 169–174.
[7] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, "QDASH: A QoE-aware DASH system," in *Proc. ACM MMSys*, 2012, pp. 11–22.
[8] 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 *Proc. ACM SIGCOMM*, 2014, pp. 187–198.
[9] F. Dobrian *et al.*, "Understanding the impact of video quality on user engagement," in *Proc. ACM SIGCOMM*, 2011, pp. 362–373.
[10] M. H. Pinson and S. Wolf, "Comparing subjective video quality testing methodologies," *Proc. SPIE*, vol. 5150, pp. 573–582, Jun. 2003.
[11] N. Cranley, P. Perry, and L. Murphy, "User perception of adapting video quality," *Int. J. Human-Comput. Stud.*, vol. 64, no. 8, pp. 637–647, 2006.
[12] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen, "Spatial flicker effect in video scaling," in *Proc. IEEE QoMEX*, Sep. 2011, pp. 55–60.
[13] V. K. Adhikari *et al.*, "Measurement study of Netflix, Hulu, and a tale of three CDNs," *IEEE/ACM Trans. Netw.*, vol. 23, no. 6, pp. 1984–1997, Dec. 2015.
[14] Netflix. *Manage Bandwidth Usage*, accessed on Mar. 24, 2014. [Online]. Available: https://support.netflix.com/en/node/87
[15] Mikkel. *Tips to Enhance Dailymotion HQ/HD Video Viewing*, accessed on Mar. 24, 2014. [Online]. Available: http://geekfp.com/tips-to-enhance-dailymotion-hqhd-video-viewing/
[16] V. K. Adhikari, S. Jain, Y. Chen, and Z.-L. Zhang, "Vivisecting YouTube: An active measurement study," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2521–2525.
[17] R. K. P. Mok, E. W. W. Chan, and R. K. C. Chang, "Measuring the quality of experience of HTTP video streaming," in *Proc. 12th IFIP/IEEE IM*, May 2011, pp. 485–492.
[18] M. Levkov, *Video encoding and transcoding recommendations for HTTP Dynamic Streaming on the Flash Platform*, Adobe, San Jose, CA, USA, Oct. 2010.

[19] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "YouTube traffic characterization: A view from the edge," in *Proc. 7th ACM/USENIX IMC*, 2007, pp. 15–28.

[20] X. Cheng, J. Liu, and C. Dale, "Understanding the characteristics of Internet short video sharing: A YouTube-based measurement study," *IEEE Trans. Multimedia*, vol. 15, no. 5, pp. 1184–1194, Aug. 2013.

[21] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs," in *Proc. ACM IMC*, 2012, pp. 211–224.

[22] T. Hoßfeld *et al.*, "Best practices for QoE crowdtesting: QoE assessment with crowdsourcing," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 541–558, Feb. 2014.

[23] B. Rainer and C. Timmerer, "A subjective evaluation using crowdsourcing of adaptive media playout utilizing audio-visual content features," in *Proc. IEEE/IFIP NOMS*, May 2014, pp. 1–7.

[24] *Subjective Audiovisual Quality Assessment Methods for Multimedia Applications*, document ITU-T Rec. P.911, Dec. 1998.

[25] R. K. P. Mok, W. Li, and R. K. C. Chang, "Detecting low-quality crowdtesting workers," in *Proc. IEEE/ACM IWQoS*, Jun. 2015, pp. 201–206.

[26] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Characterizing user sessions on YouTube," in *Proc. ACM/SPIE MMCN*, 2008.

[27] Interactive Advertising Bureau. (May 2008). *Digital Video In-Stream Ad Format Guidelines and Best Practices*. [Online]. Available: http://www.iab.net/media/file/IAB-Video-Ad-Format-Standards.pdf

[28] S. S. Krishnan and R. K. Sitaraman, "Understanding the effectiveness of video ads: A measurement study," in *Proc. ACM IMC*, 2013, pp. 149–162.

