Using Session Modeling to Estimate HTTP-Based Video QoE Metrics From Encrypted Network Traffic

Tarun Mangla, Emir Halepovic, *Member, IEEE*, Mostafa Ammar, *Fellow, IEEE*, and Ellen Zegura, *Fellow, IEEE*

***Abstract*—Understanding the user-perceived quality of experience (QoE) of HTTP-based video has become critical for content providers, distributors, and network operators. For network operators, monitoring QoE is challenging due to lack of access to video streaming applications, user devices, or servers. Thus, network operators need to rely on the network traffic to *infer* key metrics that influence video QoE. Furthermore, with content providers increasingly encrypting the network traffic, the task of QoE inference from passive measurements has become even more challenging. In this paper, we present a methodology called eMIMIC that uses passive network measurements to estimate key video QoE metrics for encrypted HTTP-based adaptive streaming (HAS) sessions. eMIMIC uses packet headers from network traffic to model an HAS session and estimate video QoE metrics, such as average bitrate and re-buffering ratio. We evaluate our methodology using network traces from a variety of realistic conditions and ground truth collected using a lab testbed for video sessions from three popular services, two video on demand (VoD) and one Live. eMIMIC estimates re-buffering ratio within 1% point of ground truth for up to 75% sessions in VoD (80% in Live) and average bitrate with error under 100 Kb/s for up to 80% sessions in VoD (70% in Live). We also compare eMIMIC with recently proposed machine learning-based QoE estimation methodology. We show that eMIMIC can predict average bitrate with 2.8%–3.2% higher accuracy and re-buffering ratio with 9.8%–24.8% higher accuracy without requiring any training on ground truth QoE metrics. Finally, we show that eMIMIC can estimate real-time QoE metrics with at least 89.6% accuracy in identifying buffer occupancy state and at least 85.7% accuracy in identifying average bitrate class of recently downloaded chunks.**

***Index Terms*—HTTP adaptive streaming, quality of experience, network operator, network measurements, QoE inference.**

I. Introduction

UNDERSTANDING the user-perceived Quality of Experience (QoE) is important for network operators, as it can help with efficient provisioning and management [2], [3]. However, estimating QoE is challenging in general since not only is it subjective, but also application-specific, and the operators do not have access to applications at end user devices to observe ground truth of key objective metrics impacting QoE. Instead, they have to rely on passive measurements of network traffic to estimate objective QoE metrics. This works well for applications whose objective QoE metrics are directly reflected by, for example, observable network Quality of Service (QoS) metrics, such as packet delay and jitter for voice quality [4]. However, this can be challenging for HTTP-based Adaptive Streaming (HAS) video, a major contributor to network traffic [5], because of its robustness to short-term variations in the underlying network QoS resulting from the use of the video buffer and bitrate adaptation.

Existing approaches [6], [7], [8], [9] for HAS video QoE estimation propose using machine learning algorithms to learn the relationship between network QoS metrics and application-layer QoE metrics. However, these approaches have several limitations. First, they require ground truth QoE metrics for initial training, which are not generally available to operators. Second, different video services use different service design parameters such as encoding bitrates and bitrate adaptation logic. Thus, relationships learned for one service do not necessarily generalize for others. Third, these approaches give a categorical estimate of the QoE metrics which might not be adequate for active QoE-based traffic management as proposed recently [2], [3].

In prior work [10], we presented a highly accurate and practical QoE inference approach (with the system details presented in [11]) for HAS video, called MIMIC. MIMIC relied on extracting information from the application layer, i.e., Uniform Resource Identifiers (URIs) and other HTTP headers. With increasing number of video service providers using end-to-end encryption, MIMIC loses visibility into the key pieces of information needed for QoE inference.

To overcome this challenge, we present a QoE estimation approach for encrypted HAS video, called eMIMIC, which works by reconstructing the chunk-based delivery sequence of a video session from packet traces of encrypted traffic. This reconstructed sequence is then used to model a video session based on high-level HAS properties, which are generally consistent across services.

Manuscript received November 23, 2018; revised March 15, 2019; accepted April 1, 2019. Date of publication June 26, 2019; date of current version September 9, 2019. This work is funded in part by NSF grant NETS 1409589. This paper is an extended version of our previous work titled "*eMIMIC: Estimating HTTP-based Video QoE Metrics from Encrypted Network Traffic*" that appeared in *Proc. IEEE TMA, 2018* [1]. The associate editor coordinating the review of this paper and approving it for publication was M. Mellia. *(Corresponding author: Tarun Mangla.)*

T. Mangla, M. Ammar, and E. Zegura are with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: tmangla3@gatech.edu).

E. Halepovic is with AT&T Labs Research, Bedminster, NJ 07921 USA.

Digital Object Identifier 10.1109/TNSM.2019.2924942

From the accurately built model, eMIMIC can estimate average bitrate, re-buffering ratio, bitrate switches and startup time, the key objective metrics that influence HAS QoE [12]. An operator may need to further model the impact of these metrics taken together on the user experience, either through user studies [13] or objective data analysis [14], [15]. This step is complimentary to the estimation of the individual objective QoE metrics and is out of scope of this paper. The key objective of this paper is to demonstrate feasibility and accuracy of the cross-layer approach to infer service-level QoE metrics from network-level passive measurements.

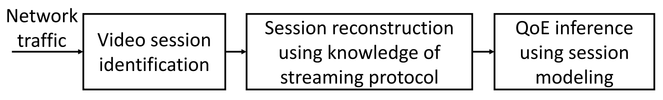
To facilitate the QoE inference from encrypted video sessions, we develop an experimental framework with automated streaming and collection of network traces and ground truth of video sessions, as well as QoE metric estimation. We use this framework to do an extensive evaluation of eMIMIC with three popular commercial video streaming services out of which two are video on demand (VoD) and one is a Live streaming service.

Furthermore, we replicate a recently proposed machine learning-based QoE estimation approach, hereon referred to as ML16 [7], by fully implementing and applying it to the same two video services. This helps in understanding the differences in performance and accuracy between the two QoE estimation approaches, so that they can be further evolved and improved. Finally, we evaluate eMIMIC in estimating real-time QoE metrics.

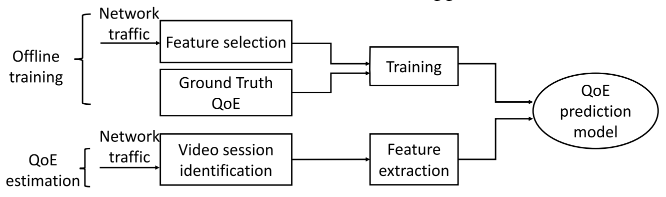
Our contributions are summarized as follows:

- We present eMIMIC, a methodology that uses passive measurements at network-layer to estimate service-level video QoE metrics of the encrypted video sessions.
- We develop an experimental framework for automated streaming and collection of network traces and ground truth QoE metrics of video sessions of three popular video streaming service providers. Using this framework under realistic network conditions, we show that eMIMIC estimates re-buffering ratio within one percentage point of ground truth for up to 75% of video sessions in VoD (80% in Live), and average bitrate with error under 100 kbps for up to 80% (70% in Live) of sessions.
- We compare eMIMIC with ML16 [7] and show that for categorical prediction (*low*, *medium* and *high*) of QoE metrics, eMIMIC has 2.8%-3.2% higher accuracy in classifying average bitrate and 9.8%-24.8% higher accuracy in classifying re-buffering ratio, without requiring training on any ground truth QoE metrics. We also find that ML16 does not generalize across video services.
- We show that eMIMIC can estimate real-time QoE metrics with at least 89.6% accuracy in identifying buffer occupancy state and at least 85.7% accuracy in identifying average bitrate class of recently downloaded chunks.

The remainder of the paper is organized as follows. We begin by describing different categories of QoE inference approaches and design requirements of an ideal approach in Section II. Section III describes the related work. Section IV presents our QoE inference methodology, followed by its evaluation in Section V. Section VI discusses some of the outstanding issues in video QoE inference from passive network measurements, while Section VII concludes the paper.


Network traffic
Video session identification
Session reconstruction using knowledge of streaming protocol
QoE inference using session modeling

(a) Overview of SM-based approach



(b) Overview of ML-based approach

Fig. 1. Overview of QoE inference approaches.

II. Background and Design Requirements

A. QoE Inference Methods

Existing video QoE inference approaches using passive network measurements can be broadly classified into two categories; *Session Modeling-based* (SM-based) and *Machine Learning-based* (ML-based).

SM-based approach: This approach infers QoE by modeling a video session using the properties of the underlying streaming protocol (Figure 1a). For unencrypted HAS video, MIMIC estimates the key video QoE metrics by modeling a video session as a sequence of chunks whose information is directly extracted from HTTP requests logged by a Web proxy [10].

ML-based approach: This approach infers QoE by correlating the network observable metrics such as packet delay, loss and throughput with the video QoE metrics using machine learning algorithms. Figure 1b shows a high-level overview of this approach. Implemented as a supervised ML-based method, it has an offline phase to build a QoE prediction model. This phase consists of selecting *useful* features to be extracted from network traffic and labeling them with corresponding ground truth, using which the algorithm learns the relationship between features and ground truth. Variants of this approach have been proposed that differ either in the feature selection or the training methodology (details in Section III).

