eMIMIC: Estimating HTTP-based Video QoE Metrics from Encrypted Network Traffic

Tarun Mangla*, Emir Halepovic†, Mostafa Ammar*, Ellen Zegura*
*Georgia Institute of Technology †AT&T Labs – Research

***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 a 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 of two popular video streaming services collected using a lab testbed. eMIMIC estimates re-buffering ratio within 1 percentage point of ground truth for up to 70% sessions and average bitrate with error under 100 kbps for up to 80% sessions. 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.**

I. Introduction

Understanding the user-perceived Quality of Experience (QoE) is important for network operators, as it can help with efficient provisioning and management [1], [2]. 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 [3]. However, this can be challenging for HTTP-based Adaptive Streaming (HAS) video, a major contributor to network traffic [4], 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 [5], [6], [7], [8] 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 [1], [2].

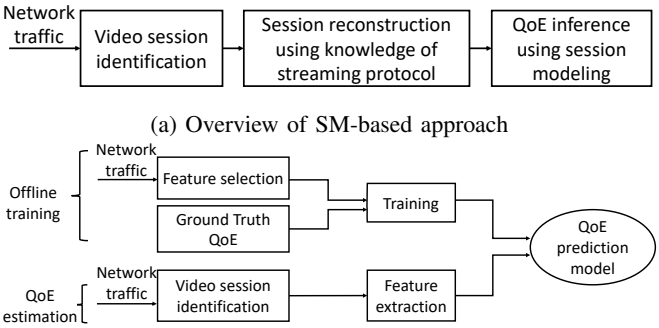
In prior work [9], we presented a highly accurate and practical QoE inference approach (with the system details presented in [10]) for HAS video, called MIMIC. MIMIC relied on extracting information from the application layer, i.e., Uniform Resource Identifiers (URIs) and other HTTP headers. With increasing 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.

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 [11]. An operator may need to further model the impact of these metrics taken together on the user experience, either through user studies [12] or objective data analysis [13], [14]. 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 two popular commercial video streaming services.

Furthermore, we replicate a recently proposed machine learning-based QoE estimation approach, hereon referred to as ML16 [6], by fully implementing and applying it to the same two video services. This helps in understanding the differences



(a) Overview of SM-based approach

(b) Overview of ML-based approach

Fig. 1: Overview of QoE inference approaches

in performance and accuracy between the two QoE estimation approaches, so that they can be further evolved and improved.

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 two 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 70% of video sessions, and average bitrate with error under 100 kbps for up to 80% of sessions.
- We compare eMIMIC with ML16 [6] 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.

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 progressive download, works by Schatz et al. [15] and Dimopoulos et al. [16] estimate video re-buffering by inspecting the packet traces and HTTP logs, respectively. 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 [9].

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 [5], [6], [7], [8].

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 [5], [7], logging unencrypted versions of the traffic [6] or using a trusted proxy [8]. 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 IV-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 [1], [2], 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.

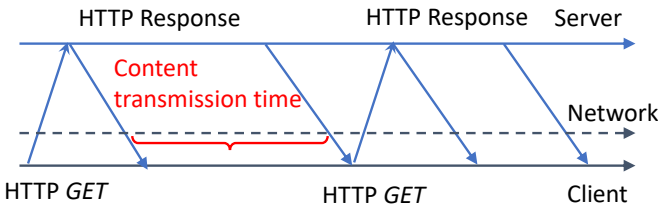


Fig. 2: Data flow of chunk requests and responses

III. 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.

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 [17], 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 [18].

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.

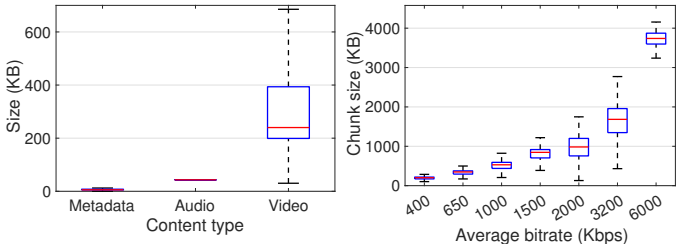
Solution: For a TCP-flow f corresponding to video session **V**, we log the *source IP* address of every packet in the flow. A packet with non-zero payload size is tagged as an HTTP request if the *source IP* address matches the client IP address. The subsequent non-zero payload size packets in f with 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: 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 IV 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



(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 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 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 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.

$$\hat{BR} = \frac{\sum_{i=1}^{N} \hat{Q}_i}{N} \quad (1)$$

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:

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

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:

$$b_i = (ET_i - ET_j) - (i-1-j+CTS) \times L \quad (3)$$

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:

$$\hat{RR} = \frac{\sum_{k=1}^{N} b_k}{N \times L + \sum_{k=1}^{N} b_k} \quad (4)$$

Bitrate switches: The number of bitrate switches are calculated by counting the total number of times the *estimated* chunk bitrate changed between consecutive chunks. We normalize this number by the total video streamed in minutes and estimate bitrate Switches Per Minute (SPM).

