EmuStream - An End-to-End Platform for Streaming Video Performance Measurement

GUANGHUI ZHANG, RUDOLF K. H. NGAN, and JACK Y. B. LEE, (Senior Member, IEEE)
Department of Information Engineering, the Chinese University of Hong Kong, Shatin, NT, Hong Kong

Corresponding author: Jack Y. B. Lee (e-mail: jacklee@computer.org).

This work was supported in part by a Direct Grant from the Research Committee of the Chinese University of Hong Kong.

ABSTRACT *Cord-cutting* has spread like wildfire as streaming video become commonplace in the Internet. This motivated intensive research in adaptive video streaming to improve its quality-of-experience (QoE) in the presence of network quality variations. Much of the existing research either employed dummy video contents or open source videos for QoE evaluation which may not capture the full spectrum of characteristics of real-world contents. This work fills this gap by developing a novel *EmuStream* platform based on real-world contents to enable realistic experiments and evaluation of *any* adaptive streaming algorithm in *any* streaming platform. First, EmuStream offers the largest (700+ titles) publicly available video bitrate trace dataset derived from real-world video contents. Second, we developed a mathematical model based on the dataset which can generate bitrate trace data for *arbitrary* target bitrates *without* actual video encoding. Third, we developed a novel Virtual Video Generator (VVG) which can generate h.264/h.265-compliant virtual videos that share the same frame/segment sizes as the source videos to enable experiments mimicking the streaming of commercial video contents. Last but not least, we developed a novel Streaming Performance Meter (SPM) that can measure detailed frame-by-frame playback performance from a video recording of the streaming playback of a virtual video. By decoding the virtual video's specially-coded frames, SPM can determine the playback timing and bitrate selected for each frame for use in computing any desired QoE metric. This paper presents the EmuStream platform and validates the trace data estimation model and VVG/SPM tools via controlled experiments.

INDEX TERMS Adaptive Video Streaming, DASH, Dataset, Quality-of-Experience.

I. INTRODUCTION

Cutting-the-cord is now a major trend where consumers replace their cable and satellite TV/Video-on-Demand (VoD) services by online streaming services from Netflix, Amazon, Hulu, Apple TV+, and so on. This is clearly an unstoppable trend but as consumers are paying for the services, they will have higher quality expectations.

Almost all streaming services today employ some form of online bitrate adaptation where the video bitrate is dynamically adjusted in response to the changing network conditions to improve quality-of-experience (QoE). To this end, intense researches have been conducted in recent years to design better and better bitrate adaptation algorithms. Interested readers are referred to [1] for an overview of adaptive streaming and [2-5] for a survey of adaptive streaming algorithms.

Unlike user-generated videos, premium streaming services offer contents such as TV series, movies, music concerts, and documentaries, where viewers have significantly higher expectation for picture quality as well as streaming performance. However, much of the existing research on video streaming relied on the use of either constant-bitrate-encoded (CBR) video or a few open-source variable-bitrate-encoded (VBR) videos, e.g., Big Buck Bunny [6] in experiments and performance evaluations [7-15]. While these approaches can offer a mean to compare the performance of different streaming algorithms, they may not be able to provide a more complete picture of the algorithms' actual performance when streaming a wide spectrum of real-world video contents.

This work fills this gap by developing a novel end-to-end platform *EmuStream* based on characteristics of real-world commercial video contents to enable realistic experiments and performance evaluation of *any* adaptive streaming algorithm in *any* streaming platform.

TABLE I
SUMMARY OF RELATED WORKS

Dataset	Video type	Codec	# of videos	# of bitrates	Bitrate range (Mbps)	Type of video content
Lederer *et al.* [7]	HD	h.264	6	20	0.05~8	open source
Lederer *et al.* [8]	HD	h.264	1	17	0.1~6	open source
Le Feuvre *et al.* [9]	UHD	h.265	1	13	1.8~18	open source and self-generated
Kreuzberger *et al.* [10]	HD (SVC)	h.264	12	12	0.6~10.4	open source and self-generated
Quinlan *et al.* [11]	HD	h.264, h.265	23	10	0.2~4.3	open source
Quinlan *et al.* [12]	UHD	h.264, h.265	3	13	0.2~40	open source
Zabrovskiy *et al.* [13]	UHD	h.264, h.265, VP9, AV1	10	19	0.1~20	open source
Bampis *et al.* [14]	HD	h.264	15	11	0.15~6	open source and self-generated
Duanmu *et al.* [15]	HD	h.264	20	11	0.2~7	open source and self-generated
This work	HD, UHD	h.264, h.265	722	13~16	0.2~50	commercial videos

This work has four contributions. First, we collected the video bitrate trace data of a wide spectrum of real-world commercial video contents. At the time of writing, the dataset includes bitrate trace data for 687 High Definition (HD) video titles from Bluray discs and 35 Ultra High Definition (UHD) video titles from UHD Bluray discs. Each set of trace data includes segment/frame size data for the entire video in multiple target bitrates (ranging from 200kbps to 50Mbps) as well as two codecs (h.264 and h.265). These trace data can be readily used in trace-driven simulations.

Second, we developed a mathematical model to estimate the segment/frame size trace data for *any* target bitrates based on the known segment/frame size trace data of existing available bitrates, thereby eliminating the need to perform the actual video encoding. This enables researchers to generate any desired set of target bitrates for specific experiments as well as to explore new bitrate allocation schemes.

Third, we developed a novel Virtual Video Generator (VVG) which can generate a *virtual* video stream from a video bitrate trace. The virtual video has the *same* frame sizes as the source video but *without* any of the source video's visual content. Instead, the virtual video comprises specially-coded video frames, each with a barcode encoding its frame number and bitrate version. Consequently, streaming these virtual videos have the *same* network resource requirements as streaming the source video, resulting in nearly the same streaming performances. This enables researchers to conduct experiments to evaluate an adaptive streaming algorithm's performance as if it is streaming actual real-world contents.

Fourth, we developed a novel Streaming Performance Meter (SPM) to measure detailed frame-by-frame playback performance from a *video recording* of the streaming playback of a virtual video generated by VVG. By decoding the virtual video's specially-coded frames, SPM can measure the playback timing and bitrate selected for each frame for computing the desired QoE metric.

Unlike existing approaches where the video players must be modified to capture streaming performance data [7,8,11,12,15], the proposed EmuStream platform can measure the streaming performance of *any* playback device, including devices where player modifications are not practical (e.g., proprietary players such as those in smartphones, smartTVs, and STBs).

The rest of the paper is organized as follows. Section II reviews the background and some previous related works; Section III presents details of the HD/UHD dataset, its statistical properties, the segment/frame size estimation model and its application to trace-driven simulation; Section IV presents the streaming performance measurement tools, i.e., the Virtual Video Generator (VVG) and the Streaming Performance Meter (SPM); Section V reports experimental results to validate the accuracy of the EmuStream platform; and Section VI summarizes the study and outlines some future work.

II. BACKGROUND AND RELATED WORK

Video streaming has gone through two fundamental changes in recent years. First, unlike early video streaming systems [16-17] which deliver video data at controlled transmission rate to the receiver over UDP, current streaming systems in the Internet have almost all switched over to transporting video data over HTTP/TCP [1-5]. This is primarily due to network compatibility considerations as UDP-based streaming protocols such as RTP are sometimes blocked or throttled by ISPs [5].

As a result, today's video servers are often implemented using HTTP servers and data are simply transported using HTTP over TCP, at a rate determined by TCP's flow and congestion control algorithms. In addition, as TCP is a reliable transport, any lost packets will be retransmitted automatically so that video degradation due to packet losses [18] have become a thing of the past. Instead, poor network conditions now result in degraded TCP throughput which eventually leads to client buffer underflow. In this case the client player will suspend playback until sufficient video data are received to resume it – also known as *rebuffering*.

Second, as TCP does not allow the application to control the transmission rate directly, HTTP-based streaming systems almost all employ adaptive video streaming. The principle is to divide a video into fixed-duration segments (e.g., each a few seconds) and then encodes them into a range of multiple target bitrates to cater to various network conditions. The client implements a bitrate adaptation algorithm which monitors the network condition (e.g., via measured throughput and buffer occupancy) and then dynamically selects the best bitrate version for downloading future video segments. This not only enables the client to stream video across a wide range of networks with different bandwidths, but also enables it to adapt to short-term bandwidth fluctuations that are common in wireless and mobile networks.