B. Design Requirements

We motivate eMIMIC by describing the design requirements of an ideal QoE inference approach for a network operator.

- *Works on encrypted traffic:* Given an increased use of end-to-end encryption in HAS, this is a critical requirement for operators. Clearly, ML-based approaches will work if the required features can be collected from encrypted traffic. However, existing SM-based approaches that rely on visibility of HTTP transactions will not.
- *Minimally dependent on QoE ground truth:* An ideal QoE estimation method should not introduce extensive overhead in incorporating ground truth. The disadvantages of ML-based approach include a requirement to

collect the extensive ground truth measurement under a wide variety of network conditions, followed by training and validation of the learned model. Recent works propose methods to obtain ground truth through player instrumentation [6], [8], logging unencrypted versions of the traffic [7] or using a trusted proxy [9]. Unfortunately, there is no guarantee that any video service will support these approaches. On the other hand, an SM-based approach needs no training and minimal ground truth for validation. It may only need a few design parameters that can be easily obtained with a handful of test runs, as we demonstrate with eMIMIC.

- *Generalizes across different services:* To understand the QoE of many video services in its network, operators would prefer an approach that generalizes well. Given that content providers differ in system design and player implementations, ML-based models learned for one service do not necessarily generalize across different services, as we show in Section V-E. An SM-based approach, however, does not significantly suffer from this limitation since the underlying HAS properties do not change much across services.
- *Provides quantitative measures:* For active QoE-based traffic management, such as QoE-based resource allocation [2], [3], operators may need quantitative measures of QoE metrics. ML-based approaches typically provide categorical estimates of QoE with two (*good* or *bad*) or three (*low*, *medium* and *high*) categories whereas an SM-based approach estimates quantitative values of QoE metrics.

Takeaway: It is clear that an SM-based QoE inference approach that also works for encrypted traffic would be preferable for operators, as it would satisfy all design requirements. Therefore, we design eMIMIC, an SM-based approach that works on encrypted traffic.

III. Related Work

A. Video Traffic Classification

In order to use the network measurements for inferring video QoE, video traffic first needs to be separated from other network traffic. Several traffic classification approaches exist that focus on classifying the application protocol (like HTTP vs FTP) based on the network traffic properties [16]. More recently, traffic classification efforts have focused on classifying applications within HTTP. Shbair *et al.* [17] use ML to classify application classes within HTTPS traffic. ML-based techniques have been proposed to specifically identify HAS video flows in the network [18], [19]. eMIMIC currently uses Server Name Indication (SNI) field in TLS handshake [20] for video traffic identification. However, it can also use aforementioned approaches for filtering video traffic in case TLS headers are not available to an operator.

B. QoE-Based Network Management

Several in-network optimizations have been suggested that take into account video QoE. Mansy *et al.* [3] and Chen *et al.* [21] present QoE-aware schedulers in the network to improve fairness among different adaptive video streams. Similarly, network-assisted bitrate adaptation frameworks have been suggested for adaptive video flows [22], [23]. Kassler *et al.* [24] propose QoE-based routing using software defined networking (SDN). Mustafa *et al.* [25] propose using SDN for in-network video QoE-aware resource management. eMIMIC can be used by these approaches to understand the impact of in-network modifications on video QoE.

C. ML-Based QoE Inference Approaches

Researchers have proposed different ML-based approaches to infer video QoE using network measurements. These ML-based approaches differ in one or more ways such as the target QoE metric, machine learning model and features, and training methodology including collection of ground truth QoE metrics.

For progressive streaming, *OneClick* [26] and *HostView* [27] propose obtaining the subjective ground truth QoE through user feedback in the training phase and using regression-based models for their estimation. *Prometheus* [6] suggests using LASSO regression trained using instrumented clients to estimate objective video QoE metrics.

For encrypted HAS, ML16 [7] and *BUFFEST* [9] capture the QoE metrics sent by the player to the content provider on the network and build decision tree-based classifiers to estimate individual QoE metrics and buffer occupancy, respectively. Orsolic *et al.* [8] develop an Android application to obtain ground truth QoE for YouTube and evaluate different machine learning approaches for inferring different QoE classes. Using data from a similar YouTube client application and statistics derived from IP headers as the feature vector, Tsilimantos *et al.* [19] train different machine learning models to estimate the buffer state from the IP packet headers. Mazhar and Shafiq [28] classify different QoE metrics in real-time using decision trees and IP headers for QUIC and TCP headers for HTTPS video. In contrast, we rely on the properties of HAS instead of using machine learning and thus minimize the training overhead.

D. ML-Based QoE Inference Approaches

SM-based approaches have been proposed for different streaming protocols. Schatz *et al.* [29] and Dimopoulos *et al.* [30] estimate video re-buffering by modeling a session using TCP headers and HTTP headers, respectively. However, these methods were designed for HTTP progressive streaming and would not work for HAS video. Our recent work, MIMIC [10], provides an SM-based approach to infer key QoE metrics in the case of unencrypted HAS. eMIMIC extends session modeling for QoE inference in encrypted HAS.

IV. Methodology

This section describes the HAS chunked delivery principles used for reconstructing video sessions using eMIMIC, the challenges and solutions in extracting chunk-level details of the session, and how QoE metrics are inferred.

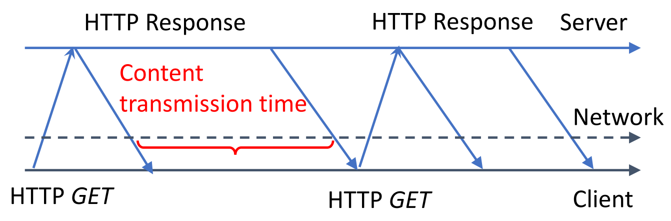


Fig. 2. Data flow of chunk requests and responses.

A. Chunked Video Delivery in HAS

In HAS, a video is split into chunks which are typically of same duration. Each of these chunks are encoded at pre-defined quality levels determined by encoding bitrate and resolution and hosted on a standard HTTP server. The video player at the client has a bitrate adaptation logic that decides the bitrate of media chunks to download. The video metadata such as chunk encoding levels and request URI is obtained by downloading a *manifest* file at the beginning of video session.

The network traffic corresponding to the media chunks in an HAS video session consists of a sequence of HTTP GET requests and responses. When the client requests the video, the player first downloads the *manifest* file by sending an HTTP GET request to the server. The player then sends an HTTP GET request for the first chunk. Once the video chunk has been fully downloaded, the player sends the request for the next chunk, whose bitrate is decided based on the past chunk throughput and/or current buffer occupancy [31], and this process repeats (Figure 2). *The video session at the client can be modeled using this strong serial request-response pattern corresponding to chunk downloads observed on the network.*

B. Challenges in Designing eMIMIC

1) HTTP Request Reconstruction: An SM-based approach abstracts an HAS video session as a sequence of video chunks appearing as HTTP GET requests on the network. For unencrypted network traffic, these requests can be logged by a passive monitor or a transparent Web proxy. However, this does not work when Transport Layer Security (TLS) is used, as is common today, where HTTP headers are encrypted. We note that parsing limited clear-text TLS headers is not a feasible approach to distinguish individual chunks, since multiple, or even all, chunks, can be requested within one TLS transaction.

Idea: We explore if TCP headers can be used for HTTP-level session reconstruction. Figure 2 shows the flow of video data for a sequence of HTTP requests and responses on a single TCP connection. The data flow in an HTTP transaction has an important traffic directionality property, i.e., request flows from client to server, followed by response flowing in the opposite direction. This directionality and sequence in the data flow of HTTP traffic can be used to identify the boundaries of HTTP request-response pairs. This methodology has been used to identify the size of Web objects in HTTPS traffic [32].

It is important to note that this approach would not work correctly if the HTTP requests were pipelined. However, in practice, video players typically do not pipeline HTTP requests. This is because pipelining may cause self-contention for bandwidth among the chunks, potentially causing head-of-line blocking, as well as diminishing the ability of the player to quickly adapt to changing network conditions.

Solution: For a TCP-flow f corresponding to video session V, we log the source IP address of every packet in the flow. A packet with non-zero payload size is tagged as an HTTP request if the source IP address matches the client IP address. The subsequent non-zero payload size packets in f with the server IP address as the source are tagged as the HTTP response. The end of the response is determined by one of the following conditions: i) a new packet from the client on the same flow indicating a new HTTP request or ii) an inactivity period of greater than some pre-defined threshold (5 seconds in our experiments) or iii) the closing of the TCP connection indicated by TCP RST or FIN flag. In addition, TCP retransmissions are logged. The size of the response is estimated by adding the payload sizes of all the packets tagged as response and adjusted to account for re-transmissions. The start time and the end time of an HTTP transaction are obtained from the timestamp of the first packet tagged as a request and the last packet tagged in the corresponding response, respectively. TCP ACKs with no payload are ignored.