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

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:

$$ST = T_{loading} + TTNC + T_{decode} \quad (6)$$

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.

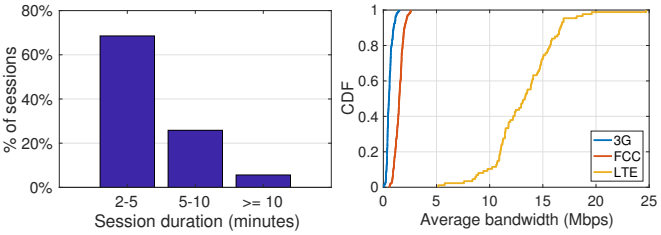
IV. Evaluation

Our evaluation is divided into the following four sections: i) Accuracy of HTTP request reconstruction, ii) Accuracy of media type classification, iii) Accuracy of QoE metrics estimation, and iv) Comparison of eMIMIC with ML16. We first describe 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. 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 [9] and is shown in Figure 4a. The distribution ranges from 2 to 20 minutes with a mean of 5 minutes.



(a) Session duration distribution (b) CDF of average bandwidth of three dataset

Fig. 4: Bandwidth traces and session duration

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

- Norway 3G dataset [19] consists of per-second throughput measurements from mobile devices streaming videos while connected to a 3G/HSDPA network.
- Belgium LTE dataset [20] is similar to Norway 3G but the network is LTE, resulting in higher throughput.
- FCC dataset [21] 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 4b 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 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

[1] www.seleniumhq.org
[2] bmp.lightbody.net
[3] www.wireshark.org/docs/man-pages/tshark.html

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

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.

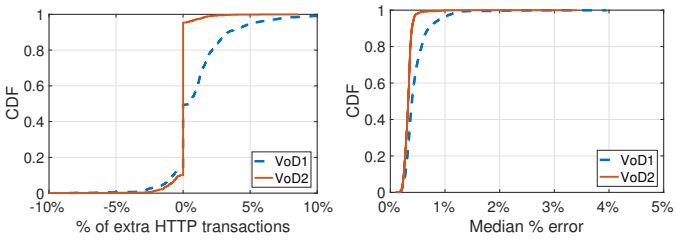
Obtaining video service design parameters: eMIMIC needs to know a few design parameters of a video service. The chunk duration is estimated by playing several video sessions completely and determining the number of chunks downloaded from HTTP logs. Video play time divided by the number of chunks gives 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: [126 KB, 136 KB] and [190 KB, 200 KB], and a single range: [40 KB, 50 KB] for identifying audio chunks in VoD1 and VoD2, respectively.

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.

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.



(a) CDF of % of extra requests in the reconstruction (b) CDF of median % error in estimated size of requests

Fig. 5: HTTP request reconstruction accuracy

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

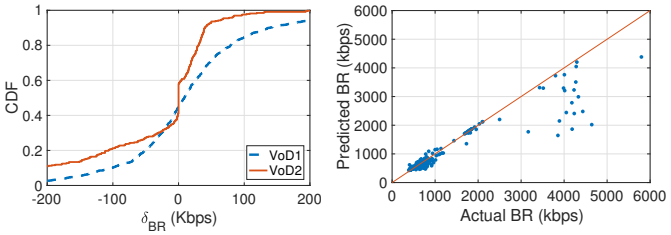
Figure 5a shows a CDF of percentage of extra transactions (negative value denotes missing transactions) in a session. We find that the accuracy of reconstruction is high with 80% of sessions from VoD2 reconstructed with 100% accuracy. The lower accuracy of reconstruction for VoD1 is because few *metadata* transactions in VoD1 are comparable in size to *video* and get misclassified as *video*.

Figure 5b 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.