The above led to the Dynamic Adaptive Streaming over HTTP standard (or simply known as DASH [1]) developed by the MPEG standard committee. To support research in DASH systems, researchers have developed video datasets for use in DASH simulations and experiments. Table I summarizes the existing video datasets in the literature that are designed for DASH.

The first DASH dataset was released by Lederer *et al.* [7] in 2012. It offers 6 HD open-source videos in various genres (i.e., animation, action, and so on) encoded into 20 bitrate versions using h.264 [19]. The dataset also provides frame-level PSNR data for some of the bitrate versions to facilitate visual quality evaluations. In a subsequent work, Lederer *et al.* [8] further proposed a distributed DASH dataset across multiple locations. Utilizing multiple BaseURL elements within the media presentation description (MPD) [1] it enables researchers to experiment with content distribution network (CDN) locations and to explore dynamic switching across multiple CDNs.

Le Feuvre *et al.* [9] provided the first DASH dataset for Ultra High Definition (UHD) videos (i.e., resolutions up to 3840×2160) and is the first dataset to encode videos using h.265 [20]. Kreuzberger *et al.* [10] provided the first DASH dataset encoded using Scalable Video Coding (SVC). It offers 12 videos at varying bitrates and spatial resolutions.

Quinlan *et al.* [11] released a dataset comprising 23 open-source videos encoded in both h.264 and h.265 to enable direct comparison of the two video codecs. Their dataset also included a segment size distribution for 10 Bluray videos. However, as the *exact* segment size is not available, the bitrate trace of the source videos cannot be recreated. In a subsequent work, Quinlan *et al.* [12] added UHD video contents to their dataset, covering a wide range of bitrates from 235 kbps to 40 Mbps.

Zabrovskiy *et al.* [13] provided a multi-codec DASH dataset comprising h.264, h.265, VP9 [21] and AV1 [22] to enable interoperability testing and research in adaptation strategies for DASH clients with multiple video codecs. In another two studies by Bampis *et al.* [14] and Duanmu *et al.* [15], the authors conducted subjective QoE research with human viewers using their own DASH datasets.

Current DASH datasets have gone a long way towards supporting research in adaptive video streaming. Nevertheless, there remains three open problems. First, although existing datasets provided videos encoded with a wide range of resolutions across multiple bitrates, the bitrate profiles used are generally fixed. To conduct experiments with different bitrate profiles (e.g., Apple [23], Adobe [24], Microsoft [25]) one will need to re-encode the videos which is possible only if access to source video data is available.

TABLE II
NUMBER OF VIDEOS BY GENRES IN THE EMUSTREAM DATASET

Genre	Action	Romance	Horror	SCI-FI	Animation	Sport
Number	64	55	54	41	39	38

Second, existing datasets are often based on open source video contents (e.g., [7-15]) or self-generated video contents (e.g., [9,10,14,15]). There is a lack of real-world commercial video contents, especially those from the entertainment industry such as movies, documentaries, concerts, and TV shows, which constitute a significant portion of contents delivered by today's streaming services. Understandably this is due to copyright restrictions but it nonetheless severely handicaps research in the area.

Third, having a high-quality DASH dataset solves only half the problem. The other half of the problem is to apply such dataset in the target video streaming platform to evaluate its performance. Existing works either rely on simulation (e.g., [14,15]) or modified video players (e.g., [7,8,11,12,15]) to enable measurement of streaming performance. The former approach does not allow performance evaluation over real network environments and streaming players, and the latter approach is limited to players that can be modified. It is currently not possible to evaluate or compare performance against proprietary players such as those inside smartTVs and STBs. In the following sections we present the EmuStream platform designed to address these challenges.

III. EMUSTREAM DATASET

In this section we present details of the EmuStream dataset, some of its statistical properties, the segment/frame size estimation model, and then discuss its application to trace-driven DASH simulation.

A. DATASET OVERVIEW

At the time of writing, the dataset includes segment/frame size trace data for 687 HD video titles from Bluray discs and 35 UHD video titles from UHD Bluray discs. Table II lists the number of video titles across common genres. We will continue to expand the dataset, especially UHD video titles, in the future. It is worth emphasizing that the EmuStream dataset contains only segment/frame size trace data in the form of comma-delimited text files. It does *not* contain *any* actual video data from the source videos and thus can be released to the research community under a Creative Commons license [26].

The HD/UHD dataset developed in this work is based on video contents from the entertainment industry such that performance of streaming systems can be evaluated over a broad spectrum of real-world video contents. However, commercial video contents such as those from Bluray releases were *not* intended for streaming but for physical medium. Consequently, the source video (i.e., original video data extracted from HD/UHD Bluray discs) is not optimized for streaming and thus the bitrate trace data may not be suitable for use in streaming performance evaluation either.

Therefore, consistent with industry practice, we re-encoded each video title into multiple bitrates in the DASH format [1] using FFmpeg [27] (using Intel Quick Sync Video [28]) in both h.264 and h.265 according to Apple's bitrate profile [23] augmented by 4 additional bitrates up to 20Mbps for HD video and 7 additional bitrates up to 50 Mbps for UHD video. Table III summarizes the encoding parameters adopted. Note also that three segment durations (2s, 4s, 10s) were encoded for each video title. These re-encoded videos are then analyzed to extract the frame-level bitrate trace data. Together, a total of 56,946 bitrate traces are available in the EmuStream dataset.

There are two versions of video bitrate traces, one for segment-level data and the other for frame-level data. Both are stored in a comma-delimited text file format where each line stores the data for one segment/frame in the following formats: {*segment index*, *timestamp (s)*, *size (bytes)*, *target bitrate (kbps)*} for segment-level trace data and {*frame index*, *timestamp (s)*, *size (bytes)*, *target bitrate (kbps), frame type*} for frame-level trace data (see Fig. 1 for an example). These segment/frame trace data can then be used as inputs to trace-driven simulations to evaluate adaptive streaming algorithms (c.f. Section III-D).

B. DATASET STATISTICAL PROPERTIES

In this section we present some statistical properties of the dataset. We first present statistical properties for the source HD/UHD videos, i.e., original encoded videos extracted from HD/UHD Bluray discs. Fig. 2 plots the probability density function (PDF) of the mean bitrate (averaged across the whole video) for all 722 video titles. Not surprisingly, UHD video titles were encoded at a much higher bitrate than HD videos, i.e., 56.8 Mbps vs 24.3 Mbps. The lowest and highest mean bitrates in the dataset are 11.3 Mbps and 45.0 Mbps for HD titles and 34.9 Mbps and 77.9 Mbps for UHD titles. Fig. 3 plots the PDF for the coefficient of variation (CoV) of the source videos' bitrates. The CoV was computed from 2-second video bitrate averages. We observe that UHD videos exhibited slightly higher CoV compared to HD videos, with an overall average of 0.22 for HD and 0.29 for UHD titles.

Next we compare in Fig. 4 the bitrate characteristics of a source HD video from Bluray disc to its re-encoded versions at 5, 12, and 20 Mbps using h.264 with segment duration of 2s. We observe that the source video exhibited far more substantial bitrate variations than the re-encoded versions. Such large bitrate variations will render streaming difficult and this is one reason why no existing streaming service offered direct streaming of Bluray or UHD Bluray contents. In contrast, the re-encoded versions have significantly more consistent bitrates. There are still some variations due to the nature of video encoding and variation in the video contents but the extents are far smaller.