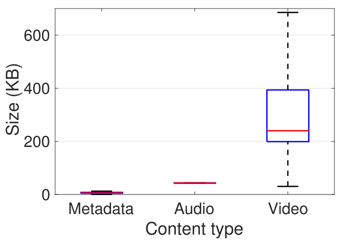
Applying this approach to all TCP flows in a video session, we can reconstruct HTTP transactions, along with the size (S_i) and download start time (ST_i) and end time (ET_i) for every chunk i. This approach can also be applied to UDP-based transport such as QUIC, assuming the same request-response sequence, but without accounting for retransmissions or using TCP flags for response termination.

2) Media Type Classification: The reconstructed HTTP transactions in the above methodology will include multiple media types namely video, audio, and metadata, such as the *manifest* file. Some services separate audio and video content which means that they appear as separate transactions in the network traffic. To model a session, it is important to identify video (and audio, if separate) chunks and filter out the metadata.

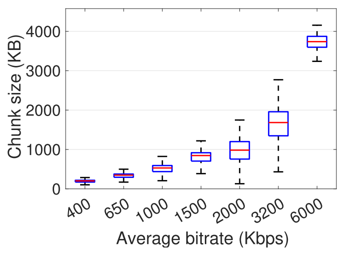
Idea: We use the estimated response sizes obtained from the HTTP reconstruction step to identify the media type. The size of metadata is usually smaller than audio or video as it consists of text files. Audio chunks are encoded at Constant Bit Rate (CBR) with one or two bitrates levels. Thus, they can be identified based on the size and its consistency.

Figure 3a illustrates this by showing the distribution of response sizes of video, audio and metadata obtained from the HTTP logs of 1005 VoD2 sessions collected by a trusted proxy (see Section V for details). The media type is identified from the request URI of the HTTP logs. Metadata HTTP logs are smaller than 30 KB and most of the audio HTTP logs are around 42 KB. However, we observe a small proportion of video chunks that are similar in size to audio chunks. To reduce the probability of misclassifying these as audio, we use the insight that audio and video playback is synchronized, and hence the amount of audio and video downloaded and stored in the buffer should be similar in terms of duration.

Solution: We first determine a minimum size threshold (S_{min}) for identifying HTTP transactions corresponding to the metadata. This is based on the minimum bitrate levels



(a) Distribution of HTTP transaction size for different media types



(b) Chunk size vs. average bitrate

Fig. 3. Chunk and bitrate characterization for VoD2.

of video and audio obtained by inspecting *manifest* files of several videos. For services that separate audio and video, the expected response size of audio chunks is calculated based on the audio bitrates used. A range $[A_{min}, A_{max}]$ is determined to identify an HTTP transaction corresponding to the audio chunks. We use a range instead of a single value for two reasons: i) there exist small variations in size of the audio chunks despite being CBR encoded; ii) the estimated size of reconstructed HTTP transaction may have errors. Furthermore, to avoid misclassifying a video chunk with actual size in the expected audio size range, we track the audio and video content downloaded in seconds. We fix a threshold T_{ahead} such that the audio content downloaded so far is no more than T_{ahead} seconds of the downloaded video content.

Thus, a reconstructed transaction is tagged as *metadata* if its size is less than S_{min}; as *audio* if its size is in the range $[A_{min}, A_{max}]$ and the audio content downloaded is at most T_{ahead} seconds more than video; and as *video* otherwise.

3) Estimating Bitrate of Video Chunks: After identifying the video chunks in a session, we need to estimate their bitrate. This is used to calculate average bitrate and bitrate switches.

Idea: One way to estimate chunk bitrate is to use its estimated size. More specifically, we can divide the chunk size by its duration and assign it to the *nearest* bitrate in the bitrate set of the video service. However, video services typically use Variable Bit Rate (VBR) encoding, which means that the chunk size can deviate, sometimes significantly, from the average bitrates based on the underlying video scene complexity. Figure 3b illustrates this by showing the distribution of chunk sizes (from HTTP logs) with their average bitrate levels (from request URI) for the 1005 VoD2 video sessions. The majority of chunk sizes are a close match to the average bitrates. However, there are cases where the chunk sizes overlap between two consecutive bitrate levels. Thus, using size alone can lead to errors in bitrate estimation.

To overcome this problem, we use an additional insight that players usually switch bitrate when the network bandwidth changes. Thus, a bitrate switch would be most likely accompanied by a change in past chunk throughput that is in the same direction as the bitrate switch. Thus, using both chunk size and observed throughput of previously downloaded chunks can improve the accuracy of bitrate estimation of a chunk.

Solution: We first estimate the bitrate of a chunk i using its size (S_i). If the estimated bitrate ($\hat{Q}_i$) is the same as the previous chunk's estimated bitrate ($\hat{Q}_{i-1}$), we keep this estimate and move to next chunk. However, if there is a switch in the estimate, we compare the download throughput observed for chunk $i-1$ and $i-2$, say T_{i-1} and T_{i-2}. We approve a change in bitrate if $|T_{i-1} - T_{i-2}| \geq |\hat{Q}_i - \hat{Q}_{i-1}|$ (a change in network throughput is detected) and $(T_{i-1} - T_{i-2}) \times (\hat{Q}_i - \hat{Q}_{i-1}) > 0$ (throughput changed in the same direction as bitrate switch). In case of a bitrate up-switch according to chunk size, we also check if T_{i-1} is greater than $\hat{Q}_i$. For the first two chunks, we just use the chunk size to estimate its bitrate as we do not have enough information about chunk throughput.

C. QoE Metrics Inference

Using the above approach for a session V, we get a sequence of video chunks along with *estimates* of the download start time (ST_i), download end time (ET_i), and bitrate ($\hat{Q}_i$) for every chunk i. Let N denote the number of chunks observed in the session and L be the chunk duration in seconds. QoE metrics are estimated from this information as follows.

Average bitrate: Average bitrate is estimated by taking an average of the estimated bitrates of chunks in the session.

$$\hat{BR} = \frac{\sum_{i=1}^{N} \hat{Q}_i}{N} \tag{1}$$

Re-buffering ratio: Intuitively, re-buffering time is estimated by keeping an account of video chunks that have been downloaded and the part of the video that should have been played so far. Let B_i denote the video buffer occupancy in seconds just before chunk i was downloaded. The re-buffering time between two consecutive chunk download times, ET_i and ET_{i-1}, is represented by b_i. Let j denote the index of chunk after which the playback resumed since last re-buffering event, and CTS denote the minimum number of chunks required in the buffer to start playback. In the beginning, $j = CTS$ and $b_k = 0$ for $k \leq CTS$ as the waiting time before video startup is considered as startup time by definition. For each subsequent chunk i, B_i is calculated as follows:

$$B_i = \max\big((i - 1 - j + CTS) \times L - \big(ET_i - ET_j\big), 0\big) \tag{2}$$

Here, $(i - 1 - j + CTS) \times L$ represents the video content that has been downloaded, and $ET_i - ET_j$ represents the total video that should have been played since the playback began last time. If $B_i > 0$, then $b_i = 0$ and we move to next chunk. Otherwise, re-buffering occured and is calculated as follows:

$$b_i = \big(ET_i - ET_j\big) - (i - 1 - j + CTS) \times L \tag{3}$$

In this case, video playback would begin after downloading CTS chunks. Thus, value of j is set to $i + CTS - 1$ and parameter b_k for chunk $k \in \{i+1, i+CTS-1\}$ is set as $ET_k - ET_{k-1}$. The remaining b_i values can be obtained in a similar way. Re-buffering ratio can be calculated as follows:

$$\hat{RR} = \frac{\sum_{k=1}^{N} b_k}{N \times L + \sum_{k=1}^{N} b_k}. \tag{4}$$

Bitrate switches: The number of bitrate switches are calculated by counting the total number of times the *estimated*

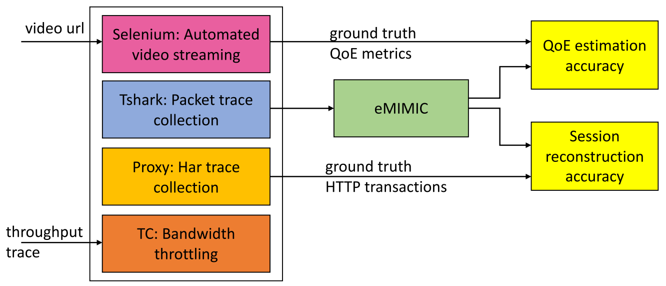

video url
Selenium: Automated video streaming
Tshark: Packet trace collection
Proxy: Har trace collection
throughput trace
TC: Bandwidth throttling
ground truth QoE metrics
eMIMIC
ground truth HTTP transactions
QoE estimation accuracy
Session reconstruction accuracy

Fig. 4. Experiment framework and evaluation methodology.

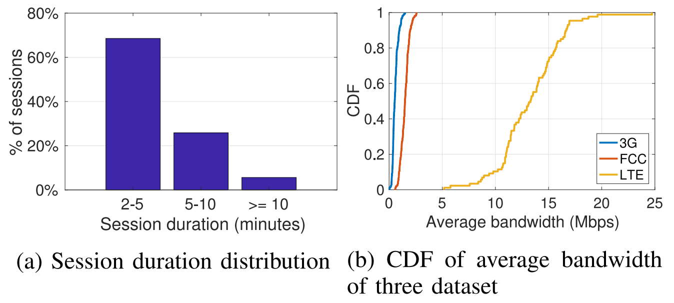


Fig. 5. Bandwidth traces and session duration.

chunk bitrate changed between consecutive chunks. We normalize this number by the total video streamed in minutes and estimate bitrate Switches Per Minute (*SPM*).

$$\hat{SPM} = \frac{\sum_{i=2}^{N} I\left(\hat{Q}_i \neq \hat{Q}_{i-1}\right) \times 60}{N \times L} \tag{5}$$

Here I is the indicator function which equals one if the consecutive chunks do not have same bitrate, zero otherwise.

