VIDEO QOE ESTIMATION USING NETWORK MEASUREMENT DATA

A Dissertation
Presented to
The Academic Faculty

By

Tarun Mangla

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Computer Science

Georgia Institute of Technology

August 2020

VIDEO QOE ESTIMATION USING NETWORK MEASUREMENT DATA

Approved by:

Dr. Mostafa Ammar, Co-advisor
School of Computer Science
Georgia Institute of Technology

Dr. Ellen Zegura, Co-advisor
School of Computer Science
Georgia Institute of Technology

Dr. Constantine Dovrolis
School of Computer Science
Georgia Institute of Technology

Dr. Ada Gavrilovska
School of Computer Science
Georgia Institute of Technology

Dr. Emir Halepovic
AT&T Labs Research

Date Presented: July 10, 2020

Is there a greater strength than that of Knowledge?

Swami Vivekananda

To my grandmother, parents, brother, and sister-in-law

for their unconditional love and support

ACKNOWLEDGEMENTS

As I look back into my six years of PhD journey, I feel thankful to a lot of people who I got the opportunity to know and learn from. The most important of them are my advisors, Mostafa Ammar and Ellen Zegura. I consider myself quite fortunate to have not one but two amazing advisors. I still remember the day when I, a new graduate student with little research experience, reached out to Mostafa to work with him. He kindly accepted me and introduced me to the world of Video Streaming. I am truly indebted to him for believing in me. I started working more closely with Ellen in my second semester when Mostafa was on a sabbatical. She has always been supportive of my ideas. I am also thankful to her for introducing me to the world of Computing and Society. Both Ellen and Mostafa have mentored me with utmost patience, let me explore problems I like, nudged me back when I lost direction, and shared wonderful nuggets of wisdom (technical and otherwise) that I will always remember. My heart-felt gratitude for their support, concern for my well being, and solid guidance. I will always look up to them as my role models.

I am thankful to Emir Halepovic who has been a wonderful mentor and coauthor for all of my thesis work. Working with him has been a great learning experience both at a professional and personal level. Emir has always been very calm and has a knack of finding teaching moments even in the midst of a deadline. Thanks Emir for hosting me twice at AT&T Labs and introducing me to some very interesting research problems.

I am also grateful to the other two members of my committee, Constantine Dovrolis and Ada Gavrilovska, whose constructive feedback has helped shaped this dissertation. The discussions with them have particularly helped me in articulating the technical contributions of the thesis in a better manner. Thanks Constantine for also mentoring me in my first semester.

I have been also fortunate to work with an amazing set of researchers through interns and projects beyond my thesis. I am thankful to Elizabeth Belding, Morgan Vigil-Hayes,

and Esther Showalter for a very joyful and insightful collaboration on LTE connectivity research. Thanks Venkat Padmanabhan for hosting me at Microsoft Research India. Venkat has always inspired me with his big picture thinking and hands-on technical rigor. I would also like to thank Ahmed Mansy and Partha Kanuparthy for hosting me at Yahoo at the end of my first year. Working with them was a very enriching experience. Ahmed had just graduated from our group before I joined and it is from him I inherited a lot of technical expertise on video streaming during my intern.

My Ph.D. journey has been enriched by my peers and faculty members from the NRG. I would like to specially mention Ahmed Saeed, who has been a great course partner, a great critic of ideas, and above all a great friend. I fondly remember our discussions on research and life over our almost-daily water break and once-in-a-while walk to the Student Center. Thanks also to his wife, Heba Kamal, for being a great host at multiple times and for the delightful conversations. Thanks to Karim Habak for the fun squash matches and his friendship and Yimeng Zhao for being a great sounding board and friend especially in the last year. I am also thankful to Long Gong, Liang Liu, Kaeser Sabrin, Payam Siyari, Kamal Shadi, Yifeng Cao, Abhinav Narain, Bilal Anwer, and Samantha Lo for being wonderful colleagues and labmates. A special thanks to Samantha Lo for also introducing me to Emir and being an amazing co-host at AT&T Labs.

I am also indebted to friends outside the lab who have added a lot of joy to my Ph.D. journey and provided great support. Thanks to Ankit and Pooja Singhal for being there when the going got tough. A special thanks to Sainyam and Ankush for being patient listeners but also tough friends when my rantings went overboard. I am thankful to Chirag Jain who is not only my brother's namesake but has also been like a brother, always there in happy and tough times. I am equally thankful to Abhiraj Sharma, Ishaan Batta, and Ashwin Lele for being wonderful roommates. A special thanks to Pradyumna Suresha for making dinners fun with his delightful food and humour, and Apaar Shanker for intellectually stimulating conversations and not-so-fruitful but fun times in the library. I would like

to thank both of them for also being my go-to ML experts. I owe gratitude to Siva, Ranjan, and Seshan for great discussions and their wonderful friendship. A huge thanks to all the members of Hindu YUVA family. They have been a wonderful emotional, intellectual, and spiritual support and I learnt a lot from each one of them.

I am also indebted to my alma mater – IIT Delhi. I formed great connections who motivated me to pursue Ph.D. and have continued to inspire me. I am particularly thankful to my B.Tech. thesis advisors, Amitabha Bagchi and Vinay Ribeiro, who introduced me to the world of research and networking. I am also thankful to Nomesh Bolia and Parag Singla who have been wonderful mentors and role models. My humble gratitude to the AINA family members who have significantly enriched my life.