TABLE III
VIDEO ENCODING PARAMETERS

Parameter	Value
Target bitrates for HD video (Mbps)	0.2, 0.4, 0.8, 1.2, 2.2, 3.3, 5.0, 6.5, 8.6, 10.0, 12.0, 16.0, 20.0
Target bitrates for UHD video (Mbps)	0.2, 0.4, 0.8, 1.2, 2.2, 3.3, 5.0, 6.5, 8.6, 10.0, 12.0, 16.0, 20.0, 30.0, 40.0, 50.0
Resolution (corresponding bitrate in Mbps)	640×360 (0.2), 768×432 (0.4), 960×540 (0.8, 1.2), 1280×720 (2.2, 3.3), 1920×1080 (5.0, 6.5), 2560×1440 (8.6), 3840×2160 (≥10)
Codec	h.264 and h.265 (Intel Quick Sync Video [28])
Segment duration (s)	2, 4, 10
Frame rate	24 fps

```
frame index, timestamp (s), size (bytes), target bitrate (kbps), frame type
   3091,      128.79166,     46442,          8600,                 P
   3092,      128.83333,     69898,          8600,                 B
   3093,      128.87500,     41891,          8600,                 P
   3094,      128.91666,     56941,          8600,                 B
   3095,      128.95833,     43307,          8600,                 P
   3096,      129.00000,     62904,          8600,                 P
   3097,      129.04166,     62903,          8600,                 P
```

FIGURE 1. **An example of frame-level trace data.**

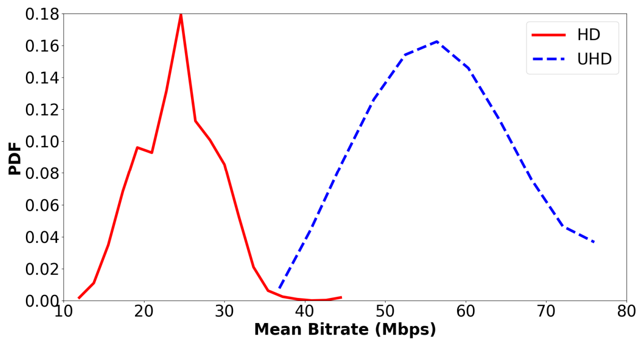


FIGURE 2. **Probability density function of mean bitrate of source HD/UHD Bluray videos.**

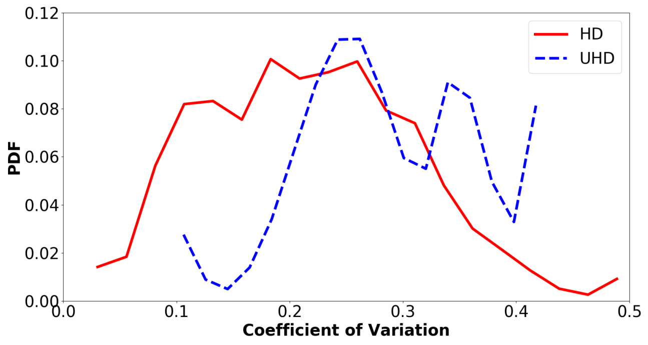


FIGURE 3. **Probability density function of bitrate CoV of source HD/UHD Bluray videos.**

As the re-encoded videos were all divided into segments, bitrate variations will manifest as segment size variations. Table IV compares the mean segment size CoV for segment durations of 2s, 4s, and 10s across three codecs. As expected, the CoV decreases with increases in segment duration as longer segment tends to average out short-term bitrate variations. However, it is a tradeoff as longer segment will also increase the startup delay (most players buffer a few segments before starting playback) as well as increase the cycle time of bitrate adaptations. Therefore, the service provider must strike a balanced between these conflicting objectives.

Another interesting finding is that software-based h.264 (using the x.264 library [29]) exhibited significantly higher segment size CoV than hardware-based h.264 (using Intel Quick Sync Video [28]) despite the fact that both were executed using the same parameters in FFmpeg. This suggested that even the choice of codec could have a significant impact on streaming performance. This also raises the question of whether one can further improve performance by a joint-design of the codec and adaptation algorithm – a worthwhile subject for future research.

C. SEGMENT/FRAME SIZE ESTIMATION MODEL

In addition to Apple's bitrate profile [23] there are also other bitrate profiles in use, e.g., Adobe [24], Microsoft [25], or custom ones. To evaluate a streaming system with such profiles means that one needs to re-encode the source video according to the target profile. Unfortunately this requires access to the source video data which cannot be redistributed due to copyright restrictions. Obviously, it is not possible for us to pre-generate trace data for all possible bitrate profiles either.

To address this challenge we propose a mathematical model to estimate the size of video segments/frames for an *arbitrary* unknown target bitrate, based on known segment/frame sizes from existing available bitrate versions. The principle is to employ kth-order polynomial regression to model the relation between segment size and target bitrate for *each* segment so that the segment/frame size for the unknown target bitrate can be estimated.

Specifically, let N be the number of available bitrate versions for a video title; v_i, i=0, 1, …, N-1, be the target bitrate for bitrate version i, and $b_{i,j}$, j=0, 1, …, M-1, be the actual segment size for segment j of bitrate version i. For each segment j we compute the set of coefficients, denoted by $\mathbb{C}_j=\{c_{j,0},c_{j,1},\cdots,c_{j,k}\}$, for the kth-order polynomial function

$$F(v_i,\mathbb{C}_j)=\sum_{l=0}^{k}c_{j,l}v_i^l \tag{1}$$

that minimizes the square of errors in estimating the size of segment j:

$$\begin{aligned}&\min_{\{\mathbb{C}_j\}}\sum_{\forall i}\left(b_{i,j}-\hat{b}_{i,j}\right)^2\\ &\text{where } \hat{b}_{i,j}=F(v_i,\mathbb{C}_j)\\ &\quad i=0,\ 1,\ \ldots,N-1\\ &\quad j=0,\ 1,\ \ldots,M-1\end{aligned} \tag{2}$$

TABLE IV
SEGMENT SIZE CoV ACROSS DIFFERENT SEGMENT DURATIONS AND VIDEO CODECS

Codec	Segment duration (s)		
	2	4	10
h.264(Software)	0.47	0.44	0.39
h.264(Hardware)	0.15	0.13	0.11
h.265(Hardware)	0.20	0.16	0.12

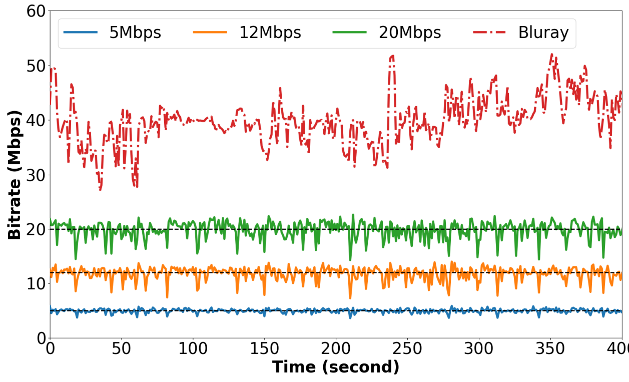


FIGURE 4. Bitrate variation over time for source Bluray video and its re-encoded versions.

With the optimized *per-segment* model we can then estimate the segment size $\hat{b}_{s,j}$ for any given target bitrate v_s from

$$\hat{b}_{s,j}=F(v_s,\mathbb{C}_j),\ v_0<v_s<v_{N-1} \tag{3}$$

without the need to re-encode the video. As will be shown in Section V this model is reasonably accurate (mean error of 1.17%). Based on $\hat{b}_{s,j}$, we can then further estimate the frame size $\hat{f}_{s,j,p}$ for the given target bitrate v_s from

$$\hat{f}_{s,j,p}=f_{m,j,p}\times\hat{b}_{s,j}\big/b_{m,j} \tag{4}$$

where $f_{m,j,p}$ and $b_{m,j}$ are the actual size of frame p in segment j and actual size of segment j of the available bitrate version m closest to the target bitrate v_s.

We note that segment size data are usually sufficient for measuring streaming performance for on-demand streaming services as most DASH players process and playback video in units of segments [2-5]. The estimated frame size will be useful for research in frame-based algorithms such as CMAF [30] in live video streaming [44].

D. APPLICATION TO TRACE-DRIVEN SIMULATION

The segment/frame size trace data can be directly used in trace-driven simulations. We illustrate it in this section by conducting simulated DASH streaming using four existing adaptive streaming algorithms: Stagefright [31], Robust-MPC [32] (henceforth called 'MPC'), Pensieve [33], and EAS-GP [34]. Stagefright is a common player implemented by Android smartphones; the remaining three are recently developed state-of-the-art adaptive streaming algorithms.