(a) CDF of error in average bitrate estimation (b) VoD2: scatter plots of ground truth and estimated average bitrate

Fig. 6: Error in average bitrate estimation

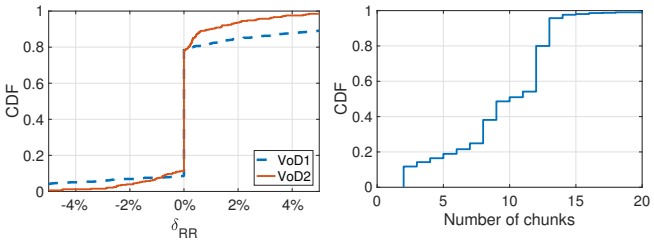


Fig. 7: CDF of error in re-buffering ratio estimation

Fig. 8: CDF of of chunks in the buffer at startup

D. QoE inference accuracy

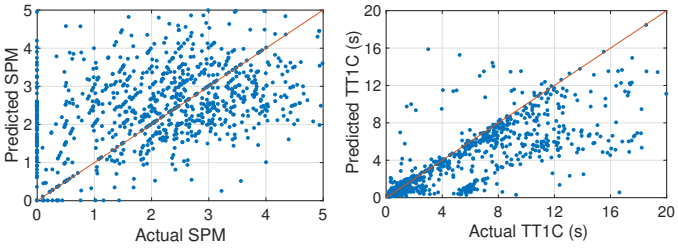
Here, we present the comparison of QoE metrics estimated by eMIMIC with ground truth QoE metrics.

Average bitrate: Figure 6a 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 zero error in VoD1 partially because we do not know the exact values of bitrate levels and use approximate values instead.

Figure 6b 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 7 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 8 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 ($48s$ video) in its buffer before starting video playback. Similar behavior was also seen 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.



(a) VoD1: Scatter plot of SPM (b) VoD2: Scatter plot of TT1C

Fig. 9: Error in estimating the bitrate switches and startup time

Bitrate switches: Figure 9a 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 9b 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 [8] 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. [6]. 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 [22] 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 [23].

Average bitrate: We use three categories for average bitrate estimation. For VoD2, average bitrate is classified as *low* if $BR < 800$ kbps, *med* if $BR \in [800$ kbps, 2000 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*,

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 *BR*	predicted *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 *BR*	predicted *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%

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.

Re-buffering ratio: For estimating re-buffering using ML16, a video is categorized into one of the following three categories (same as in [6]): *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.

V. 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 two popular video service providers. 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.

In future work, we plan to improve eMIMIC by automating the inference of video service design parameters, adapt it to new protocols like QUIC, and understand the impact of user interactions such as fast-forward and rewind on the estimation accuracy. Another goal is to devise methods for an operator to use the QoE estimation by eMIMIC for network management.

TABLE V: Confusion matrix: VoD2 re-buffering ratio

(a) eMIMIC

actual *RR*	predicted *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 *RR*	predicted *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%

Acknowledgments

This work is funded in part by NSF grant NETS 1409589.

References

[1] 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 TCSVT*, 2015.

[2] A. Mansy, M. Fayed, and M. Ammar, "Network-layer fairness for adaptive video streams," in *Proc. of IFIP Networking*, 2015.

[3] "VoIP QoE." [Online]. Available: www.voip-info.org/wiki/view/QoS

[4] "Cisco VNI: Forecast and methodology, 2016-2021," 2017.

[5] 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.

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

[7] 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 and applications*, 2017.

[8] 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.

[9] 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.

[10] T. Mangla, E. Halepovic, R. Jana, K.-W. Hwang, M. Platania, M. Ammar, and E. Zegura, "VideoNOC: Assessing Video QoE for Network Operators using Passive Measurements," in *Proc. of ACM MMSys*, 2018.

[11] (2018) Video QoE metrics. [Online]. Available: https://mux.com/blog/the-four-elements-of-video-performance/

[12] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao, "Deriving and validating user experience model for DASH video streaming," *IEEE Transactions on Broadcasting*, 2015.

[13] 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. of ACM SIGCOMM*, 2013.

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

[15] R. Schatz, T. Hossfeld, and P. Casas, "Passive YouTube QoE monitoring for ISPs," in *Proc. IMIS*, 2012.

[16] G. Dimopoulos, P. Barlet-Ros, and J. Sanjus-Cuxart, "Analysis of YouTube user experience from passive measurements," in *Proc. CNSM*, 2013.

[17] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," *ACM SIGCOMM CCR*, 2015.

[18] A. Hintz, "Fingerprinting websites using traffic analysis," in *PET*, 2003.

[19] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video streaming using a location-based bandwidth-lookup service for bitrate planning," *ACM TOMCCAP*, 2011.

[20] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks," *IEEE Comm. Letters*, 2016.

[21] (2017) FCC dataset. [Online]. Available: https://www.fcc.gov/measuring-broadband-america

[22] (2018) Scikit: Python library. [Online]. Available: scikit-learn.org/stable

[23] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of artificial intelligence research*, 2002.