Last but not the least, this thesis would not have been possible without the support of my family. I am indebted to my caring grandmother who, although has not gone through much of schooling herself, has inspired me with her raw genius and intellect. I would like to believe that whatever intelligence I have is inherited from her. I would also like to remember my grandfather who passed away during my Ph.D. but would always be thrilled and proud of smallest of my accomplishments. Thanks to my brother, who always lightens up the mood with his sense of humour and my sister-in-law who has been a great friend. I am also thankful to the sheer bundle of joy, my nephew, who was born during the last year of my Ph.D. My deepest gratitude to my father who has always made sure of my well-being, provided me with best of resources and care, has taught me some great life-lessons, and has been a source of my strength. I cannot thank my mother enough for her unconditional love and support. She has taught me to be sincere in my efforts without worrying much about the results. She has always been able to sense if things are going rough and has provided great words of wisdom during those times. This thesis is dedicated to my family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

More than even before, last-mile Internet Service Providers (ISPs) need to efficiently provision and manage their networks to meet the growing demand for Internet video (expected to be 82% of the global IP traffic in 2022). This network optimization requires ISPs to have an in-depth understanding of end-user video Quality of Experience (QoE). Understanding video QoE, however, is challenging for ISPs as they generally do not have access to applications at end user devices to observe key objective metrics impacting QoE. Instead, they have to rely on measurement of network traffic to estimate objective QoE metrics and use it for troubleshooting QoE issues. However, this can be challenging for HTTP-based Adaptive Streaming (HAS) video, the *de facto* standard for streaming over the Internet, because of the complex relationship between the network observable metrics and the video QoE metrics. This largely results from its robustness to short-term variations in the underlying network conditions due to the use of the video buffer and bitrate adaptation. In this thesis, we develop approaches that use network measurement to infer video QoE. In developing inference approaches, we provide a toolbox of techniques suitable for a diversity of streaming contexts as well as different types of network measurement data.

We first develop two approaches for QoE estimation that model video sessions based on the network traffic dynamics of the HAS protocol under two different streaming contexts. Our first approach, MIMIC, estimates unencrypted video QoE using HTTP logs. We do a large-scale validation of MIMIC using ground truth QoE metrics from a popular video streaming service. We also deploy MIMIC in a real-world cellular network and demonstrate some preliminary use cases of QoE estimation for ISPs. Our second approach is called eMIMIC that estimates QoE metrics for encrypted video using packet-level traces. We evaluate eMIMIC using an automated experimental framework under realistic network conditions and show that it outperforms state-of-the-art QoE estimation approaches.

Finally, we develop an approach to address the scalability challenges of QoE inference.

We leverage machine learning to infer QoE from coarse-granular but light-weight network data in the form of Transport Layer Security (TLS) transactions. We analyze the scalability and accuracy trade-off in using such data for inference. Our evaluation shows that that the TLS transaction data can be used for detecting video performance issues (*low* video quality or *high* re-buffering) with a reasonable accuracy and significantly lower computation overhead as compared to packet-level traces.

CHAPTER 1
INTRODUCTION

End-user application Quality of Experience (QoE) is a major driver for customer growth and business revenue in the Internet economy [1, 2]. A recent study by Akamai shows that a re-buffering of 1% in the Internet video can reduce the watch time by 5% [2]. Thus, providing an "exceptional" QoE is of high importance for all the stakeholders involved in content delivery over the Internet. While there are several kinds of content over the Internet, in this thesis, we focus on video and its QoE. This is because video dominates the Internet traffic (75% of the traffic in 2017 [3]) and continues to grow with the advent of bandwidth-hungry video streaming formats (e.g., *virtual reality*, *4K streaming*) and an increasing shift from traditional pay-TV provider to Internet video [4]. The growing demand for Internet video has made it imperative for all stakeholders in the video delivery ecosystem to adapt their network infrastructure and provide "exceptional" video QoE.

Due to the flattening of the Internet, most video over the Internet traverses only two networks, the content provider network and the Internet Service Provider (ISP) network where the end-user is connected (or end-user ISP for short). Thus, the responsibility for delivering the best possible video QoE to users is shared between these two networks. In some instances, the video content can be served from within the end-user ISP; e.g., Netflix [5] partners with end-user ISPs and embeds their servers inside the ISP network to serve video. In this case, it is mostly the end-user ISP's responsibility to insure the delivery of video QoE to users.

End-user ISPs, therefore, need an in-depth understanding of end-user video QoE and its relationship to network performance. This is a first step in the ISP's own network management in support of video QoE – for short term trouble shooting and problem mitigation [6], and for long term QoE-aware network planning and provisioning [7, 8]. Estimating video

QoE, however, is challenging for an ISP, since they typically do not have access to the video application on user devices, the device itself, or the server. Unlike content providers, they cannot use in-app plug-ins [9, 10] for measuring the video QoE. Recent work proposes an alternative networking paradigm in which the ISPs and the content providers collaborate, allowing the latter to share QoE information with the former [11, 12]. However, this approach requires significant effort and is not immediately realizable. End-user ISPs are, therefore, currently constrained to use data derived from within their network to estimate video QoE.

Interpreted literally, video QoE is "a measure of the delight or annoyance" of an end-user's video streaming experience [13]. Thus, measuring video QoE is difficult even for content providers, let alone ISPs, as user experience is subjective and hard to quantify. Recent literature and standardization efforts propose to characterize video QoE using multiple objective metrics such as *average bitrate* and *re-buffering ratio* [14, 15, 16, 17, 2]. The challenge before ISPs and hence *our focus in this thesis is to use network measurement data to estimate the objective video QoE metrics.*

Most of the video today is streamed using techniques that comprise HTTP client-server interactions and that adapt the video quality to network conditions [18, 19, 20, 21, 22, 23]. There are a number of variations of such techniques [24, 25, 26] which we will refer to collectively as *HTTP Adaptive Streaming (HAS)*. Unlike applications such as IP telephony where objective QoE metrics are directly reflected by observable network Quality of Service (QoS) metrics (e.g., packet delay, jitter), the relationship between QoE metrics and QoS metrics in HAS video is complex. This is mainly because of two factors, namely, bitrate adaptation and the use of video buffer. Both of these result in robustness to short-term variations in the underlying network QoS; hence, making it challenging to quantify the relationship between network QoS metrics and application QoE metrics. In this thesis, we design techniques that use network data to infer QoE metrics for HAS video.

In designing QoE inference approaches for HAS, we address several research chal-

lenges. One of the challenges is that there is a diversity in the *streaming context*. There are different video services available that although all are based on HAS, each of them use their own service design parameters. Similarly, there is diversity in the the end-user devices wherein a video can be streamed over a smartphone, desktop, or smart TV. The diverse *streaming contexts* translate to difference in the corresponding video QoE even under the same network condition. We design QoE inference techniques that are able to handle this diversity in *streaming contexts*. Another major challenge is that an increasing amount of video service providers are using end-to-end encryption. This severely limits the amount of information ISPs have access to within the network data. We design techniques that rely on this limited information to infer video QoE metrics. Finally, QoE estimation at a network-wide scale also poses scalability challenges. We consider the case wherein scalability issues arise at the point where the network measurements are taken. More specifically, it can be challenging and inefficient to collect very fine-granular network data such as packet traces for the entire network. An alternative approach then is to use more aggregate forms of network data for estimation. A major challenge with this data is that it is quite coarse-granular. In this thesis, we design an inference technique that uses readily-available, light-weight, but coarse-granular network data.