Fig. 5 shows the simulated network topology. The bottleneck link (i.e., mobile network) was simulated using TCP throughput trace data captured using a stationary mobile device connected to a production 3G mobile network [35]. Interested readers are referred to Liu and Lee [36] for more details on the trace data capture platform and to Zhang and Lee [34] for details on the simulation methodology. Streaming performance is measured using the QoE function proposed by Yin *et al.* [32]:

$$Q = \frac{1}{K}\left(\sum_{k=1}^{K} r_k - \lambda \sum_{k=1}^{K-1} \left| r_{k+1} - r_k \right| - \mu Z_p - \mu_\theta Z_\theta \right) \tag{5}$$

where Z_p is the total rebuffering duration – defined as the total time at which playback is suspended due to client buffer underflow, Z_θ is the startup delay, r_k is the bitrate selected for segment k, K is the total number of segments and the component weights follow the *Balanced* setting [32], i.e., $\lambda = 1$ and $\mu = \mu_\theta = 3000$. The choice of QoE function here is for illustration only and one can utilize any desired QoE function (e.g., [37-40]) in the simulations.

Fig. 6 compares the four streaming algorithms' QoE performance in streaming 60 different video titles from the EmuStream dataset. Stagefright achieved lowest QoE performance as it is, by-design, a conservative algorithm that tends to underutilize available bandwidth. In comparison, the three recently-proposed algorithms achieved much higher QoE performance.

More interestingly, both Pensieve and EAS-GP exhibited substantial QoE performance variations across the 60 video titles. For EAS-GP this is because it was designed for and trained using CBR video so the unexpected bitrate variations in the VBR trace data resulted in the QoE variations. By contrast, Pensieve was designed for VBR video streaming. Consistent with its original study [33], we trained Pensieve using segment size trace data from a single video title (video #22). This raises an interesting question of whether the observed QoE variations could be due to the choice of the training video.

To test this idea we trained *two* different Pensieve algorithms (named Pensieve-1 and Pensieve-2) using two different video titles as inputs and then compare their streaming performance across all 722 video titles in Fig. 7. To ease comparison, the video index in Fig. 7 were sorted in increasing order of the QoE differences between the two Pensieve algorithms.

There are two observations. First, the choice of training video indeed has substantial impact to Pensieve's performance. More importantly, neither one of them consistently outperformed the other. This strongly suggests that using a single video title for training [33] may result in an over-specialized algorithm.

Second, the QoE gap – defined as the QoE difference between Pensieve-1 and Pensieve-2 for the same video title, can vary from a low of 0 to a high of 0.89. This strongly suggests that it is necessary to test an algorithm over a much larger dataset than was commonly done in the literature in order to provide a more complete picture of an algorithm's performance in delivering real-world contents.

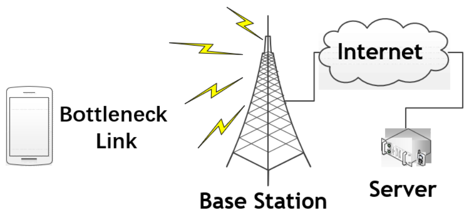


FIGURE 5. Simulated network topology.

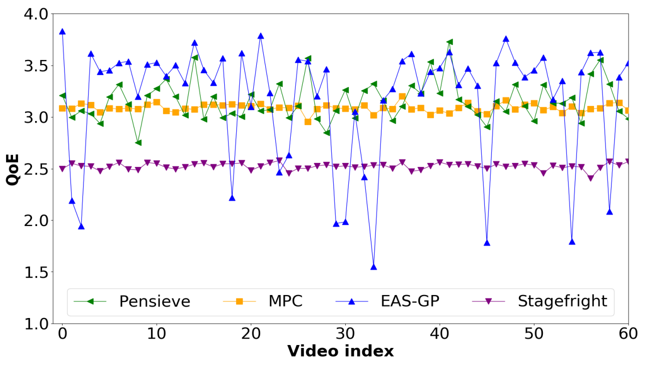


FIGURE 6. QoE performance across different video titles.

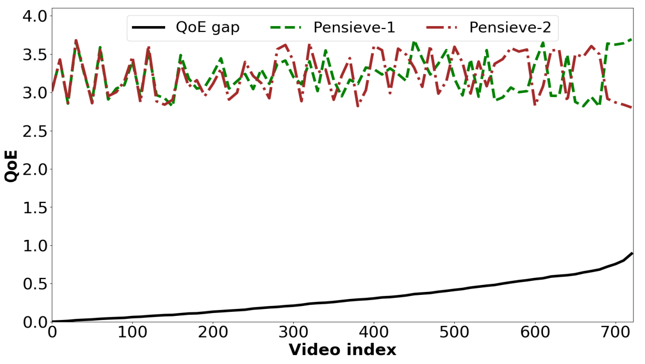


FIGURE 7. QoE differences between two Pensieve models trained using different videos.

The above experiment raises the important question of what (and how many) videos (or video bitrate characteristics) constitute good candidates as inputs for training machine-learning-based adaptive streaming algorithms. To our knowledge, this is uncharted territory but the availability of the EmuStream dataset opens up the path for future research in this direction.

IV. STREAMING PERFORMANCE MEASUREMENT

In addition to simulation, researchers often need to conduct experiments to validate the performance of a streaming system, or to investigate its behavior in real-world networks. One approach is to emulate video delivery by sending dummy data to a custom player which then performs accounting of the data received as well as (simulated) playback timings. However, as the data are not real, it does not account for actual decoding and playback behavior (e.g., during rebuffering). In addition, this approach can only be applied to custom video players, which precludes comparison to other existing players (i.e., most commercial video players deployed in the industry).

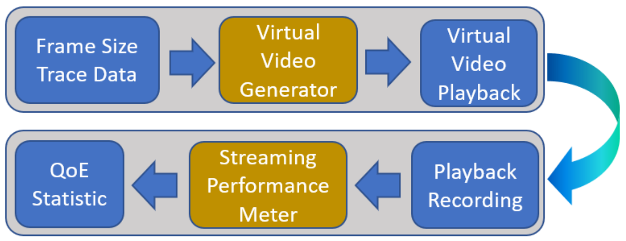


FIGURE 8. The data flow of QoE measurement with EmuStream platform.

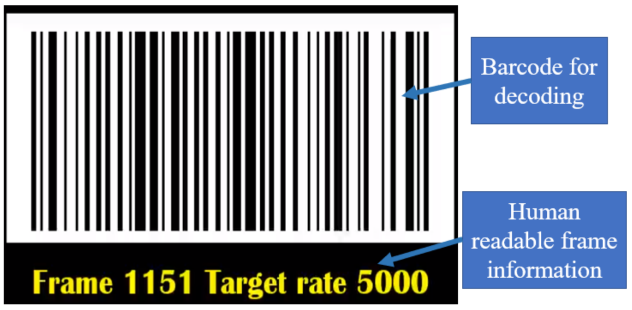


FIGURE 9. A video frame in a virtual video generated by VVG.

To improve the fidelity of experiments, one needs to stream real video data encoded using the target encoders at the desired set of bitrate profiles. Up to now this is only possible with open source videos such as Big Buck Bunny [6] or self-generated video contents [9,10,14,15]. These limited datasets clearly cannot capture the wide spectrum of video contents delivered by the industry and yet as demonstrated in Section III-D, streaming algorithms' performance can and do vary substantially across different video contents.

We tackle this challenge in this section by developing two new tools for the EmuStream platform: Virtual Video Generator (VVG) and Streaming Performance Meter (SPM). They enable controlled experiments for offline QoE evaluation (as opposed to real-time QoE monitoring in production environments [41]) for any existing streaming platforms, even proprietary ones.

A. VIRTUAL VIDEO GENERATOR

Fig. 8 illustrates the data flow of QoE measurement in the EmuStream platform. The first tool is a novel Virtual Video Generator (VVG) designed for generating a *virtual video stream* that shares the same frame sizes as given in a frame-level trace file. The virtual video stream is a standard-compliant h.264/h.265 encoded video that can be played back normally by any h.264/h.265-compatible video players. Instead of visual content from the source video, each frame in the generated virtual video comprises two visual components as depicted in Fig. 9.

First, a barcode at the center of each frame which encodes its frame index (an integer) and target bitrate (integer in units of kbps). Second, a human-readable text line at the bottom of each frame showing the same information. The barcode is designed for decoding by the SPM tool to extract streaming performance data automatically (see Section IV-B).