[29] YouTube. *YouTube TrueView Video Ads*, accessed on Jan. 2, 2012. [Online]. Available: http://www.youtube.com/advertise/trueview.html

[30] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," *IEEE/ACM Trans. Netw.*, vol. 11, no. 4, pp. 537–549, Aug. 2003.

[31] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient available bandwidth estimation for network paths," in *Proc. PAM*, 2003, pp. 1–11.

[32] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 6, pp. 879–894, Sep. 2003.

[33] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proc. 3rd ACM/USENIX IMC*, 2003, pp. 39–44.

[34] Ookla. *Speedtest*, accessedd on Sep. 1, 2015. [Online]. Available: http://www.speedtest.net/

[35] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzr: Illuminating the edge network," in *Proc. 10th ACM/USENIX IMC*, 2010, pp. 246–259.

[36] M. Kaplan, M. Zeljkovic, M. Claypool, and C. Wills, "How's my network? Predicting performance from within a Web browser sandbox," in *Proc. IEEE 37th Conf. LCN*, Oct. 2012, pp. 521–528.

[37] W. Li, R. K. P. Mok, R. K. C. Chang, and W. W. T. Fok, "Appraising the delay accuracy in browser-based network measurement," in *Proc. ACM IMC*, 2013, pp. 361–368.

[38] Endace. *DAG Packet Capture Cards*, accessed on May 3, 2013. [Online]. Available: http://www.endace.com

[39] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-based traffic shaping for stabilizing oscillating adaptive streaming players," in *Proc. ACM NOSSDAV*, 2013, pp. 19–24.

[40] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in *Proc. USENIX NSDI*, 2012, p. 24.

[41] E. W. W. Chan, A. Chen, X. Luo, R. K. P. Mok, W. Li, and R. K. C. Chang, "TRIO: Measuring asymmetric capacity with three minimum round-trip times," in *Proc. ACM CoNEXT*, 2011, Art. no. 15.

[42] Ookla. *Speedtest.net Mini*, accessed on Jun. 5, 2013. [Online]. Available: http://www.speedtest.net/mini.php

[43] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP reno performance: A simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 133–145, Apr. 2000.

[44] W. Bao, V. W. S. Wong, and V. C. M. Leung, "A model for steady state throughput of TCP CUBIC," in *Proc. IEEE GLOBECOM*, Dec. 2010, pp. 1–6.

[45] B. A. A. Nunes, K. Veenstra, W. Ballenthin, S. Lukin, and K. Obraczka, "A machine learning approach to end-to-end RTT estimation and its application to TCP," in *Proc. IEEE ICCCN*, Jul./Aug. 2011, pp. 1–6.

[46] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to TCP throughput prediction," *IEEE/ACM Trans. Netw.*, vol. 18, no. 4, pp. 1026–1039, Aug. 2010.

[47] I. El Khayat, P. Geurts, and G. Leduc, "Machine-learnt versus analytical models of TCP throughput," *Comput. Netw., Int. J. Comput. Telecommun. Netw.*, vol. 51, no. 10, pp. 2631–2644, 2006.

[48] M. Dhawan *et al.*, "Fathom: A browser-based network measurement platform," in *Proc. ACM/USENIX IMC*, 2012, pp. 73–86.

[49] X. Luo, E. W. W. Chan, and R. K. C. Chang, "Design and implementation of TCP data probes for reliable and metric-rich network path monitoring," in *Proc. USENIX ATC*, 2009, p. 4.

[50] L. De Cicco and S. Mascolo, "An experimental investigation of the Akamai adaptive video streaming," in *Proc. USAB*, 2010, pp. 447–464.

[51] Netflix. *The ISP Speed Index From Netflix*, accessed on Sep. 10, 2014. [Online]. Available: http://ispspeedindex.netflix.com/

[52] Google. *Google Video Quality Report*, accessed on Sep. 10, 2014. [Online]. Available: https://www.google.com/get/videoqualityreport/

[53] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann, 1993.

[54] P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Analysis and modelling of YouTube traffic," *Trans. Emerg. Telecommun. Technol.*, vol. 23, no. 4, pp. 360–377, 2012.