1.1 Primary Contributions

The goal of this thesis is to design a *toolbox* of QoE inference approaches suitable for a variety of streaming contexts as well as different granularity of network measurement data. The thesis statement is that *it is possible to estimate end-user video QoE metrics using different kinds of network measurement data, if the inference methods are appropriately designed based on the knowledge of HAS protocol and the corresponding traffic patterns on the network.* The accuracy of inference varies based on the granularity of the data collected on the network. For fine-granular network data, we design inference approaches, even for encrypted video, that *generalize* across video services with minimal additional

information about the service design parameters. The generalizability is achieved by using an approach called *Session Modeling* (SM). It utilizes the knowledge of the underlying streaming protocol which is common across services. For coarse-granular network data, we develop machine learning-based methods that can learn patterns from the network data annotated with ground truth QoE to identify *low* QoE sessions. We analyze the trade-off between scalability and accuracy of inference. Our analysis shows that aggregate forms of network data can enable a coarse-granular QoE estimation with reasonable accuracy. This is particularly useful in detecting video performance issues in a light-weight manner. An ISP can then collect fine-granular data and use the corresponding inference techniques for further troubleshooting.

More concretely, we develop three techniques for QoE estimation, each of them suitable for a different streaming context and network measurement data. We make the following contributions, also summarized in in § 1.2, § 1.3, and § 1.4:

- **Inference for unencrypted traffic:** We first develop an algorithm, called MIMIC, that estimates video QoE metrics such as *re-buffering* and *video quality* for unencrypted HTTP-based video. The algorithm is based on the insight that an HTTP-based video session can be modeled as a sequence of chunk downloads appearing as HTTP transactions on the network. We do a large-scale validation of our algorithm using ground truth QoE metrics from a popular video streaming service. We deploy MIMIC in a real-world cellular network and demonstrate some preliminary use cases of QoE estimation for ISPs such as understanding the current video demand and understanding the impact of network or application-layer changes on video QoE.

- **Inference for encrypted traffic:** We then develop eMIMIC, a methodology that uses passive measurements at network-layer to estimate QoE metrics of the encrypted video sessions. eMIMIC relies on information extracted from the TCP headers of the packet-level video traffic to reconstruct HTTP transactions and uses it for modeling the video session. We also 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 outperforms state-of-the-art QoE estimation approaches.

- **Inference using coarse-grained data:** We develop a machine learning-based method that infers QoE using coarse-grained network data in the form of Transport Later Security (TLS) transactions. We develop features from this coarse-granular network data based on the knowledge of HAS protocol. We also develop a session-identification heuristic based on the access patterns to video servers at the beginning of the session. We extensively evaluate the method over three popular streaming services. We compare the estimation accuracy with packet-level data and the associated computation and memory overhead.

1.2 Inference for unencrypted video traffic

In Chapter 3, we present an SM-based methodology, called MIMIC, for QoE estimation using unencrypted video traffic.

Idea: MIMIC is designed based on the the HAS protocol. In HAS, the video is divided into *chunks* usually of equal playback duration, with each *chunk* encoded at multiple bitrate levels chosen from a pre-defined set. When the client opens a HAS video, the player first sends an HTTP GET request to the server to download a *manifest* file that has information about the media chunks. The player then sends an HTTP GET request for the first chunk. The quality of the requested chunk is usually specified in the chunk URI. Once the video chunk has been fully downloaded, it is decoded and played on the screen. Meanwhile, the player sends the request for the next chunk, whose quality is decided based on an adaptation algorithm [21, 18], and this process continues. MIMIC *exploits this strong serial request-response pattern for modeling a video session using HTTP logs*.

Approach: We log HTTP transactions corresponding to video traffic by deploying a

web proxy in the network. The web proxy provides information about the HTTP headers including the request URI, response completion timestamp, and content size. Note that information about HTTP logs can also be obtained by recording packet traces using a packet monitor, but at a higher processing cost. We collect information about the chunks such as chunk identifier and bitrate or quality from its URI. We use the session identifier field in the request URI to group the request logs into video sessions. Thus, we get request completion time (T_i), chunk quality (Q_i) and chunk size (S_i) for every chunk request i in the video session V. We use this information to estimate different video quality metrics namely, *average bitrate*, *re-buffering ratio*, and *number of bitrate switches*.

1.3 Inference for encrypted video traffic

In Chapter 4, we design eMIMIC, an SM-based approach that uses packet traces for QoE inference in encrypted video. It works by reconstructing video chunk transactions in a session from the packet traces and then using an approach similar to MIMIC to estimate QoE metrics. In designing eMIMIC, we solve three key challenges:

- **HTTP request reconstruction**: The first challenge is to identify the HTTP transactions in a video session. We use TCP headers for HTTP-level session reconstruction. We use the insight that 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.

- **Media type classification**: The second challenge is to identify video chunks in the reconstructed HTTP transactions as it may also include transactions corresponding to *metadata* and *audio* (if delivered separately). 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 with one or two bitrates levels. Thus, they

can be identified based on the size and size consistency and the remaining HTTP transactions must correspond to video.

- **Estimating bitrate of video chunks**: The third challenge is to estimate bitrate of the video chunks as it is required to calculate average bitrate and bitrate switches. We use both chunk size and observed throughput of previously downloaded chunks to estimate the bitrate of a chunk.

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. We use the QoE inference methodology similar to MIMIC to infer different QoE metrics.

1.4 Inference using coarse-grained network data

In Chapter 5, we present a methodology to infer video QoE using readily-available and light-weight network data in the form of TLS transactions. A major challenge with this data is that it is coarse-grained. It is no longer possible to obtain video segment information that was earlier available from packet-level traces. We address this challenge by using machine learning and formulate it as a supervised learning problem. The rationale here is that there may be patterns in how TLS transactions differ based on the QoE that can be inferred using machine learning models.