Startup time: We use the time taken to download minimum number of chunks to begin playback, denoted by *TTNC* as a proxy for startup time. Note that normally startup time is defined as the time taken to play the video from the time user opened the video and constitues of following delays:

$$ST = T_{loading} + TTNC + T_{decode} \tag{6}$$

Here, $T_{loading}$ is the time to prepare the video, including delays like rights management. T_{decode} is time to decode and render the downloaded chunks on screen. $T_{loading}$ and T_{decode} are mostly application induced, while *TTNC* depends on the network. An operator would like to monitor only the network contribution (*TTNC*) to startup time since improving the network does not directly impact the other two delays. Therefore, we use *TTNC* as a proxy for startup time.

V. Evaluation

We first evaluate eMIMIC over two popular VoD services. More specifically, we consider the following in our evaluation: i) accuracy of HTTP request reconstruction, ii) accuracy of media type classification, and iii) accuracy of QoE metrics estimation. This is followed by a comparison of eMIMIC with a recently proposed ML-based approach (ML16). We then validate eMIMIC over a Live streaming service, and finally evaluate the accuracy of eMIMIC in estimating the QoE metrics in real-time. We begin by describing our experimental setup.

A. Experimental Setup

We build an automated browser-based framework that streams video sessions of a video service in a Web browser under emulated network conditions and collects packet traces, HTTP traces and ground truth video QoE metrics (see Figure 4). We use Java implementation of a popular browser automation framework, known as *Selenium*.[1] The HTTP logs of encrypted sessions are collected using a trusted proxy, *BrowserMob proxy*,[2] that is easy to integrate with *Selenium*. We use *TShark*[3] for capturing packet-level network traffic and Linux Traffic Control (*tc*) to emulate different network conditions.

Video sessions: We use two popular premium video services that stream Video on Demand (VoD). VoD1 streams primarily full-length movies, with some TV show selection, offering content in many countries world-wide. VoD2 is a U.S. VoD service offering primarily popular TV shows, including also full-length movies. Both services are available on most mobile and desktop devices, with up to 1080p video resolutions. Evaluating with two different video services helps in understanding the impact of differences in service design parameters on the accuracy of eMIMIC. We collected URIs of 100 videos each from both services, covering different genres such as animated videos, talk shows and action movies. The intent was to capture a diversity of content complexities and encoding bitrates. The duration of each session is pre-determined based on a distribution obtained from the video network dataset collected in [10] and is shown in Figure 5a. The distribution ranges from 2 to 20 minutes with a mean of 5 minutes.

Bandwidth traces: We use the following throughput traces to evaluate eMIMIC under realistic network conditions:

- Norway 3G dataset [33] consists of per-second throughput measurements from mobile devices streaming videos while connected to a 3G/HSDPA network.
- Belgium LTE dataset [34] is similar to Norway 3G but the network is LTE, resulting in higher throughput.
- FCC dataset [35] consists of per-5 seconds throughput measurements of broadband networks. We sample traces from this dataset with the same end-points and an average throughput under 3 Mbps to induce bitrate switching and make it more challenging to estimate QoE metrics.

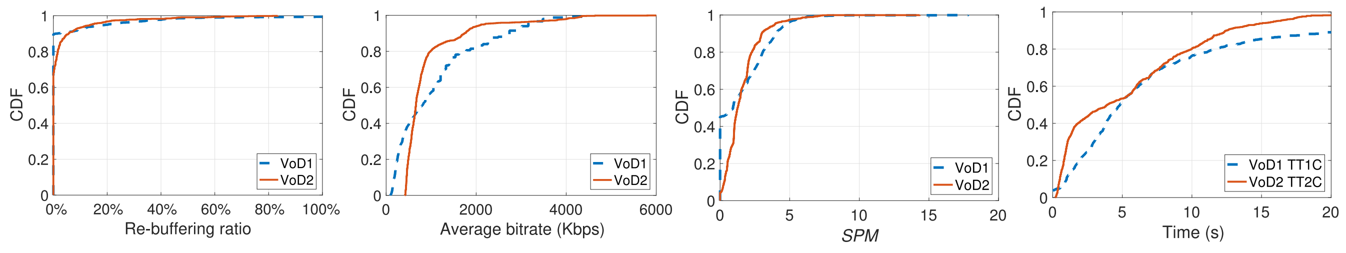
Figure 5b shows the CDF of average bandwidth of these traces.

Ground truth QoE metrics: These metrics in video streaming are available within the video player itself. We monitor the player buffer using the JavaScript API exposed by the *Video* element of the HTML5 MSE-based video players of these services. We found two functions, `buffered` and `played`, that return the range of video content that has been buffered and played, respectively. Calling them together enables us to

[1]www.seleniumhq.org

[2]bmp.lightbody.net

[3]www.wireshark.org/docs/man-pages/tshark.html



(a) Ground truth re-buffering ratio (b) Ground truth average bitrate (c) Ground truth number of bitrate switches per minute (d) Ground truth $TTNC$

Fig. 6. CDF of ground truth QoE metrics.

TABLE I
DESIGN PARAMETERS OF VOD1 AND VOD2

Design parameter	Audio		Video	
	VOD1	VOD2	VOD1	VOD2
Bitrate levels	2	1	10	7
Bitrate range (kbps)	64 - 96	64	100 - 4000	400 - 6000
Chunk duration (s)	16	5	4	5
Chunks to start	1	1	2	1

infer the size of buffer. However, it still does not give any information about other QoE metrics such as video bitrate.

We then explore the APIs available in the minified JavaScript source of the video players of the two video services. We found a function for VOD1, which when called returns the size of the buffered content in seconds and bytes, bitrate of the currently playing video and a boolean variable indicating if the playback is currently stalled. In our testing framework, we insert per-second calls to this function. Similarly, for VOD2 we found a function which closes the video playback and returns a session-summary of all the video QoE metrics, including average bitrate, re-buffering duration, number of bitrate switches and time taken to download the first chunk (*TT1C*). We insert a call to this function in our experiments at the end of the video session. Thus, by hooking into the functions of these players, we can obtain per-second ground truth QoE metrics for VOD1 and per-session ground truth QoE metrics for VOD2.

Obtaining video service design parameters: eMIMIC needs to know a few design parameters of a video service. The chunk duration is estimated by playing several video sessions completely and determining the number of chunks downloaded from HTTP logs. Video play time divided by the number of chunks gives average chunk duration. The bitrate levels for VOD2 are obtained by inspecting the *manifest* of few videos. VOD1 uses different bitrate levels across videos. As getting per-video bitrate levels is infeasible, we use approximate levels obtained by averaging bitrate levels observed for multiple videos. The number of chunks required to start (*CTS*) playing is obtained by inspecting the *manifest* for VOD2. For VOD1, we streamed several video sessions and collected the ground truth QoE metrics using the methodology described above. Using these metrics, we found that *CTS* varied but was always greater than 2, which we assume as *CTS* for VOD1. Table I summarizes the values of these design parameters. We note that these design parameters are prone to change for a service which can impact eMIMIC performance. In future, we plan to devise methods to automatically detect these changes.

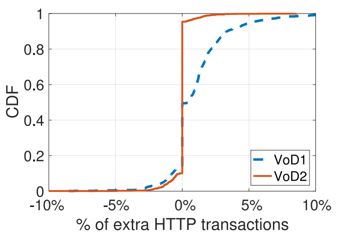
Based on the obtained (or inferred, if needed) design parameters, we set S_{min} to 35 KB and T_{ahead} to 40s for both services. We use two ranges, i.e., [126 KB, 136 KB] and [190 KB, 200 KB], and a single range, i.e., [40 KB, 50 KB] for identifying audio chunks in VOD1 and VOD2, respectively. We currently infer the video service design parameters and the corresponding eMIMIC parameters manually. Our future work will explore methods to automate this process so that any changes in the video design parameters can be detected and accommodated automatically.

Experiment: We use our testbed to stream video sessions from both VOD1 and VOD2 in Firefox. The bandwidth conditions in a session are emulated based on a trace selected randomly from the set of bandwidth traces. The packet traces, HTTP logs, and ground truth QoE metrics collected using the testbed are stored after the end of the session. In total, we ran 985 sessions for VOD1 and 1005 sessions for VOD2. Figure 6 shows the CDF of different ground truth QoE metrics for these sessions.