To generate a virtual video stream with encoded frame sizes matching the ones in the input frame size trace file presents a challenge, as the virtual video has completely different visual content than the source video. Moreover the encoded frame size heavily depends on the actual video content in the frame as well as the frame type (i.e., I, P, or B) [42] and thus cannot be controlled precisely.

To tackle this challenge, we modified the h.264/h.265 encoders in FFmpeg to first generate the raw virtual video frames in RGB format and then encodes them at a low target bitrate (e.g., 50 kbps) to generate encoded frames of sizes smaller than the required target frame sizes.

Next, the modified encoder adds *padding* data units to each frame until it matches the desired frame size. The padding units are h.264/h.265-compliant which will be ignored by the decoder during playback. Most importantly, the player will download the entire frame *including* padding units so that the amount of data transferred matches the source video frame sizes specified in the input trace data. Therefore, although a virtual video does not contain visual data from the source video, the network resources consumed in streaming it will be the same as streaming the source video, resulting in very similar streaming performance.

Researchers can use the VVG to generate virtual video streams for any of the 700+ video titles in the bitrate trace dataset according to any desired bitrate profile (either existing ones or new ones generated by the estimation model in Section III-C). These virtual videos can then be used in real-world DASH streaming experiments provided that the player supports direct recording of streaming performance metrics. Otherwise one can employ the SPM tool in the next section to measure the streaming performance of proprietary video players.

B. STREAMING PERFORMANCE METER

To measure the real-world performance of adaptive streaming systems one must record the frame-level playback timings and bitrate selections in a streaming session. While bitrate selections can be easily recorded at the server side (e.g., via server log), playback timings are more challenging to capture. One common approach is to modify the streaming player to insert instrumentation codes to record the playback timings at runtime (e.g., [7,8,11,12,15]). However, this requires access to the source codes of the player which is not always available. Moreover, care must be taken to ensure that such modifications will not produce unintended side-effects.

To tackle this problem, we developed a novel Streaming Performance Meter (SPM) to enable one to measure the streaming performance of *any* player *without* any modification. As illustrated in Fig. 8, the idea is to *record* the *video output* of a streaming player at a sufficiently high frame rate (e.g., 50~120 fps) in streaming a virtual video. For example, one can employ screen capture software (e.g., Screencast [43]) to record the video player's playback into an encoded video file in any supported format. If screen capture is not available or not accurate (c.f. Section V-B), then one can simply use a conventional video camera (e.g., GoPro) to record the playback device's streaming output directly as illustrated in Fig. 10.

In any case, the recorded video captures the virtual video frames which the SPM then analyzes to decode the: (i) frame index; (ii) bitrate version; and (iii) playback timing. These raw data can then be used as inputs to compute individual metrics such as average video bitrate, number/duration of playback rebuffering as well as overall QoE according to any desired QoE function.

V. PERFORMANCE VALIDATIONS

In this section we report experimental results to validate the accuracy of the segment/frame size estimation model in Section III-C and the SPM tool in Section IV-B.

A. SEGMENT/FRAME SIZE ESTIMATION MODEL

The polynomial-based estimation model takes segment sizes of known bitrate versions as input to compute the estimated segment sizes for an unknown target bitrate. To assess the accuracy of the model, we selected and re-encoded 100 HD Bluray video titles in both h.264 and h.265 according to three bitrate profiles (c.f. Table V): (i) 13 bitrate versions according to the augmented Apple profile [23]; (ii) 13 bitrate versions according to the Adobe profile [24]; and (iii) 15 bitrate versions according to the Microsoft profile [25]. In all cases, three segment durations (2s, 4s, 10s) were encoded for each video title.

We then applied the estimation model in Section III-C using segment size data from videos encoded using the augmented Apple profile to estimate the segment sizes for the *unknown* bitrate versions in the Adobe (11 unknown bitrates) and Microsoft (10 unknown bitrates) profiles. The estimated segment sizes were then compared to the ground-truth (obtained from actual re-encoding) to calculate the normalized segment size estimation error ε using

$$\varepsilon = \mathrm{E}\left[\left|\frac{\hat{b}_{s,j} - b_{s,j}}{b_{s,j}}\right| \forall j\right] \times 100\% \qquad (6)$$

where $\hat{b}_{s,j}$ is the estimated size for segment j for target bitrate version s while $b_{s,j}$ is the corresponding ground-truth segment size. We compare our 7th-order polynomial-based model with the linear model, logarithmic model, and exponential model. According to our experiments, the estimation errors across the two codecs (i.e., h.264 and h.265) and the three segment durations (i.e., 2s, 4s, 10s) are similar so we only show the results for h.264-encoded video with 2s segment duration.

TABLE V
BITRATE PROFILES FOR VALIDATION TEST
(ESTIMATED TARGET BITRATES IN BOLD)

	Bitrate profile (Mbps)
Augmented Apple Profile [23]	0.2, 0.4, 0.8, 1.2, 2.2, 3.3, 5.0, 6.5, 8.6, 10.0, 12.0, 16.0, 20.0
Adobe Profile [24]	**0.3**, **0.6**, **0.9**, 1.2, **1.5**, **2.0**, **2.5**, **3.0**, **3.5**, **4.0**, 5.0, **6.0**, **8.0**
Microsoft Profile [25]	0.4, **0.65**, **1.0**, **1.5**, **2.25**, **3.4**, **4.7**, **6.0**, **8.0**, 10.0, 12.0, **14.0**, 16.0, **18.0**, 20.0

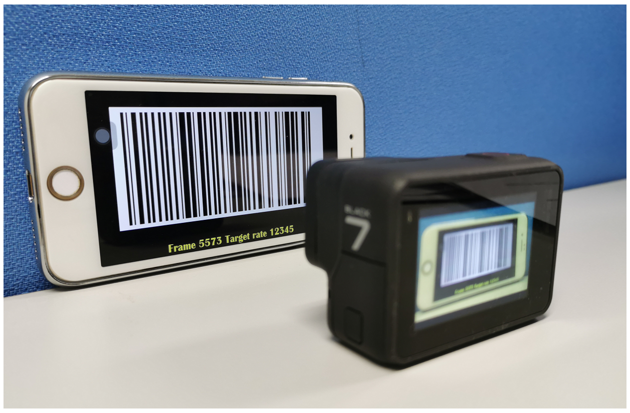


FIGURE 10. Recording the video playback of a smartphone through an external camera for subsequent SPM processing.

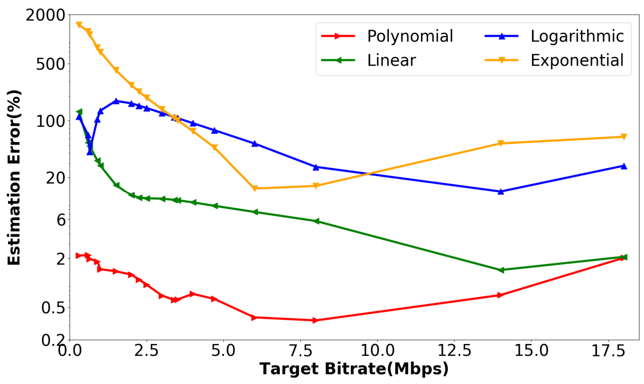


FIGURE 11. Estimation error across different target bitrates.

Fig. 11 plots the estimation error versus target bitrates. We can observe that the 7th-order polynomial model achieved the lowest estimation errors. The errors across different target bitrates are all within 2.2% and the mean error is 1.17%. Fig. 12 plots the estimation errors across 100 movies. The estimation error achieved by the polynomial model is again much lower than the others. Finally, the choice of 7th-order polynomial model is based on a sensitivity analysis as shown in Table VI.

Next, we evaluate the impact of the estimation error on trace-driven simulations, using MPC [32] as the adaptation algorithm for simulated streaming over 15 days' TCP throughput trace data [35]. We first estimated the segment sizes for the unknown bitrate versions in the Adobe profile from the augmented Apple's profile as described earlier. Next, we conducted two sets of simulations, both simulated streaming of 100 different video titles using the same 3G throughput trace data.