Given the coarse-granular nature of the data, we consider categorical estimation of the QoE metrics into *low*, *medium*, and *high*. In addition to the individual QoE metrics, we also estimate combined QoE. We find that the TLS transaction data may not always be useful in accurately estimating each individual QoE metric. However, it can estimate the combined QoE metric with reasonable accuracy for all the three considered video services.

We construct features for machine learning model using the sequence of TLS transactions in a video session. For every transaction in the session, we have its downlink data

size, uplink data size, start time, and end time. We use this data to construct the following three kinds of features: i) *session-level* features consisting of overall session metrics, ii) *transaction statistics* which capture different transaction-level metrics, and iii) *temporal statistics* to capture any temporal variations in the network conditions. We also develop features based on our experience from developing session-modeling based approaches. More specifically, we calculate transaction data rate (TDR) and downlink-to-uplink (D2U) ratio. *TDR* is the transaction downlink data divided by transaction duration, and intuitively is an indicator of the network quality. *D2U ratio* is the ratio of the downlink data to the uplink data. In HAS, the uplink data is typically an indicator of the number of segments requested. Hence, *D2U ratio* represents the amount of data downloaded per segment. This can be a useful indicator of the video quality. We use data collected from controlled experiments under diverse emulated network conditions to train our models.

CHAPTER 2
BACKGROUND AND RELATED WORK

In this chapter, we first explain HTTP-based Adaptive Streaming (HAS) and its Quality of Experience (QoE). This is followed by an overview of the related work on video QoE estimation using network measurement data and other relevant areas.

2.1 Internet video

Video has become the *killer app* for the Internet with no signs of slowing down (expected to be 82% of the global IP traffic in 2022 [3]). Most of the Internet video is primarily from *video-on-demand* (VoD) services but also includes a small but growing proportion of *live* video [27]. The ability to watch desired content whenever and wherever, rise of content providers like YouTube and Netflix, and continued improvement in encoding mechanisms and Internet bandwidth have been some of the factors that have led to this growth [4].

2.1.1 Streaming protocols

The growth of Internet video has been accompanied (and also impacted) by an evolution in the streaming protocols. Video in the early Internet was streamed using protocols such as RTMP (by Macromedia) [28] and RTSP (by IETF) [29]. These protocols were mainly UDP-based and offered several advantages such as dynamic seeking of content, support for live content delivery, and dynamic video bitrate based on the client's network conditions and device capabilities. However, they required deployment of special stateful servers that would push video content to the end-users. This implied a greater cost and complexity to deliver video content. As a result, these protocols were later replaced by HTTP-based adaptive bitrate streaming (HAS) protocol [30]. The use of HTTP allowed the content providers to use existing CDN infrastructure and it is also more firewall friendly. HAS still

allowed for bitrate adaptation by moving the adaptation logic to the client as the servers are stateless [30].

In HAS, a video is split into chunks or segments which are typically of same duration. Each of these chunks are encoded at predefined quality levels determined by encoding bitrate and resolution and hosted on a standard HTTP server. The bitrate adaptation algorithm at the client adapts the streaming video quality based on past TCP-throughput [18], current buffer occupancy [21, 20], or a combination of both [19, 22, 23]. The video metadata such as chunk encoding levels and request URI is obtained by downloading a manifest file at the beginning of video session. There are different implementations of HAS protocol with HLS [25] and MPEG-DASH [26] being the most popular. While parts of the framework for these technologies are standardized, they still allow content providers to choose some key video design parameters such as bitrate adaptation algorithms, encoding video and audio bitrates, and chunk duration [31]. Different choices of these parameters often lead to different end-user experience under similar network conditions [32, 33].

2.1.2 Video QoE

Streaming quality is estimated through a video *Quality of Experience (QoE)* measure. QoE is highly subjective as it depends on a lot of factors such as video content, user context, delivery quality etc [34]. This makes it hard to measure and quantify QoE even for content providers, let alone ISPs. Recent standardization efforts and literature propose to characterize QoE in HAS by multiple objective metrics namely *average bitrate, re-buffering ratio, quality switches*, and *startup delay* [14, 15, 16, 17, 2]. *Average bitrate* reflects the streamed *video quality*, based on the requirement for more bits to encode a higher quality image. In general, the bitrate of a particular quality level depends on the complexity of the video content being encoded and the encoding efficiency. The selection of quality levels made available to the video player is determined by each content provider. *Re-buffering ratio* is the proportion of the viewing time the video stalled because of playout buffer under-run.

Bitrate switches represent the video quality variations in the session. *Startup delay* is the time taken to start playing the video from the time the user opens the video.

An operator may further model the impact of these metrics taken together on the user experience, either through user studies [16] or data analysis [17, 2]. ITU-T has proposed a P.1203 model that takes into input per-second values of the key video QoE metrics and outputs a single mean opinion score metric (MOS) [35]. Our focus in this thesis is on estimation of the QoE metrics; modeling their impact on user QoE is a complimentary step.

2.2 Related work on QoE estimation using network data

Existing inference approaches can be broadly classified into two categories: ML-based and SM-based approach. An SM-based approach infers QoE by modeling a video session using the properties of the underlying streaming protocol. An ML-based 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. Here we give an overview of related work for both kinds of approaches:

2.2.1 ML-based approaches

Related work has proposed different ML-based approaches to infer video QoE using passive network measurement data. Most of these approaches assume access to packet-level traces. They differ in one or more of these dimensions, namely, the target QoE metric, machine learning model and features, and training methodology including collection of ground truth QoE metrics.

For progressive streaming, *OneClick* [36] and *HostView* [37] propose obtaining the subjective ground truth QoE through user feedback in the training phase and using regression-based models for their estimation. *Prometheus* [38] suggests using LASSO regression trained using instrumented clients to estimate objective video QoE metrics.

For encrypted HAS, ML16 [39] and *BUFFEST* [40] 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. [41] 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. [42] train different machine learning models to estimate the buffer state from the IP packet headers. Mazhar et al. [43] 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.