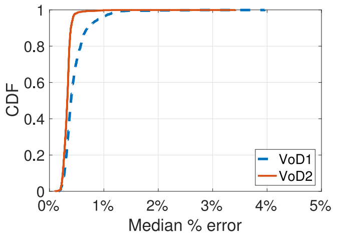
B. Session Reconstruction Accuracy

We first evaluate the accuracy of eMIMIC in reconstructing HTTP transactions corresponding to audio and video in a session. We filter out transactions less than S_{min} from the reconstructed HTTP transactions. We then match the remaining transactions with the ground truth HTTP logs corresponding to audio and video collected using trusted proxy. The matching process works as follows: for every reconstructed HTTP transaction of size greater than S_{min}, we search for an HTTP log in the corresponding proxy logs which has a start time within 500 milliseconds of the start time of the reconstructed transaction. If a matching log is found, we consider it as true transaction and remove the ground truth HTTP log. If there are multiple matches found, we use the one closest in size to the reconstructed log's size. After this matching process is finished, the unmatched reconstructed HTTP transactions are tagged as extra, and the unmatched ground truth HTTP logs are tagged as missing transactions.

Figure 7a shows a CDF of percentage of extra transactions (negative value denotes missing transactions) in a session. We find that the accuracy of reconstruction is high with 80% of sessions from VOD2 reconstructed with 100% accuracy. The lower accuracy of reconstruction for VOD1 is because few

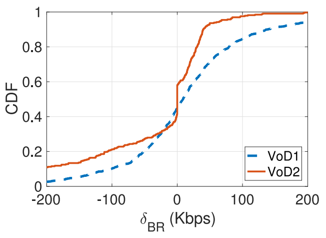


(a) CDF of % of extra requests in the reconstruction

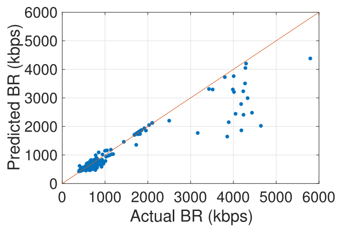


(b) CDF of median % error in estimated size of requests

Fig. 7. HTTP request reconstruction accuracy.



(a) CDF of error in average bitrate estimation



(b) VoD2: scatter plots of ground truth and estimated average bitrate

Fig. 8. Error in average bitrate estimation.

TABLE II
MEDIA CLASSIFICATION CONFUSION MATRIX FOR VOD1

actual	predicted	
	audio	video
audio	99.2%	0.8%
video	1.1%	98.9%

(a) With A/V buffer tracking

actual	predicted	
	audio	video
audio	99.3%	0.7%
video	2.4%	97.6%

(b) Without A/V buffer tracking

metadata transactions in VoD1 are comparable in size to *video* and get misclassified as *video*.

Figure 7b shows the CDF of median percentage error in the estimated size of reconstructed transactions. Note that it is important to accurately estimate the size of transaction as it is used to identify video chunks and their bitrates. The median error is within 1% of the actual size of HTTP transaction for both VoD1 and VoD2 which suggests that eMIMIC can estimate the size of HTTP transactions accurately.

C. Media Type Classification Accuracy

Table IIa shows the confusion matrix of audio/video (A/V) classification of the reconstructed HTTP transactions for VoD1. The ground truth was obtained by inspecting the request URI of HTTP logs collected by the proxy. The overall accuracy of classification is high (99.15%). The classification error is mainly due to two reasons: i) small video chunks in the range of expected audio chunk size get misclassified as audio ii) errors in estimated size of reconstructed audio chunk leads to audio chunk misclassified as video. The results are similar for VoD2 (omitted due to lack of space).

We also show the confusion matrix (Table IIb) when the A/V classification is done only using the size of the HTTP transaction. Tracking A/V buffer (Table IIa) helps in reducing the error of misclassifying video chunks as audio by 1.26% without significantly impacting the error in misclassifying audio chunks as video.

D. QoE Inference Accuracy

Here, we present the comparison of QoE metrics estimated by eMIMIC with ground truth QoE metrics.

Average bitrate: Figure 8a shows the CDF of difference in estimated and ground truth average bitrate, denoted by δ_{BR}, for VoD1 and VoD2. We see that eMIMIC accurately predicts average bitrate within an error of 100 kbps for 75% sessions in VoD1 and 80% sessions in VoD2. The error is in fact zero for nearly 20% sessions in VoD2. We do not observe

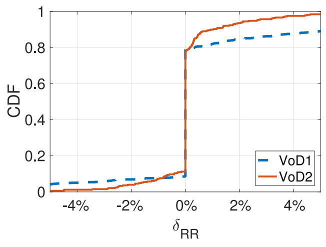


Fig. 9. CDF of error in re-buffering ratio estimation.

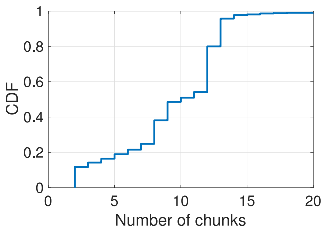


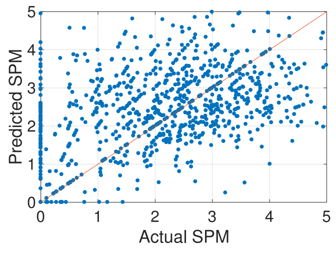
Fig. 10. CDF of chunks in the buffer at startup.

zero error in VoD1 partially because we do not know the exact values of bitrate levels and use approximate values instead.

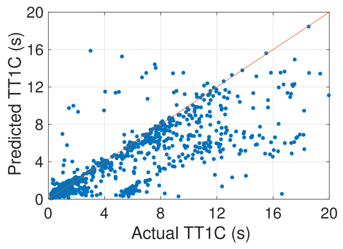
Figure 8b shows a scatter plot of ground truth average bitrate and estimated average bitrate for VoD2 sessions. The points are close to the identity line in most cases except at higher bitrates (around 4 Mbps). We found this is because of eMIMIC underestimating chunks with bitrate 3.2 Mbps and 6 Mbps due to higher variation in the chunk sizes in this range. Nevertheless, these are still estimated as more than 2 Mbps, which would be considered *high* bitrate for most purposes, if used for categorical classification.

Re-buffering ratio: We calculate the difference (δ_{RR}) between the estimated re-buffering ratio and ground truth re-buffering ratio. Figure 9 shows a CDF of δ_{RR} for VoD1 and VoD2. We see that eMIMIC can predict re-buffering ratio with a high overall accuracy, i.e., within an error of 1% for around 70% sessions in VoD1 and 65% sessions in VoD2.

We observe heavy-tails in δ_{RR} distribution for VoD1. On closer inspection, we found this has to do with an unusual buffering behavior in VoD1 player. The player would not begin a session even if it had video (and audio) chunks in its buffer. Figure 10 shows the CDF of number of video chunks in player buffer when the playback first started. The player sometimes waits until it has 12 chunks (48*s* video) in its buffer before starting video playback. Similar behavior was also seen



(a) VoD1: Scatter plot of SPM



(b) VoD2: Scatter plot of TT1C

Fig. 11. Error in estimating the bitrate switches and startup time.

when re-buffering event happened. This leads to errors in estimating re-buffering ratio since we assume that playback begins as soon as player receives a fixed number of chunks (two in this case) in its buffer.

Bitrate switches: Figure 11a shows a scatter plot of ground truth and estimated *SPM* for VoD1. We find that eMIMIC does not estimate *SPM* with high accuracy. This is because accurate bitrate switch estimation requires accurate estimate of bitrate of every video chunk in a session. Even a single wrong bitrate estimation of chunk can lead to significant errors in SPM estimation. We plan to explore alternate methods of bitrate switch estimation in our future work.

Startup time: Figure 11b shows a scatter plot of ground truth and estimated *TT1C* for VoD2. For most sessions, the network estimated *TT1C* is somewhat smaller than ground truth *TT1C* obtained from the player. This underestimation has been discussed in a previous study [9] and is mainly because the players experience additional network and operating system delays before they receive a chunk. Overall, eMIMIC shows high accuracy. It can predict startup delay within 2 seconds of ground truth for 65% sessions in VoD1 and 70% sessions in VoD2.

E. Comparison With ML-Based Approach

Here, we compare eMIMIC with ML16, an ML-based approach described by Dimopoulos *et al.* [7]. We use this approach for comparison because it gives categorical estimates of individual video metrics namely re-buffering ratio and average bitrate as opposed to other ML-based approaches that estimate overall QoE class assuming a specific model. The approach trains a Random Forest model using network QoS metrics such as round trip time and packet loss and chunk statistics such as size and download time. We implement ML16 using the scikit-learn library [36] in Python. We use 67% of our collected data for training the machine learning model and use remaining 33% for testing both ML16 and eMIMIC. We balance the QoE metric classes while training using a popular oversampling algorithm [37].

Average bitrate: We use three categories for average bitrate estimation. For VoD2, average bitrate is classified as *low* if $BR < 800$ kbps, *med* if $BR \in [800 \text{ kbps}, 2000 \text{ kbps}]$, and *high* otherwise. The *low* bitrate category corresponds to the two lowest bitrates, *med* to the next two bitrates and *high* to the top two bitrates. Similarly, thresholds of 600 kbps and 1400 kbps are chosen to classify sessions of VoD1 into *low*, *med* and *high*. The overall classification accuracy of eMIMIC is slightly higher (around 3%) than ML16 (row 1 of Table III). Table IV shows the confusion matrix of bitrate classification for VoD1. eMIMIC identifies *low* and *med* sessions with a higher accuracy, 2% and 7% respectively, than ML16.