TABLE VI
SENSITIVITY ANALYSIS FOR POLYNOMIAL ORDER

Polynomial order	1	4	7	11
Mean estimation error (%)	22.7	3.90	1.17	43.41

TABLE VII
NORMALIZED PERCENTAGE ERROR (%) OF FOUR STREAMING PERFORMANCE METRICS

	Streaming Performance Metrics			
Estimation Model	Mean Bitrate	Bitrate Fluctuation	Rebuffering Duration	Rebuffering Number
Polynomial	0.85	3.36	4.77	4.27
Linear	1.44	5.05	18.6	17.5
Logarithmic	14.5	23.9	200.5	200.4
Exponential	10.1	24.6	220.5	218.4

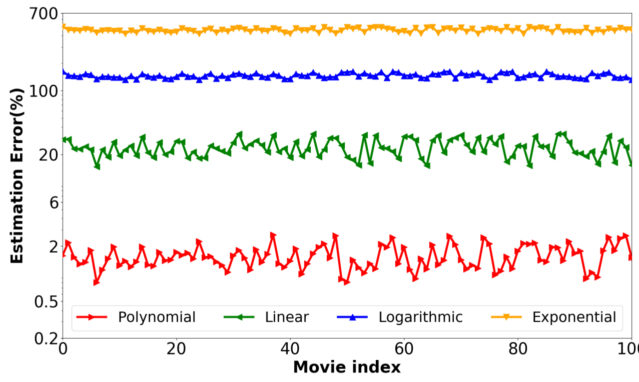


FIGURE 12. **Estimation error across different movies.**

The first set of simulation made use of the *estimated* segment size trace data while the second set made use of the *ground-truth* segment size trace data. The goal is to evaluate the impact of segment size estimation error on various streaming performance metrics and QoE. We define a normalized percentage error to evaluate each metric:

$$\partial = \left(\left| \hat{q} - q \right| / q \right) \times 100\% \tag{7}$$

where $\hat{q}$ and q are the values of the performance metric obtained via estimated and ground-truth segment size trace data respectively.

We measured four streaming performance metrics, namely *mean bitrate* – defined as the average bitrate selected by the adaptation algorithm; *bitrate fluctuation* – defined as the total magnitude of video bitrate changes; *rebuffering duration* – defined as the total time at which playback is suspended due to client buffer underflow in each session; and *rebuffering number* – defined as the total number of rebuffering events in each session. Note that the client suspends playback whenever it runs into buffer underflow and then resumes playback once a complete video segment is received [2-5].

Table VII summarizes the normalized percentage error for the four metrics presented earlier. We observe that the segment size estimation model achieved 0.85% and 3.36% error for mean bitrate and bitrate fluctuation respectively. The error in rebuffering is also very small – 4.77% and 4.27% for rebuffering duration and rebuffering number respectively. In all cases, the proposed polynomial model achieved significantly lower errors than the other three models.

To further evaluate the impact to overall QoE, we used the raw metrics to compute the overall QoE according to seven existing QoE functions:

QoE Function 1 (*U*1): Developed by Mok *et al.* [37]. Their QoE function is defined by

$$U_1 = 4.32 - 0.0672 L_{ti} - 0.742 L_{fr} - 0.106 L_{tr} \tag{8}$$

where L_{ti}, L_{fr} and $L_{tr} \in \{1,2,3\}$ are scores mapped according to Table III in [37] for startup delay, rebuffering number, and rebuffering duration respectively.

QoE Function 2 (*U*2): Developed by Joskowicz *et al.* [38]. Their QoE function is defined by

$$U_2 = E\left[4.75 - 4.5 e^{-0.77 R_i} \mid \forall i \right] \tag{9}$$

where R_i is the video bitrate selected for segment i.

QoE Function 3 (*U*3): Developed by Liu *et al.* [39]. Their QoE function is defined by

$$U_3 = -3.7 \times \mathit{buffratio} + \frac{\mathit{bitrate}}{20} \tag{10}$$

where *bitrate* is mean bitrate in kbps and *buffratio* is the ratio of total video rebuffering duration to the total video playback duration in percentage.

QoE Function 4 (*U*4): Developed by Yin *et al.* [32]. Their QoE function is defined in (5) earlier.

QoE Function 5 (*U*5): Developed by Mao *et al.* [33]. Their QoE function is defined by

$$U_5 = \frac{1}{K}\left(\sum_{k=1}^{K} l_k - \sum_{k=1}^{K-1} \left| l_{k+1} - l_k \right| - 2.66 Z_p \right) \tag{11}$$

where Z_p is the total video rebuffering duration, K is the total number of segments and

$$l_k = \log(r_k / r_{\min}) \tag{12}$$

where r_k is the bitrate selected for segment k and $r_{\min}$, is the lowest bitrate in the profile.

QoE Function 6 (*U*6): Developed by Mao *et al.* [33]. Their QoE function favors High Definition (HD) video:

$$U_6 = \frac{1}{K}\left(\sum_{k=1}^{K} h_k - \sum_{k=1}^{K-1} \left| h_{k+1} - h_k \right| - 8.0 Z_p \right) \tag{13}$$

where h_k (the mapping between bitrate and h_t is shown in Table VIII) is the video quality for segment k, K is the total number of segments, and Z_p is the total video rebuffering duration.

QoE Function 7 (*U*7): Developed by Li *et al.* [40]. Their QoE function is defined by

$$U_7 = 5(e^{-5.71 QoS_u})(e^{-1.607 QoS_l})(e^{-0.0416 QoS_d})(QoS_r^{\,8.94} e^{-8.94(QoS_r - 1)}) \tag{14}$$

TABLE VIII
THE MAPPING BETWEEN BITRATE AND H_T FOR ADOBE PROFILE

Bitrate(Mbps)→h_k
0.3→1, 0.6→1.67, 0.9→2.33, 1.2→3, 1.5→11, 2.0→12.4, 2.5→13.9, 3.0→15.5, 3.5→17.1, 4.0→18.9, 5.0→22.7, 6.0→26.8, 8.0→36.2

TABLE IX
NORMALIZED PERCENTAGE ERROR (%) UNDER SEVEN QOE FUNCTIONS

Estimation Model	QoE function						
	U_1	U_2	U_3	U_4	U_5	U_6	U_7
Polynomial	0.17	0.65	0.97	0.32	0.69	1.18	1.11
Linear	7.01	1.41	15.0	14.5	9.60	3.98	15.7
Logarithmic	178.9	2.07	98.3	174.5	141.8	68.0	162.4
Exponential	142.2	3.40	92.7	208.8	160.0	74.1	155.9

TABLE X
EVALUATION SETTING FOR SPM

Parameters	Values
Virtual video segment duration	2s
Virtual video frame rate	24 fps
Adaptation algorithm	MPC
Throughput	15 days 3G TCP throughput trace data
Recorded video frame rate	60 fps
Camera	GoPro Hero 7 Black
Desktop computer	Intel Core i7 3.4Ghz with Win10
Smartphone	IPhone 7 with iOS 12.4.1
TV	LG 55-inch 3D smart LED FHD TV
Screen-capture software (desktop)	Screencast [43]
Screen-capture software (smartphone)	iOS's built-in screen-recording software

```
frame index,  timestamp (s),  size (bytes),  target bitrate (kbps),  frame type
   3740,        155.91666,       31902,           6500,                P
   3741,        155.95833,       48570,           6500,                P
   3741,        155.95833,       48570,           6500,                P
   3741,        155.95833,       48570,           6500,                P
   3741,        155.95833,       48570,           6500,                P
   3742,        156.00000,       15492,           3300,                I
   3743,        156.04166,       11216,           3300,                P
```

FIGURE 13. A special frame-level trace data file with repeated frames (#3741) to emulate rebuffering events.

where QoS_u is the ratio of total video rebuffering duration to the total video playback duration, QoS_l is packet loss rate (set to 0 here as TCP is used in DASH), QoS_d is startup delay, and QoS_r is normalized video playout rate (set to 1 here as video playout adaptation [40] is not employed in the experiments).

Table IX summarizes the normalized percentage error for the seven QoE functions over 100 simulated streaming sessions. We observe that the proposed polynomial estimation model achieved significantly lower errors in all seven QoE functions, with mean errors all within 1.2%. The results clearly show that the errors introduced by the estimation model are sufficiently small so as to not have any significant impact to both streaming performance metrics and QoE obtained in simulations.