2.2.2 SM-based QoE inference approaches

SM-based approaches have been proposed for mainly HTTP-based progressive streaming protocols. Schatz et al. [44] estimate video re-buffering by modeling a session using TCP headers, while Dimopoulos et al. [45] extract key pieces of information from the HTTP headers to infer re-buffering. However, these methods were designed for HTTP progressive streaming and would not work for HAS video. In this thesis, we develop SM-based approaches for HAS video. More specifically, we develop MIMIC [46] and eMIMIC [47], that use session modeling to infer QoE metrics in the case of unencrypted and encrypted video, respectively.

2.2.3 Scalability of inference approaches

Most of the existing methods focus on improving QoE inference accuracy. More recent work has started to consider the scalability aspects of using packet-level data by exploring the computation overhead of the collected features [48], using a minimal set of features [49], or proposing in-network processing [50]. In this thesis, we also consider an

alternative approach that uses coarse-granular network data instead of packet traces for inference.

2.3 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 [51]. More recently, traffic classification efforts have focused on classifying applications within HTTP. Shbair et al. [52] use ML to classify application classes within HTTPS traffic. ML-based techniques have been proposed to specifically identify HAS video flows in the network [53, 42]. eMIMIC currently uses Server Name Indication (SNI) field in TLS handshake [54] 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.

2.4 QoE-based network management

Several in-network optimizations have been suggested that take into account video QoE. Chen et al. [55] and Mansy et al. [56] 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 [57, 58]. Kassler et al. [59] propose QoE-based routing using software defined networking (SDN). Mustafa et al. [60] propose using SDN for in-network video QoE-aware resource management. We believe that QoE estimation can help network operators in finding network locations that need attention and assess the impact of in-network modifications after deployment.

2.5 Video performance characterization

Several active measurement studies exist that characterize performance of the commercial video players [33, 32, 61] and CDNs [62, 63, 64]. Similarly, large-scale measurement efforts have been undertaken to characterize video QoE using passive data collected from different vantage points in the video delivery path, namely, commercial video players [65], CDNs [66] or both [67]. Erman et al. [68] characterize mobile video performance from the unique perspective of MNOs. We deploy MIMIC in a part of the network of a major US MNO. Using data collected from these deployment over a period of 12 days, we provide a fresh take on the findings by Erman et al. focusing on the mobile video evolution in the past few years as well as a few additional insights analyzing the impact of mobility and network demand on video performance.

CHAPTER 3

MIMIC: QOE INFERENCE FOR UNENCRYPTED VIDEO

3.1 Introduction

In this chapter, we propose a methodology called MIMIC[1], that uses semantics of HAS to estimate video QoE metrics from passive network measurements for unencrypted HAS videos. We design MIMIC as a session modeling (SM)-based approach. This makes it minimally dependent on ground truth QoE metrics. Furthermore, it can *generalize* well across different services as it relies on the semantics of the HAS protocol to estimate video QoE which are generally consistent across services. Finally, MIMIC gives a fine-granular view of video QoE metrics *per session* by estimating their exact values as opposed to the categorical estimates. This makes it useful in the case of active QoE-based resource allocation.

MIMIC is based on the observation that the HTTP logs of HAS videos can give significant information to model a video session on the client. We use HTTP logs to estimate three video quality metrics: *average bitrate, re-buffering ratio*, and *bitrate switches*. An MNO can either use each of these metrics individually or feed the estimated values into an appropriate QoE prediction model, such as [16], to get a single score reflecting the QoE for the video session. In addition, we also monitor network overhead due to chunk replacement which is an important metric for network operators. MIMIC is similar in spirit to [44] and [45] that use TCP-layer metrics and HTTP logs respectively to model a video session at client. However, these methods were designed for HTTP progressive streaming and would not work for HAS video.

[1]MIMIC stands for **M**easur**i**ng **M**ultimedia QoE **i**n **C**ellular network. The acronym is suggestive of our approach, *i.e.* we try to mimic the client video session playback using network traces

MIMIC is developed in a controlled environment and then validated on a large-scale using network data from a major cellular provider and ground truth QoE metrics from a major content provider. MIMIC can accurately predict the average bitrate within a relative error of 10% for 70%-90% of video sessions. MIMIC is able to predict re-buffering ratio within an absolute error of 1% for 65%-90% of video sessions. We quantify the network overhead due to video chunk replacement and observe that a significant number of sessions can incur a high overhead of 20% or more.

We also deploy MIMIC in the real network of a large U.S. based Mobile Network Operator (MNO). This is done as a part of a QoE inference system we designed, called VideoNOC [69]. Using data collected from the deployment, we demonstrate that MIMIC can provide an MNO with a spatio-temporal view of video demand and QoE across the network. We analyze the impact of network factors such as user mobility and demand on the video QoE. Furthermore, our analysis provides unique insights from the network data otherwise not available from QoE monitoring performed at end devices by Content Providers (CPs) or on servers and CDNs. These insights motivate both best practices as well as opportunities for cooperation between MNOs and CPs for the benefit of all stakeholders in the mobile video ecosystems.

The rest of the chapter is organized a follows: Section 3.2 describes methodology to estimate video QoE metrics in detail. In Section 3.3, we discuss the results from controlled experiments as well as cellular network data. We present the measurement analysis from MIMIC deployment in Section 3.4. We conclude the chapter in Section 3.5.

3.2 Methodology

MIMIC is based on the insight that *there exists a strong serial request-response pattern in the HTTP transactions in a HAS session that can be used for modeling a video session* Conceptually, we utilize HTTP logs collected from the network to model a video session on the client. We illustrate MIMIC with a mobile app, referred to as VideoApp (anonymized

for confidentiality), from a large mobile video service. We first describe measurements collected passively and then discuss how these measurements can be used to estimate different video QoE metrics. We then briefly describe the benchmark data that we use to evaluate the accuracy of our approach.

3.2.1 Passive Measurements

Our passive measurements include HTTP logs for VideoApp recorded by deploying a web proxy in the network. We identify the logs as belonging to the VideoApp by inspecting the request URIs. In addition to the request URI, the web proxy also provides the HTTP response completion timestamp and content size. Note that deep packet inspection techniques could also be used for this purpose, but at higher processing cost. The log data used for this study is anonymized and does not contain any user-identifiable information. We only consider logs that correspond to video chunk requests.