TABLE III
CLASSIFICATION ACCURACY OF eMIMIC AND ML16

QoE metric	Classification accuracy			
	VoD1		VoD2	
	eMIMIC	ML16	eMIMIC	ML16
Average bitrate	87.8%	84.5%	93.6%	90.8%
Re-buffering ratio	80.5%	71.7%	85.1%	61.3%

TABLE IV
CONFUSION MATRIX: VoD1 AVERAGE BITRATE

(a) eMIMIC

actual	predicted *BR*		
BR	low	med	high
low	91.9%	8.1%	0.0%
med	12.2%	82.7%	5.1%
high	0.0%	15.2%	84.8%

(b) ML16

actual	predicted *BR*		
BR	low	med	high
low	89.4%	8.8%	1.8%
med	17.4%	75.5%	7.1%
high	0.0%	13.0%	87.0%

TABLE V
CONFUSION MATRIX: VoD2 RE-BUFFERING RATIO

(a) eMIMIC

actual	predicted *RR*		
RR	zero	mild	high
zero	87.6%	12%	0.4%
mild	51.5%	44.9%	3.6%
high	3.1%	8.4%	88.4%

(b) ML16

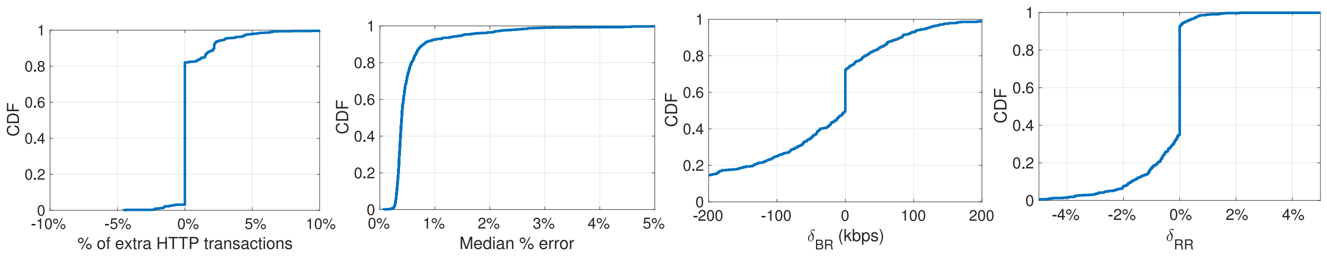
actual	predicted *RR*		
RR	zero	mild	high
zero	61.1%	37.0%	1.9%
mild	39.4%	48.5%	12.1%
high	26.9%	30.8%	42.3%

Re-buffering ratio: For estimating re-buffering using ML16, a video is categorized into one of the following three categories (same as in [7]): *zero* stall when there is no re-buffering, *mild* stalls when $0 < RR \leq 10\%$, and *high* stalls when $RR > 10\%$. ML16 was trained separately for both VoD1 and VoD2. Row 2 in Table III shows the re-buffering ratio classification accuracy of eMIMIC and ML16 over the test data. eMIMIC can estimate re-buffering ratio with significantly higher accuracy (10%-25%) than ML16. Table V shows the confusion matrix for re-buffering classification of VoD2. eMIMIC can predict *low* and *high* stalls with much higher accuracy than ML16. The accuracy of ML16 may improve with more training data.

Finally, we test if ML16 generalizes across services by using the ML16 model learned for VoD2 to estimate re-buffering ratio for VoD1. The classification accuracy of the model dropped to 31% on VoD1 from 61% on VoD2. This shows that ML16 does not generalize and needs separate training for each service whereas eMIMIC faces no such issues.

F. QoE Inference Accuracy for a Live Service

Live streaming has been growing over the last few years [38]. Live video differs from VoD in terms of few key design parameters, i) the buffer in Live streaming is small to reduce the latency to the live broadcast. As a result video players in Live may react more aggressively to changing network conditions leading to more bitrate switches. ii) At the same time. Live streams are not as efficiently encoded, which may



(a) CDF of % extra requests in the reconstruction (b) CDF of median % error in the estimated size of requests (c) CDF of error in average bitrate estimation (d) CDF of error in re-buffering ratio estimation

Fig. 12. LIVE1: Session reconstruction and QoE metrics estimation error.

TABLE VI
DESIGN PARAMETERS OF LIVE1

Design parameter	Audio	Video
Bitrate levels	3	6
Bitrate range (kbps)	48 - 96	160 - 2232
Chunk duration (s)	6	6
Chunks to start	1	1

lead to better bitrate estimation, due to less variability in segment sizes. These design differences make it important to understand the accuracy of eMIMIC in estimating QoE metrics for Live streaming. We use a popular subscription-based Live streaming service, referred to as LIVE1 here. LIVE1 delivers content from popular cable channels over the Internet. Table VI shows the design parameters for LIVE1, obtained by the methodology described in Section V-A. Based on these design parameters, we set S_{min} to 30 KB and use three ranges to identify audio chunks, i.e., [34 KB, 42 KB], [46 KB, 54 KB], and [70 KB, 76 KB]. We set T_{ahead} to 12 seconds as we found that the typical buffer size for the service is close to 12 seconds.

For obtaining ground truth QoE metrics in LIVE1, we found a combination of keystrokes, which when pressed, displays a box showing the different metrics such as the current playback bitrate, available bitrates, and video stall time. We program our testing framework to collect this information every second. We then use the testing framework to stream 629 sessions of LIVE1 under variety of network conditions and collect the corresponding packet traces, HTTP logs, and ground truth QoE metrics.

Session reconstruction accuracy: Figure 12a shows the CDF of percentage of extra transactions (calculation methodology described in Section V-B) in a session. The accuracy of reconstruction is high in general with around nearly 80% of sessions reconstructed with 100% accuracy. We do find few extra transactions in some sessions which are mainly because of two reasons: 1) *metadata* transactions that are comparable in size to *video* transactions get misclassified as *video*, and 2) HTTP aborts that happen more frequently in Live are not detected accurately. The estimated size of matched transactions is quite accurate with the median error within 1% of the actual size of HTTP transaction for nearly 90% of sessions (see Figure 12b).

QoE estimation accuracy: Figure 12c shows the CDF of difference in estimated and ground truth average bitrate (δ_{BR}). eMIMIC can accurately predict average bitrate within an error of 100 kbps for 75% sessions with zero error for nearly 20% sessions. The difference between the estimated re-buffering ratio and ground truth re-buffering ratio, denoted by δ_{RR}, is shown in Figure 12d. We see that eMIMIC can estimate re-buffering ratio within 1% of the ground truth for 80% of the sessions. We observe that eMIMIC tends to underestimate the re-buffering ratio for LIVE1 sessions. This can be attributed to errors in the session reconstruction step, where some non-video chunks are identified as video, leading to overestimation of video buffer, and hence underestimation of re-buffering.

G. Real-Time QoE Inference

Our evaluation, so far, has focused on understanding accuracy of eMIMIC in estimating QoE metrics over the entire session. An operator may want to infer video QoE metrics in real-time for QoE-aware active network resource management. For instance, the operator may temporarily boost the available bandwidth for sessions with low buffer occupancy in order to reduce the probability of re-buffering [39]. In this section, we evaluate the accuracy of eMIMIC in making such real-time prediction of QoE metrics. We limit our analysis to sessions from VOD1, as we have fine-granular ground truth QoE metrics only for VOD1.

Buffer occupancy: We evaluate the accuracy of eMIMIC in identifying *low* buffer occupancy conditions in a session. More specifically, for a window of T_{class} seconds within a session, the buffer occupancy is classified as *low*, if the buffer occupancy is lower than a threshold (C_{buff}) at any point of time in the window, and *high* otherwise. An operator could set different values for T_{class} and C_{buff} based on some policy.

We first evaluate the impact of varying T_{class}, using 20 seconds as the buffer occupancy threshold. Table VII shows the accuracy, precision, and recall values for different classification windows ranging from 5 seconds (short-term variations) to 1 minute (long-term degradation). We consider an instance to be a true positive if it is correctly identified as a *low* buffer occupancy instance. We observe that accuracy of correctly classifying buffer occupancy state is at least 90.0%, and it increases as the duration of classification window increases. The precision of classification is low (54.7% - 65.4%), while the recall is high (95.1% - 97.9%). This means that eMIMIC can correctly classify most of the *low* buffer occupancy instances, while a few *high* buffer occupancy instances are misclassified as *low*. Note that from a network

TABLE VII
IMPACT OF WINDOW ON BUFFER OCCUPANCY CLASSIFICATION, $C_{buff} = 20$ SECONDS

T_{class} (seconds)	Accuracy	Precision	Recall
5	90.0%	56.5%	97.9%
10	90.2%	61.2%	97.8%
20	90.7%	68.6%	97.6%
30	90.9%	73.7%	97.3%
40	91.3%	77.7%	98.0%
50	91.5%	80.4%	97.9%
60	91.8%	82.9%	97.8%

TABLE VIII
IMPACT OF THRESHOLD ON BUFFER OCCUPANCY CLASSIFICATION, $T_{class} = 10$ SECONDS

Low threshold (s)	Accuracy	Precision	Recall
5	91.3%	54.7%	95.1%
10	91.4%	59.2%	97.8%
15	90.9%	60.7%	97.7%
20	90.2%	61.2%	97.8%
25	90.2%	64.3%	97.8%
30	89.6%	65.4%	97.9%

TABLE IX
IMPACT OF NUMBER OF LAST DOWNLOADED CHUNKS ON BITRATE CLASSIFICATION, $C_{bitrate} = 600$ kbps

N_{class} (number of chunks)	Accuracy	Precision	Recall
2	85.7%	87.6%	88.9%
4	87.3%	89.3%	89.9%
6	88.0%	89.2%	90.9%
8	88.8%	89.9%	91.5%
10	88.6%	89.8%	91.5%
12	89.5%	90.1%	92.6%
14	89.4%	90.0%	92.4%
16	89.7%	89.6%	93.2%

TABLE X
IMPACT OF THRESHOLD ON BITRATE CLASSIFICATION, $N_{class} = 4$ CHUNKS

Bitrate threshold (kbps)	Accuracy	Precision	Recall
400	85.9%	85.5%	85.9%
600	87.3%	89.3%	89.9%
1200	90.9%	95.3%	93.6%

operator's perspective, it is more important to correctly identify all instances of low buffer occupancy (high recall), so that appropriate actions can be taken to reduce the probability of video re-buffering.