B. STREAMING PERFORMANCE METER

In this section we validate the accuracy of the streaming performance meter (SPM) via controlled experiments. The method is to first generate a special virtual video stream that *recorded* known bitrate changes and rebuffering events, and then employ the SPM to analyze it to obtain the raw streaming performance data. By comparing the measured data with ground-truth we can then quantify the measurement errors.

Specifically, we first conducted trace driven simulation using 15 days' 3G throughput trace to simulate streaming of 100 video titles encoded using the augmented Apple profile with MPC [32] as the adaptation algorithm. The videos were encoded by h.264 with segment duration of 2s. For each streaming session we recorded the bitrate adaptation sequence and rebuffering timings as ground truth.

Based on the recorded data we can then generate a special frame-level trace file that incorporated the bitrate sequence and rebuffering events. This is illustrated in Fig. 13 where video frame #3741 was repeated three times, indicating that rebuffering occurred after playback of the frame. This corresponds to the video player pausing playback during rebuffering, thereby repeating the same video frame, until playback resuming three frame times later. By feeding this special trace data file to the Virtual Video Generator we can then generate a *normal* video file that, when played back, will *reproduce* the visual impact of bitrate adaptation and rebuffering (i.e., via repeating frame #3741).

Next we played back the generated video file *locally* from disk (to avoid unintended variation in frame timings) and then recorded the video playback. The recorded video frame rate is set to 60 fps which is higher than the virtual video frame rate of 24 fps to increase the temporal resolution in recording frame playback timings.

We first evaluate the accuracy of five different video recording methods: (i) using an external camera to capture the video playback on a desktop computer; (ii) using an external camera to capture the video playback on a smartphone; (iii) using an external camera to capture the video playback on a TV; (iv) using an internal screen-capture software to screen-capture the video playback on a desktop computer; and (v) using an internal screen-capture software to screen-capture the video playback on a smartphone. Table X summarizes the devices and configurations adopted in the experiments.

The recorded videos were then analyzed by SPM as described in Section IV-B to obtain the raw streaming performance data, i.e., frame index, bitrate version and playback timing. By comparing the SPM-measured data against ground-truth data, we can then quantify the measurement errors.

TABLE XI
NORMALIZED PERCENTAGE ERROR (%) OF FOUR STREAMING PERFORMANCE METRICS MEASURED BY SPM

Normalized Percentage Error (%)	QoE Metric			
	Mean Bitrate	Bitrate Fluctuation	Rebuffering Duration	Rebuffering Number
Camera capture (desktop)	0	0	1.52	3.31
Camera capture (mobile phone)	0	0	1.73	4.01
Camera capture (TV)	0	0	1.44	3.25
Screen capture (desktop)	0	0	0.95	2.73
Screen capture (mobile phone)	0.77	3.56	10.73	13.54

TABLE XII
NORMALIZED PERCENTAGE ERROR (%) UNDER SEVEN QOE FUNCTIONS MEASURED BY SPM

Normalized Percentage Error (%)	QoE function						
	U_1	U_2	U_3	U_4	U_5	U_6	U_7
Camera capture (desktop)	0	0	0.38	0.55	0.78	0.61	0.77
Camera capture (mobile phone)	0	0	0.43	0.65	0.81	0.87	0.95
Camera capture (TV)	0	0	0.33	0.56	0.72	0.66	0.60
Screen capture (desktop)	0	0	0.29	0.40	0.44	0.42	0.39
Screen capture (mobile phone)	2.1	0.5	3.77	7.18	7.01	8.20	9.70

Table XI summarizes the normalized percentage errors for each of the four raw performance metrics calculated according to (7). There are three observations. First, the three experiments using video camera recordings all achieved reasonably accurate results, with zero error in mean bitrate and bitrate fluctuations, and relatively small errors in rebuffering duration and number. The errors in the latter are due to: (i) the video player's playback framerate/timing jitters; (ii) the video recorder's recording framerate/timing jitters; and (iii) quantization errors due to the finite recording frame rate – this is why a higher frame rate is desirable for recording the streaming playback.

Second, the experiment using screen-capture in desktop computer also performed well, with no error in bitrate data and relatively small errors in rebuffering data. Finally, the results from running screen-capture in smartphone (iPhone 7 in this case) didn't perform so well, with some errors in bitrate data and considerably larger errors in rebuffering data (>10%). This is caused by the smartphone's limited processing power in running both the video player *and* the screen-capturer concurrently, resulting in missed frame captures and unstable frame capture timings.

In addition to individual metrics, we also computed the normalized errors for the seven QoE functions and summarized the results in Table XII. In all cases, except the smartphone screen-capture experiment, the error for U_1 and U_2 are both zero. This is because in U_1 the rebuffering duration and rebuffering number were converted into discrete scores {1, 2, 3}, which masked small errors in the raw data. U_2 is computed solely from video bitrate and thus was not affected by errors in the rebuffering data. The mean normalized errors for U_3 to U_7 are negligible at <1%. Again, the smartphone with screen-capture experiment exhibited larger errors as expected. Newer smartphones with more powerful CPUs may reduce the errors but one may want to validate the device's screen-capture performance beforehand. In any case, using a video camera is always an option and it also prevents any potential interferences with the streaming player (when running screen-capture concurrently).

Overall, the results demonstrated that SPM can measure streaming performance data with good accuracy using merely a video recording of the streaming playback, thereby enabling the measurement of *any* streaming platforms, even proprietary ones such as the iOS Quicktime player illustrated here, in DASH streaming experiments.

VI. SUMMARY AND FUTURE WORK

The EmuStream platform developed in this work offers new opportunities for research in adaptive video streaming. First, the availability of a large real-world bitrate trace dataset paves the way to explore properties of real-world HD/UHD video contents. Further research is warranted to analyze the bitrate trace dataset to uncover useful statistical properties or models that could be applied to the optimization of existing and the design of future adaptive streaming algorithms.

Second, the Virtual Video Generator (VVG) together with the Streaming Performance Meter (SPM) offer a new way to conduct experiments in adaptive video streaming. By eliminating the need to modify the video player, it not only simplifies streaming experiments, but also enables measurement of proprietary video players commonly used in the industry.

The EmuStream platform are available at *http://emustream.mclab.org*. The software is released under GPL while the bitrate trace dataset is released under the Creative Commons Attribution 4.0 International License [26].

ACKNOWLEDGMENT

The authors wish to thank the associate editor and the anonymous reviewers for their insightful comments in improving this paper.

REFERENCES

[1] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP: Standards and Design Principles," *Proc. ACM Multimedia Syst.*, San Jose, USA, Feb 2011, pp.133-144.

[2] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hobfeld and P. Tran-Gia,"A Survey on Quality of Experience of HTTP Adaptive Streaming," *IEEE Commun. Surveys Tuts.*, vol.17(1), Q1 2015, pp.469-492.

[3] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of Quality of Experience of Video-on-Demand Services: A Survey," *IEEE Commun. Surveys Tuts.*, vol.18(1), Feb 2016, pp.401-418.

[4] J. Kua, G. Armitage, and P. Branch, “A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming over HTTP,” *IEEE Commun. Surveys Tuts*., vol.19(3), Mar 2017, pp.1842-1866.

[5] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, R. Zimmermann, “A Survey on Bitrate Adaptation Schemes for Streaming Media over HTTP,” *IEEE Commun. Surveys Tuts*., vol.21(1), Aug 2018, pp.562-585

[6] *Big Buck Bunny*. [Online] https://peach.blender.org/

[7] S. Lederer, C. Müller, C. Timmerer, “Dynamic Adaptive Streaming over HTTP Dataset,” *Proc. ACM Multimedia Syst*., Chapel Hill, North Carolina, Feb 2012, pp. 89-94.

[8] S. Lederer, C. Mueller, C. Timmerer, “Distributed DASH Dataset”, *Proc. ACM Multimedia Syst*., Oslo, Norway, Feb. 2013, pp. 131-135.

[9] J. Le Feuvre, J. M. Thiesse, M. Parmentier, “Ultra High Definition HEVC DASH Data Set”, *Proc. ACM Multimedia Syst*., Singapore, Singapore, Mar. 2014, pp. 7-12.

[10] C. Kreuzberger, D. Posch, H. Hellwagner, “A Scalable Video Coding Dataset and Toolchain for Dynamic Adaptive Streaming over HTTP”, *Proc. ACM Multimedia Syst*., Portland, Oregon, Mar. 2015, pp. 213-218.