Figure 3.1 shows a sample chunk requst URI from the VideoApp session. We use the session identifier field in the request URI to group the request logs into video sessions. From the URIs of these chunk requests, we then extract the chunk identifier and bitrate or quality. We also collect some meta-data about each session such as device OS, OS version, content type and content identifier from the request URI and headers. Note that chunk bitrate (quality) and content identifiers in the URI are typically represented by arbitrary or randomized service-specific unique identifiers rather than human-readable values.

Chunk replacement: It is common to observe multiple chunk requests with the same chunk identifier, but different bitrate in the logs. This essentially means that the video client replaced an already requested chunk. This behavior has been alluded to in a previous study [33] but has not been quantified at large scale. Chunk replacement can happen

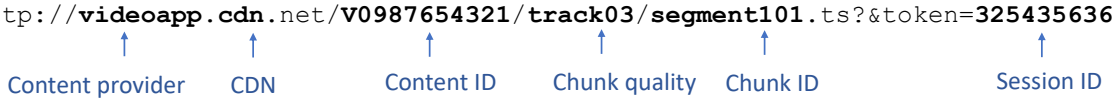


Figure 3.1: Chunk URI template and the extracted information

due to several reasons, including (i) player trying to improve user experience when network conditions allow, and (ii) recovering from various errors or overly aggressive but aborted attempts for high-quality chunks. We handle chunk replacement by using only the most recently downloaded bitrates of each chunk to estimate QoE metrics, while carefully accounting for all replaced chunks. The replaced chunks are important from the perspective of both MNO (represent wasted network resources) and end user (represent wastage of limited data plan). We call the total amount of data due to replaced chunks as *chunk replacement (CR) overhead*.

3.2.2 Estimating QoE metrics

From the passive measurements, we get request completion time (T_i), chunk quality (Q_i) and chunk size (S_i) for every chunk request i in the video session V. The chunk duration (L) in seconds is obtained by investigating the *manifest* file for few videos in out-of-band experiments. We observed that VideoApp uses different chunk duration for Live and Video on Demand (VoD) content. The total number of chunks downloaded in a session after accounting for chunk replacement are denoted by N. We use this information to estimate different video quality metrics in the following manner:

- *Average bitrate*: We estimate the average bitrate by taking the time average of the collected chunk size of the session.

$$\hat{br} = \frac{\sum_{i=1}^{N} S_i}{N \times L} \tag{3.1}$$

- *Re-buffering ratio*: Intuitively, re-buffering time is estimated by keeping an account of video chunks that have been downloaded and the part of that video content that should have been played so far. Let B_i denote the total re-buffering since the beginning of the play until the time chunk i has been downloaded i.e. T_i. Clearly, B_1 is zero by definition as the initial re-buffering is termed as the video startup time.

The re-buffering time between two consecutive chunk download times T_i and T_{i-1} is denoted by b_i. Then, B_i can be simply written as $\sum_{k=1}^{i} b_k$ and each of the b_i can be calculated as follows:

$$b_i = \max(T_i - T_1 - B_{i-1} - (i-1) \times L, 0) \quad \forall i \geq 2 \tag{3.2}$$

Here, $T_i - T_1 - B_{i-1}$ represents the total video that should have been played since the beginning of the session and $(i-1) \times L$ represents the video content that has been downloaded. The re-buffering ratio can then simply be written as

$$\hat{rr} = \frac{B_N}{N \times L + B_N} \tag{3.3}$$

- *Number of bitrate switches*: Number of bitrate switches can be calculated simply by calculating the number of times the chunk quality changed between consecutive chunks. Here I is the indicator function and has a value of 1, if the consecutive chunks do not have same quality, zero otherwise.

$$\hat{br\_switch} = \sum_{i=2}^{N} I(Q_i \neq Q_{i-1}) \tag{3.4}$$

3.2.3 Ground Truth

To validate the accuracy of our approach we require the actual video QoE metrics. For this purpose, we use session-level QoE metrics collected by a mainstream 3rd party video analytics SDK built into the VideoApp. This is the typical approach used by commercial video services for QoE monitoring. It is our understanding that SDKs use API calls to the native player to register events such as playing, stopped, buffering at the application layer. The detailed logic is proprietary to each vendor. The QoE metrics provided by in-app SDK consist of average bitrate, re-buffering ratio, and video startup time. We assume

Table 3.1: Bandwidth profiles used for controlled experiments.

BW1	constant bandwidth of 10000 kbps
BW2	2000 kbps with 20 kbps from t=180 s to t=240 s
BW3	bandwidth alternating between 2000 and 20 kbps every 30 s
BW4	bandwidth changes every 10 seconds, range=[20,10000] kbps, mean=2951 kbps, stddev=3932 kbps

Table 3.2: QoE metrics estimation results from controlled experiments: VoD content, OS1

Bandwidth profile	Average bitrate (kbps)			Re-buffering ratio (%)	
	M.V.	P.V.	G.T.	M.V.	G.T.
BW1	3213	3680	3690	0.61%	0.09%
BW2	1220	1442	1440	3.48%	3.41%
BW3	855	1003	1030	32.08%	32.7%
BW4	1983	2276	2330	11.41%	12.0%

that metrics from the SDK accurately capture user-perceived QoE. The in-app SDK does not report bitrate switches and chunk replacement. The ground truth logs are anonymized for privacy by the SDK, so that users or end devices could not be identified.

3.3 Evaluation

The evaluation involves measuring the QoE metrics of VideoApp sessions using the network data and comparing them to the QoE metrics reported by the in-app SDK, referred to as ground truth. We first conduct experiments in a controlled environment, followed by validation using the real network data. In our analysis, we split the VideoApp sessions by content type, i.e., Live or VoD, and Operating System (OS). This is because the video system parameters and the exact HAS implementation may depend on the content type and OS. We consider the two popular mobile OS-s and refer to them as OS1 and OS2.