Similarly, we vary the buffer occupancy threshold (C_{buff}) for a fixed classification window of 10 seconds. The accuracy of classification decreases while precision and recall increase with increase in C_{buff} (see Table VIII). This is because both the number of *low* buffer occupancy instances correctly classified as *low* and the number of *high* buffer occupancy instances misclassified as *low* increase as the buffer occupancy threshold is increased.

Thus, eMIMIC estimates the buffer occupancy states with a high overall accuracy (89.6% - 91.8%). Moreover, it tends to underestimate buffer occupancy for VoD1 in general. This leads to a higher chance of misclassifying a *high* buffer occupancy state as *low*. However, it also leads to a high recall (95.1% - 98.0%), i.e., a higher probability of correctly detecting a *low* buffer occupancy state, which is desirable for an operator.

Average bitrate: Similar to buffer occupancy, an operator could allocate more resources to sessions downloading *low quality* of video chunks. We evaluate the accuracy of eMIMIC in classifying the average bitrate class of a fixed number of most recently downloaded chunks (N_{class}). More specifically, we classify the bitrate as *low* if the average bitrate of the last N_{class} downloaded chunks is lower than a threshold bitrate ($C_{bitrate}$) and *high*, otherwise. We study the effect of varying N_{class} and $C_{bitrate}$ on classification accuracy.

Table IX shows the accuracy, precision, and recall as N_{class} varies from 2 chunks (considered instantaneous quality) to 16 chunks (consider long-term quality) with a bitrate threshold of 600 kbps. Note that a classification instance is considered to be a true positive, if it has been correctly classified as a *low* average bitrate. The overall accuracy of classification is at least 85.7%. Furthermore, the accuracy increases with the increasing number of recently downloaded chunks considered for average bitrate classification. This is because using fewer chunks for classification is more sensitive to any errors in bitrate estimation of individual chunks as opposed to using more chunks. We also observe high precision and recall values, 87.6%-90.1% and 88.9%-93.2%, respectively. The recall values increases as N_{class} increases, thus leading to higher probability of identifying sessions with *low* bitrate chunks.

Similarly, we study the impact of varying the bitrate threshold on accuracy, while using the last 4 chunks for bitrate classification (see Table X). The classification accuracy is at least 85.9%. We also observe that all three metrics, i.e., accuracy, precision, and recall improve as bitrate threshold is increased. This is because, the difference in bitrates of chunks corresponding to lower video quality levels is smaller, and it increases with higher quality levels. Given that video chunks in VoD1 are VBR-encoded, there is a higher overlap in chunk sizes of lower bitrate levels than higher bitrate levels. Thus, errors in individual chunk bitrate estimation become smaller as the chunk quality increases, ultimately leading to an increase in classification accuracy when a *high* bitrate threshold is used.

This shows that eMIMIC can estimate average bitrate class of most recently downloaded chunks with high accuracy, precision, and recall.

VI. DISCUSSION AND FUTURE WORK

We discuss three outstanding issues pertaining to video QoE inference from passive network measurements.

A. Scalability

QoE inference approaches fundamentally require processing of network data. This network data can be enormous given the scale of today's networks, thus raising the need to design scalable inference systems.

One way to handle scalability is to leverage the trend of virtualization of network functions by operators [40]. Specifically, virtualization enables the design of flexible software-based network monitors that can be customized to meet the monitoring requirements of underlying inference approach. For instance, in the case of eMIMIC, network monitors can be designed that reconstruct the HTTP transactions in a flow instead of simply collecting and storing the entire packet traces for offline processing. This significantly reduces the storage and transport overhead of the collected data.

Sampling is another way to mitigate the issue of scaling by reducing the collected network data. The sampling of network data can be done in two ways. The first way is to sample video flows and monitor only a subset of video sessions instead of all sessions on the network. This can be useful if the goal is to understand and optimize the video performance in the network at a macro-level instead of optimizing per-user video performance. The challenge here is to determine the optimal sampling level such that the network data collected is minimized but it still provides enough information about the video performance at different network locations.

Another way to use sampling is to sample packets within a flow. However, packet sampling could potentially lead to the loss of critical information required for QoE inference. Therefore, appropriate packet sampling mechanisms need to be used based on the underlying QoE inference technique. For instance, eMIMIC could still work if packet sampling is used for downlink traffic while completely monitoring the uplink traffic. This is because eMIMIC uses the uplink traffic to identify the HTTP transaction boundaries and the downlink traffic to identify the size of the HTTP transactions. Sampling in uplink direction can lead to errors in identifying the HTTP transactions. However, the transaction size can still be determined from the sampled downlink traffic with reasonable accuracy.

Our future work will consider evaluating the usefulness of these techniques to implement a scalable QoE inference monitoring system.

B. QoE Inference for New Protocols

QoE inference approaches are typically designed for specific application and transport-layer protocols. However, these protocols constantly evolve, thus requiring continuous recalibration of the inference approach. For instance, eMIMIC has been designed for traditional HAS that uses HTTP over TCP. It uses TCP headers in the packets to reconstruct the HTTP transactions. However, TCP headers are no longer available in QUIC [41], a UDP-based protocol, requiring recalibration of eMIMIC. In our future work, we will explore using IP headers to reconstruct a session for video services using QUIC.

Another issue with new protocols such as QUIC and HTTP/2 [42] is that they allow request multiplexing. For eMIMIC, it can lead to error in session reconstruction as a new uplink packet is assumed as an indicator of end of the last HTTP transaction and beginning of a new transaction. However, it is not clear if the streaming services would use request multiplexing in practice as it leads to resource contention and reduced flexibility of bitrate adaptation (see Section IV-B1). One possible use case of multiplexing in streaming could be requesting the audio and video chunks for the same video segment in parallel. In our future work, we plan to characterize the multiplexing behavior of video streaming services that use these new protocols and adapt eMIMIC based on the observed behavior.

C. Impact of User Interaction

Existing QoE inference approaches, including eMIMIC, typically consider a linear video playback, i.e., there is no content *skip* or *pause* during the session. In practice, user interactions could be possible in a session and that can lead to errors in QoE inference. For instance, in the case of eMIMIC, video *skip* would lead to overestimation of the video buffer as it would not know that part of the video buffer would have been discarded due to the *skip*. Similarly, in the case of a video *pause*, eMIMIC would continue depleting the video buffer assuming linear playback leading to overestimation of re-buffering. Although an operator may not be as much concerned about video *pause* as about video *skip*, because it can miss a potential QoE impairment in the latter case.

One way to detect video *skip* in eMIMIC is by carefully monitoring the player buffer evolution. Video players typically have a fixed size (either number of bytes or duration) buffer. If eMIMIC's video buffer estimate at any point in the session is significantly higher than the maximum buffer size, there is a possibility that the user skipped part of the video and the operator can discard the session from QoE inference. Note that it still does not enable us to detect a *skip* in case the buffer level is lower than the *maximum* buffer in a session. Designing methods to detect and handle user interactions is a part of our future work.

VII. Conclusion

We present eMIMIC, a methodology to estimate QoE metrics of encrypted video using passive network measurements. To facilitate extensive evaluation, we develop an experimental framework that enables automated streaming and collection of network traces and ground truth QoE metrics of three popular video service providers, including both VoD and live content. Using the framework, we demonstrate that eMIMIC shows high accuracy of QoE metrics estimation for a variety of realistic network conditions. We compare eMIMIC with ML16, a machine learning-based approach and find that eMIMIC outperforms ML16 without requiring any training on ground truth QoE metrics. We finally show that eMIMIC can also be used for estimating QoE metrics in real-time with a high accuracy, thus enabling operators to detect any QoE degradation in the network.

References

[1] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "eMIMIC: Estimating HTTP-based video QoE metrics from encrypted network traffic," in *Proc. IEEE TMA*, 2018, pp. 1–8.

[2] A. E. Essaili, D. Schroeder, E. Steinbach, D. Staehle, and M. Shehada, "QoE-based traffic and resource management for adaptive HTTP video delivery in LTE," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 6, pp. 988–1001, Jun. 2005.