[55] A. Botta, A. Dainotti, and A. Pescapé, "A tool for the generation of realistic network workload for emerging networking scenarios," *Comput. Netw.*, vol. 56, no. 15, pp. 3531–3547, Oct. 2012.

[56] C. Dovrolis, P. Ramanathan, and D. Moore, "Packet-dispersion techniques and a capacity-estimation methodology," *IEEE/ACM Trans. Netw.*, vol. 12, no. 6, pp. 963–977, Dec. 2004.

[57] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.

[58] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE," in *Proc. ACM CoNEXT*, 2012, pp. 97–108.

[59] V. Paxson, "End-to-end routing behavior in the Internet," *IEEE/ACM Trans. Netw.*, vol. 5, no. 5, pp. 601–615, Oct. 1997.

[60] Y. Zhang, V. Paxson, and S. Shenker, "The stationarity of Internet path properties: Routing, loss, and throughput," AT&T Centre Internet Res. ICSI, Berkeley, CA, USA, Tech. Rep. ACIRI-May-2000, 2000.

[61] N. Staelens *et al.*, "Subjective quality assessment of longer duration video sequences delivered over HTTP adaptive streaming to tablet devices," *IEEE Trans. Broadcast.*, vol. 60, no. 4, pp. 707–714, Dec. 2014.

[62] *Methodology for Subjective Assessment Quality Television Pictures*, document ITU-R Rec. BT.500-12, Sep. 2009.

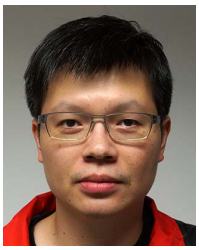
[63] J. K. Hsu, "Startup bitrate in adaptive bitrate streaming," U.S. Patent 8 516 144 B2, Aug. 20, 2013.

[64] X. Liu *et al.*, "A case for a coordinated Internet video control plane," in *Proc. ACM SIGCOMM*, 2012, pp. 359–370.

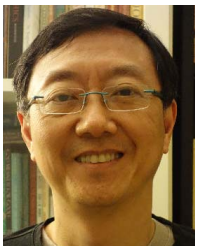
[65] J. Fardous and S. S. Kanhere, "On the use of location window in geo-intelligent HTTP adaptive video streaming," in *Proc. IEEE ICON*, Dec. 2012, pp. 46–51.

[66] Y.-L. Chien, K. C.-J. Lin, and M.-S. Chen, "Machine learning based rate adaptation with elastic feature selection for HTTP-based streaming," in *Proc. IEEE ICME*, Jun./Jul. 2015, pp. 1–6.

Ricky K. P. Mok received the B.Eng. degree in computer engineering from The Chinese University of Hong Kong, in 2009. He is currently pursuing the Ph.D. degree with the Department of Computing, The Hong Kong Polytechnic University, and is a Team Member of the Internet Infrastructure and Security Research Laboratory. His research interests include QoE evaluation of video services, HTTP video streaming systems, networked embedded systems, and network path quality measurement.



Weichao Li received the B.Sc. and M.Eng. degrees from the South China University of Technology, in 2002 and 2009, respectively. He is currently pursuing the Ph.D. degree with The Hong Kong Polytechnic University and is a Team Member of the Internet Infrastructure and Security Research Laboratory. His research interests include network measurement, and network operations and management. His current research focuses on accuracy of end-to-end network path quality measurement, mobile network measurement, and hardware-assisted network measurement.



Rocky K. C. Chang received the bachelor's degree from Virginia Tech and Vincennes University, and the M.Sc. and Ph.D. degrees from the Rensselaer Polytechnic Institute. He received his post-doctoral training with the Computer Science Department, IBM Thomas J. Watson Research Center. He is currently serving as an Associate Professor with the Department of Computing, The Hong Kong Polytechnic University (PolyU).

He is leading an Internet Infrastructure and Security Research Laboratory, addressing network measurement, network security, and network operations and management problems. He also leads a Research Group on Internet Services Monitoring and Diagnostics with the Division of Smart Cities under PolyU's Research Institute for Sustainable Urban Development.