Table 3.3: QoE metrics estimation results from controlled experiments: Live content, OS1

Bandwidth profile	Average bitrate (kbps)			Re-buffering ratio (%)	
	M.V	P.V	G.T.	M.V.	G.T.
BW1	3012	3242	3270	0.00%	0.14%
BW2	1146	1344	1260	8.87%	10.3%
BW3	769	917	951	12.69%	14.3%
BW4	994	1092	1190	11.70%	13.31%

3.3.1 Evaluation using controlled experiments

Experimental Setup: The experimental setup consists of a smartphone with the VideoApp installed and a Linux box acting as a WiFi hotspot. Squid proxy is deployed on the Linux box to log all HTTP traffic from the smartphone. We use Linux tc to control the downlink bandwidth to the smartphone. A single test run constitutes of streaming a specific video in VideoApp for a duration of 5 minutes and under a specific bandwidth profile. The collected proxy logs from these test runs correspond to the logs we can get from the real network and we use them to calculate the video QoE metrics and then compare them to the ground truth. Creating a special test user allows us to positively match each test run with the VideoApp ground truth logs.

We run experiments with one representative video from Live and VoD content served by VideoApp. Each video was continuously streamed under four different bandwidth profiles described in Table 3.1. The goal of these experiments is to do an initial rather than exhaustive validation of our methodology.

Results: Table 3.2 shows the comparison between the measured value (M.V.) and ground truth (G.T.) value of QoE metrics under different bandwidth profiles for VoD on OS1. Our methodology appears to consistently underestimate the average bitrate. Upon a closer examination, we find that the difference in our measurements can be attributed to the difference between the way we compute average bitrate and the way the in-app SDK calculates it. The in-app SDK uses the declared bitrate in the manifest whereas we consider the actual size of the chunks on the network while estimating the bitrate value. The

declared bitrate represents the peak chunk bitrate over the entire video, as required by HLS design, thus leading to higher estimate of average bitrate by the in-app SDK. To verify this, we also calculate the predicted value (P.V.) using the declared bitrates in the *manifest* file. As shown in Table 3.2 the average bitrate predicted using this method is very close to the ground truth.

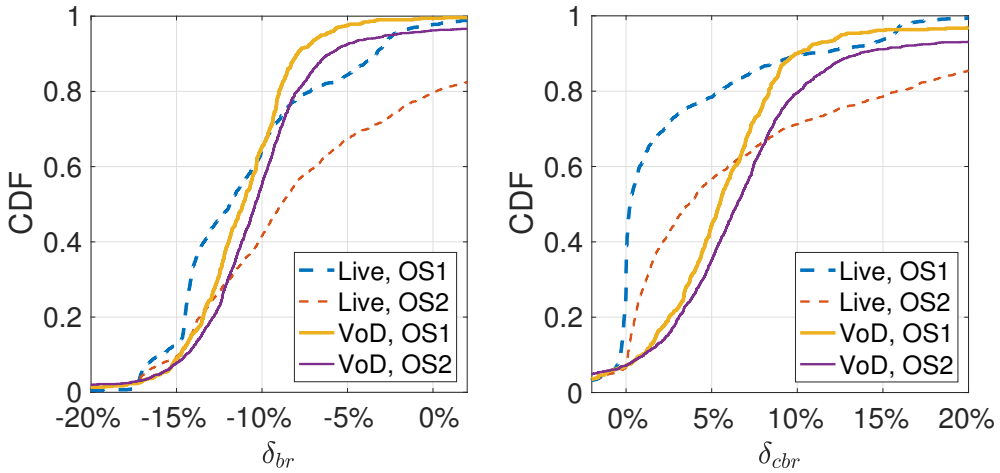
Our methodology can predict re-buffering ratio quite accurately with an absolute error less than 1% for VoD. The results are similar for experiments with Live video as shown in Table 3.3. In addition, we also calculate the number of bitrate switches and CR overhead, not shown here as we do not have the ground truth for these. The controlled experiments show that we can very accurately predict the video QoE metrics using the network data.

3.3.2 Evaluation using large-scale real network data

Dataset: We deployed a web proxy in the cellular network of a major operator in the U.S. to obtain HTTP logs for VideoApp sessions. The logs were collected for a period of 12 days in the year 2017, in a part of the network covering a fraction of packet gateways. We estimate the QoE metrics for the logged sessions using MIMIC. Note that we are not able to store the raw HTTP logs due to large space overhead but only process the logs in a streaming fashion and retain the session-level QoE metrics and associated meta-data.

Ground Truth: We again use the QoE metrics collected by the in-app SDK. However, unlike the active experiments, it is not trivial to match the sessions collected on the network with their corresponding in-app SDK logs. Due to the anonymization of both data sets, we have to resort to the following matching logic. We use the session content identifier, session start time and duration, OS kind and version, and CDN to match sessions. We further filter out sessions less than 1 minute since it is challenging to match the network session duration with the ground truth playtime for such short sessions. In the case of a single match between data sets, we use it for comparison, while in the case that a single VideoApp session matched with more than one proxy sessions, we discard all of them.

For the time period under study, we were able to match 70,214 sessions, which is a small fraction of all VideoApp sessions. Both Live and VoD are well-represented in this set. Although the efficiency of our highly simplistic matching logic is low, we still get a large number of sessions to validate our methodology.



(a) Using chunk size

(b) Using declared bitrates

Figure 3.2: CDF of relative error in average bitrate estimation

Average bitrate: For *average bitrate*, we calculate the signed relative error (δ_{br}) in the estimated average bitrate ($\hat{br}$) and the ground truth average bitrate (br), defined as $\frac{\hat{br}-br}{br}$. Figure 4.8a shows the CDF of the relative error for sessions split by OS kind and content type. As expected, we underestimate the the average bitrate when compared to the in-app SDK reported bitrate since we are using the chunk size to estimate the bitrate as opposed to the bitrate values from the manifest by the in-app SDK. Note that from the point of view of MNO, the bitrate calculated using chunk size gives a more accurate indication of network load. We also estimate average bitrate using the declared values from the manifest in the controlled experiments. Figure 3.2b shows the CDF of the relative error (δ_{cbr}). We can predict the average bitrate within a relative error of 10% for at least 70% (90% for OS1) of sessions.

We also note some challenges in bitrate estimation. We consistently overestimate average bitrate, although by a small value, when using declared values from the manifest.