[3] A. Mansy, M. Fayed, and M. Ammar, "Network-layer fairness for adaptive video streams," in *Proc. IFIP Netw.*, 2015, pp. 1–9.

[4] *VoIP QoE*. Accessed: Mar. 2019. [Online]. Available: www.voip-info.org/wiki/view/QoS

[5] (2017). *Cisco VNI: Forecast and Methodology, 2016–2021*. Accessed: Mar. 2019. [Online]. Available: http://bit.ly/2xr5mva

[6] V. Aggarwal, E. Halepovic, J. Pang, S. Venkataraman, and H. Yan, "Prometheus: Toward quality-of-experience estimation for mobile apps from passive network measurements," in *Proc. ACM HotMobile*, 2014, Art. no. 18.

[7] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video QoE from encrypted traffic," in *Proc. ACM IMC*, 2016, pp. 513–526.

[8] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "A machine learning approach to classifying YouTube QoE based on encrypted network traffic," *Multimedia Tools Appl.*, vol. 76, no. 21, pp. 22267–22301, 2017.

[9] V. Krishnamoorthi, N. Carlsson, E. Halepovic, and E. Petajan, "BUFFEST: Predicting buffer conditions and real-time requirements of HTTP(S) adaptive streaming clients," in *Proc. ACM MMSys*, 2017, pp. 76–87.

[10] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "MIMIC: Using passive network measurements to estimate HTTP-based adaptive video QoE metrics," in *Proc. IEEE TMA/MNM*, 2017, pp. 1–6.

[11] T. Mangla *et al.*, "VideoNOC: Assessing video QoE for network operators using passive measurements," in *Proc. ACM MMSys*, 2018, pp. 101–112.

[12] (2018). *Video QoE Metrics*. [Online]. Available: https://mux.com/blog/the-four-elements-of-video-performance/

[13] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao, "Deriving and validating user experience model for DASH video streaming," *IEEE Trans. Broadcast.*, vol. 61, no. 4, pp. 651–665, Dec. 2015.

[14] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for Internet video," in *Proc. ACM SIGCOMM*, 2013, pp. 339–350.

[15] 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.

[16] T. T. T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 56–76, 4th Quart., 2008.

[17] W. M. Shbair, T. Cholez, J. Francois, and I. Chrisment, "A multi-level framework to identify HTTPS services," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, Apr. 2016, pp. 240–248.

[18] J. Garcia, T. Korhonen, R. Andersson, and F. Västlund, "Towards video flow classification at a million encrypted flows per second," in *Proc. IEEE 32nd Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, 2018, pp. 358–365.

[19] D. Tsilimantos, T. Karagkioules, and S. Valentin, "Classifying flows and buffer state for Youtube's HTTP adaptive streaming service in mobile networks," in *Proc. ACM Multimedia Syst. Conf. (MMSys)*, 2018, pp. 138–149.

[20] "The transport layer security (TLS) protocol version 1.2," Internet Eng. Task Force, Fremont, CA, USA, RFC 3954, Aug. 2008.

[21] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang, "A scheduling framework for adaptive video delivery over cellular networks," in *Proc. ACM MobiCom*, 2013, pp. 389–400.

[22] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and N. Shahmehri, "Helping hand or hidden hurdle: Proxy-assisted HTTP-based adaptive streaming performance," in *Proc. Int. Symp. Model. Anal. Simulat. Comput. Telecommun. Syst. (MASCOTS)*, San Francisco, CA, USA, 2013, pp. 182–191.

[23] S. Benno, J. O. Esteban, and I. Rimac, "Adaptive streaming: The network HAS to help," *Bell Labs Tech. J.*, vol. 16, no. 2, pp. 101–114, 2011.

[24] A. Kassler, L. Skorin-Kapov, O. Dobrijevic, M. Matijasevic, and P. Dely, "Towards QoE-driven multimedia service negotiation and path optimization with software defined networking," in *Proc. Softw. Telecommun. Comput. Netw. (SoftCOM)*, 2012, pp. 1–5.

[25] I. B. Mustafa, T. Nadeem, and E. Halepovic, "FlexStream: Towards flexible adaptive video streaming on end devices using extreme SDN," in *Proc. ACM Multimedia Conf. Multimedia Conf. (MM)*, 2018, pp. 555–563.

[26] K.-T. Chen, C.-C. Tu, and W.-C. Xiao, "OneClick: A framework for measuring network quality of experience," in *Proc. IEEE INFOCOM*, 2009, pp. 702–710.

[27] D. Joumblatt, J. Chandrashekar, B. Kveton, N. Taft, and R. Teixeira, "Predicting user dissatisfaction with Internet application performance at end-hosts," in *Proc. IEEE INFOCOM*, 2013, pp. 235–239.

[28] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for HTTPS and QUIC," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Honolulu, HI, USA, Apr. 2018, pp. 1331–1339.

[29] R. Schatz, T. Hoßfeld, and P. Casas, "Passive YouTube QoE monitoring for ISPs," in *Proc. IMIS*, 2012, pp. 358–364.

[30] G. Dimopoulos, P. Barlet-Ros, and J. Sanjuàs-Cuxart, "Analysis of YouTube user experience from passive measurements," in *Proc. CNSM*, 2013, pp. 260–267.

[31] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. ACM SIGCOMM CCR*, 2015, pp. 325–338.

[32] A. Hintz, "Fingerprinting websites using traffic analysis," in *Proc. PET*, 2003, pp. 171–178.

[33] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video streaming using a location-based bandwidth-lookup service for bitrate planning," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 8, no. 3, 2011, Art. no. 24.

[34] J. van der Hooft *et al.*, "HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks," *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2177–2180, Nov. 2016.

[35] (2017). *FCC Dataset*. [Online]. Available: https://www.fcc.gov/measuring-broadband-america

[36] (2018). *Scikit: Python Library*. [Online]. Available: scikit-learn.org/stable

[37] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.

[38] *Live Stream Growing as Big as VOD*. Accessed: Mar. 2019. [Online]. Available: https://www.muvi.com/blogs/live-stream-growing-big-vod.html

[39] V. Krishnamoorthi, N. Carlsson, and E. Halepovic, "Slow but steady: Cap-based client-network interaction for improved streaming experience," in *Proc. IEEE/ACM Int. Symp. Qual. Service (IWQoS)*, 2018, pp. 1–10.

[40] (2018). *ECOMP*. [Online]. Available: https://about.att.com/content/dam/snrdocs/ecomp.pdf

[41] J. Iyengar and M. Thomson, "QUIC: A UDP-based multiplexed and secure transport," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-quic-transport-16, Oct. 2018. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-16

[42] M. Belshe, R. Peon, and M. Thomson, "Hypertext transfer protocol version 2 (HTTP/2)," Internet Eng. Task Force, Fremont, CA, USA, Rep. 7540, 2015.

Tarun Mangla received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Delhi, in 2014, and the M.S. degree in computer science from the Georgia Institute of Technology in 2018, where he is currently pursuing the Ph.D. degree with the School of Computer Science, under the guidance of M. Ammar and E. Zegura. His research interests span video streaming, network measurements, and cellular networks. He is also interested in using computing for social good.

Emir Halepovic (M'03) received the Ph.D. degree from the University of Calgary, Canada, in 2010. He is a Principal Inventive Scientist with AT&T Labs Research. His interests are in the areas of networking, wireless, content delivery, video streaming, cross-layer interactions, quality of experience, and data analytics. He is building both network-scale platforms for measurement and analytics, as well as client and server side components for next generation video streaming. He is a member of the Association for Computing Machinery.

Mostafa Ammar (F'02) received the S.B. and S.M. degrees from the Massachusetts Institute of Technology and the Ph.D. degree from the University of Waterloo, ON, Canada. He is a Regents' Professor with the School of Computer Science, Georgia Institute of Technology, where he served as an Associate Chair of the School of Computer Science from 2006 to 2012. His research interests are in network architectures, protocols, and services. He has contributions in the areas of multicast communication and services, multimedia streaming, content distribution networks, network simulation, disruption-tolerant networks, network virtualization and mobile cloud computing, and network virtualization. He has published extensively in the above areas. He has served the networking research community in multiple roles. He served as the Editor-in-Chief for the IEEE/ACM TRANSACTIONS ON NETWORKING from 1999 to 2003. He was the Co-TPC Chair of the IEEE ICNP in 1997, ACM CoNEXT in 2006, and ACM SIGMETRICS in 2007 conferences. He was elected as a fellow of ACM in 2003.

Ellen Zegura (F'11) is the Stephen Fleming Professor of computer science with Georgia Tech and an Executive Faculty Co-Director of the Center for Serve Learn Sustain. Her research interests span computer networking and computing for social good. Recently, she has brought these together in a set of projects in collaboration with Elizabeth Belding at UC Santa Barbara to study and expand Internet access on Native American tribal lands. Her research has been funded by NSF, DARPA, the Army Research Lab, and Cisco. She was the First Chair of the School of Computer Science, Georgia Tech from 2005 to 2012. She is a fellow of ACM and the incoming Chair of the Computing Research Association Board.