[11] J. J. Quinlan, A. H. Zahran, C. J. Sreenan, “Datasets for AVC (H. 264) and HEVC (H. 265) Evaluation of Dynamic Adaptive Streaming over HTTP (DASH)”, *Proc. ACM Multimedia Syst*., Klagenfurt, Austria, May 2016, pp. 51-57.

[12] J. J. Quinlan, C. J. Sreenan, “Multi-profile Ultra High Definition (UHD) AVC and HEVC 4K DASH Datasets”, *Proc. ACM Multimedia Syst*., Amsterdam, Netherlands, Jun. 2018, pp. 375-380.

[13] Zabrovskiy, C. Feldmann, C. Timmerer, “Multi-Codec DASH Dataset”, *Proc. ACM Multimedia Syst*., Amsterdam, Netherlands Jun. 2018, pp. 438-443.

[14] C. G. Bampis, Z. Li, A. K. Moorthy, “Study of Temporal Effects on Subjective Video Quality of Experience,” *IEEE Transactions on Image Processing*, vol.26 (11), Nov 2017, pp. 5217-5231.

[15] Z. Duanmu, A. Rehman, Z. Wang, “A Quality-of-Experience Database for Adaptive Video Streaming,” *IEEE Transactions on Broadcasting*, vol.64(2), Jun 2018, pp. 474-487.

[16] V. Jacobson, R. Frederick, S. Casner, and H. Schulzrinne, “Real-Time Transport Protocol (RTP),” https://www.ietf.org/rfc/rfc3550.txt, 2014, online; accessed on Nov. 21, 2017.

[17] H. Schulzrinne, “Real Time Streaming Protocol Version 2.0,” https://tools.ietf.org/html/rfc7826, 2016, online; accessed on Nov. 21, 2017.

[18] M. Leszczuk, M. Hanusiak, M. C. Q. Farias, “Recent Developments in Visual Quality Monitoring by Key Performance Indicators,” *Multimedia Tools and Applications*, vol. 75(17), Feb 2018, pp 10745-10767

[19] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC Video Coding Standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol.13(7), July 2003, pp.560–576

[20] G. J. Sullivan, J. R. Ohm, Woo-Jin Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,” *IEEE Transactions on Circuits and Systems for Video Technology,* vol.22(12), Dec. 2012, pp.1649-1668

[21] D. Mukherjee, J. Han, J. Bankoski, R. Bultje, A. Grange, J. Koleszar, P. Wilkins, and Y. Xu, “A Technical Overview of VP9: The Latest Open-Source Video Codec,” *SMPTE Motion Imaging Journal,* vol.124(1), Oct 2015, pp.44-54.

[22] *AV1 Software Repository*, retrieved on Mar 2018, [Online] https://aomedia.googlesource.com/aom

[23] *Best Practices for Creating and Deploying HTTP Live Streaming Media for the iPhone and iPad*, Apple Inc, retrieved on Aug 2016. [Online] https://developer.apple.com/library/ios/technotes/tn2224/_index.html

[24] *Video Encoding and Transcoding Recommendations for HTTP Dynamic Streaming on the Adobe® Flash® Platform,* Oct 2010, [Online]. http://download.macromedia.com/flashmediaserver/http_encoding_recorecommendat.pdf

[25] *Microsoft Video Encoding and Transcoding Recommendations* [Online] https://docs.microsoft.com/en-us/azure/media-services/previous/media-services-mes-preset-h264-multiple-bitrate-1080p

[26] *Creative Commons License* [Online] https://creativecommons.org/licenses/by/4.0/

[27] *FFmpeg*. [Online] https://ffmpeg.org

[28] *Intel Quick Sync Video* [Online] https://www.intel.co.uk/content-/www/uk/en/architecture-and-technology/quick-sync-video/-quick-sync-video-general.html

[29] *x264 Library* [Online] https://www.videolan.org/developers/x264.-html

[30] *Common Media Application Format (CMAF)* [Online] https://mpe-g.chiariglione.org/standards/mpeg-a/common-media-application-format

[31] *The Android Open Source Project. (2015, Feb.)* Android Git Repositories. [Online] https://android.googlesource.com/platform/frameworks/av/+/master/media/li-bstagefright/httplive/LiveSession.cpp

[32] X. Yin, A. Jindal, V. Sekar and B. Sinopoli, “A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP,” *Proc. ACM SIGCOMM*, London, United Kingdom, Aug 2015, pp.325-338.

[33] H. Mao, R. Netravali, M. Alizadeh, “Neural Adaptive Video Streaming with Pensieve,” *Proc. ACM SIGCOMM*, Los Angeles, CA, USA, Aug 2017, pp. 197-210.

[34] Guanghui Zhang, Jack Y. B. Lee, “Ensemble Adaptive Streaming-A New Paradigm to Generate Streaming Algorithms via Specializations”, *IEEE Transactions on Mobile Computing*, Apr. 2019.

[35] *Mobile Throughput Trace Data*. [Online] http://sonar.mclab.info/tracedata/TCP/3G/

[36] Y. Liu and J.Y.B. Lee, “Post-Streaming Rate Analysis - A New Approach to Mobile Video Streaming with Predictable Performance,” *IEEE Transactions on Mobile Computing*, vol.16(12), Dec 2017, pp.3488-3501.

[37] R.K. Mok, E.W. Chan and R.K. Chang, “Measuring the Quality of Experience of HTTP Video Streaming,” *Proc. IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Dublin, Ireland, May 2011, pp. 485-492.

[38] J. Joskowicz and J. C. L. Ardao, “A Parametric Model for Perceptual Video Quality Estimation,” *Telecommunication Systems*, vol. 49(1), Jan 2012, pp 49-62.

[39] X. Liu, F. Dobrian, H. Milner, J. Sekar, V. Jiang, I. Stoica and H. Zhang, “A Case for a Coordinated Internet Video Control Plane,” *Proc. ACM SIGCOMM*, Helsinki, Finland, Aug 2012, pp 359-370.

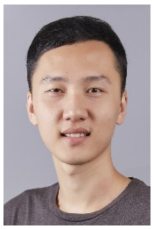
[40] M. Li, C. Y. Lee, “A Cost-effective and Real-time QoE Evaluation Method for Multimedia Streaming Services,” *Telecommunication Systems*, vol. 59(3), Jul. 2015, pp 317-327.

[41] M. Li, C. L. Yeh, S. Y. Lu, “Real-time QoE Monitoring System for Video Streaming Services with Adaptive Media Playout,” *International Journal of Digital Multimedia Broadcasting*, vol. 2018, Article ID 2619438, 11 pages, 2018.

[42] *Video Frame Type* [Online] https://en.wikipedia.org/wiki/Video_compression_picture_types

[43] *Screencast* [Online] https://screencast-o-matic.com/

[44] Guanghui Zhang and Jack Y. B. Lee, "LAPAS: Latency-Aware Playback-Adaptive Streaming," *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, Marrakech, Morocco, April 15-19, 2019.



GUANGHUI ZHANG received his B.Eng. in Electronic Information Engineering from Shandong Normal University, Jinan, China, in 2013 and M.Sc. degree in Electronic Science and Technology from Peking University, Beijing, China, in 2016. He is currently a Ph.D. candidate in the Department of Information Engineering, the Chinese University of Hong Kong, where he participated in the research and development of multimedia technology.

RUDOLF K.H. NGAN received the BSc degree in Physics from the Chinese University of Hong Kong in 1995 and MSc degree in Computer Science from the City University of Hong Kong in 2016. He has been working as a research assistant in the Department of Information Engineering of the Chinese University of Hong Kong from 2000 to 2019.

JACK Y. B. LEE (M'95–SM'03) received his B.Eng. and Ph.D. degrees in Information Engineering from the Chinese University of Hong Kong, Shatin, Hong Kong, in 1993 and 1997, respectively. He is currently an Associate Professor with the Department of Information Engineering of the Chinese University of Hong Kong. His research group focuses on research in multimedia communications systems, mobile communications, protocols, and applications. He specializes in tackling research challenges arising from real-world systems. He works closely with the industry to uncover new research challenges and opportunities for new services and applications. Several of the systems research from his lab have been adopted and deployed by the industry.