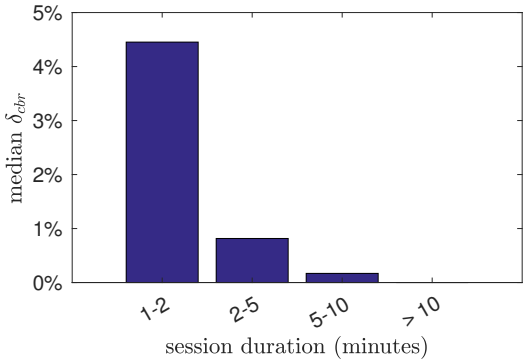


Figure 3.3: Median relative error in average bitrate estimation vs. session duration

One reason for this overestimation is that we use all of the downloaded chunks during the session for average bitrate calculation unlike the in-app SDK which only considers chunks that have been played. From experiments, we observed that VideoApp sessions typically start with a low chunk quality and the quality increases later on as the playback progresses. Hence, if chunks in the playback buffer are also considered, it leads to overestimation of average bitrate. To validate this hypothesis, we classify the sessions into bins according to their duration and compute the median of the relative error in bitrate estimation for each bin. As shown in figure 3.3, the median relative error decreases as the session duration increase. This is because the relative contribution of the fixed-size buffer in the average bitrate decreases as the session playtime increases. Tracking the buffer occupancy in bitrate calculation can help make our estimation more accurate.

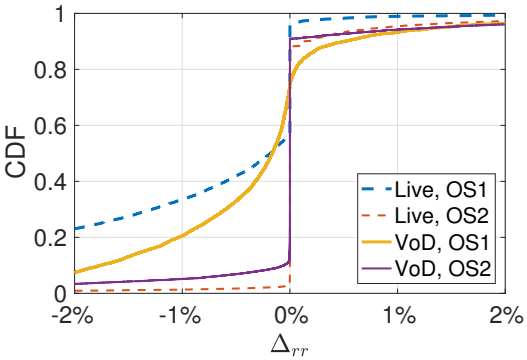


Figure 3.4: CDF of error in re-buffering ratio estimation

We also observed that the errors in bitrate estimation are higher in general for sessions

Table 3.4: Re-buffering ratio confusion matrix, VoD OS1

Ground Truth	Predicted re-buffering ratio		
	low rr	medium rr	high rr
low rr	90.6%	7.7%	1.6%
medium rr	49.9%	47.0%	3.1%
high rr	1.7%	15.5%	82.8%

Table 3.5: Re-buffering ratio confusion matrix, VoD OS2

Ground Truth	Predicted re-buffering ratio		
	low rr	medium rr	high rr
low rr	94.4%	4.6%	1.0%
medium rr	56.5%	32.2%	11.3%
high rr	10.7%	9.7%	79.6%

on OS2 as compared to OS1. We speculate that it could either be because of a different chunk replacement policy in OS2 or a different methodology used by in-app for calculating bitrate. We plan to investigate this in detail in our future work.

Re-buffering ratio: We calculate the signed difference (Δ_{rr}) between the estimated re-buffering ratio ($\hat{rr}$) and the ground-truth re-buffering ratio (rr), defined as $\hat{rr} - rr$. Figure 3.4 shows the CDF of Δ_{rr} split by OS and content type. We can accurately predict the re-buffering ratio within an absolute error of 1% for 90% of session on OS2 and for 65% of sessions on OS1. Our methodology appears to underestimate the re-buffering ratio.

To understand this further, we categorize the re-buffering ratio into low ($rr < 1\%$), medium ($1\% < rr < 10\%$) and high ($rr > 10\%$) and compare the categorical predictions with ground truth. Table 3.4 and 3.5 show the confusion matrices of these categorical predictions for VoD sessions on OS1 and OS2 respectively. For both OS types, our methodology can predict low and high re-buffering sessions with reasonably high accuracy. However, many sessions with medium re-buffering are classified low re-buffering. From a network operator's perspective, it is more important to identify sessions with higher re-buffering as compared to medium re-buffering.

One possible reason for re-buffering underestimation could be because of "trick play". Fast-forwarding or rewinding content in the video usually leads to resetting of buffer oc-

Table 3.6: Network overhead due to chunk replacement

Content type	OS type	% sessions w/ non-zero CR	mean CR overhead (% bytes)	% sessions w/ CR overhead $\geq 20\%$	CR overhead (% total data)
Live	OS1	52.8%	7.5%	5.2%	2.8%
Live	OS2	89.2%	18.3%	35.8%	6.2%
VoD	OS1	91.9%	7.0%	10.9%	5.2%
VoD	OS2	92.6%	6.5%	9.4%	2.8%

cupancy. Our estimation methodology does not detect and take into account trick play and would end up underestimating the re-buffering ratio in these cases. We also observed higher number of sessions with medium re-buffering on OS1 as compared to OS2. We speculate that this could be because of difference in HAS implementation or in-app SDK's reporting methodology. Identifying the root cause of this behavior is a part of our future work.

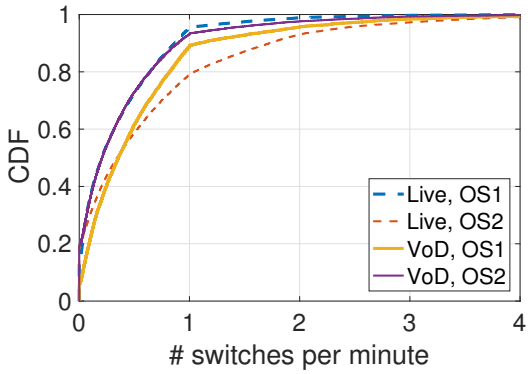


Figure 3.5: CDF of number of switches per minute

3.3.3 Additional metrics estimated from network data

Here we show the distribution of two additional metrics for which we do not have the ground truth. However, they demonstrate that we can obtain additional insights not otherwise available from in-app measurements.

Chunk replacement (CR) overhead: CR overhead is defined as the % of data in a video session transmitted due to replaced chunks. Table 3.9 shows the CR overhead for VideoApp sessions split by OS and content type. A large number of sessions, as high as 90%, have non-

zero chunk replacement. This can be attributed to the fact that most of the VideoApp sessions always start by downloading multiple lower quality chunks to quickly fill the playback buffer. The player replaces these chunks with higher quality chunks if it infers there is enough network bandwidth. This behavior is due to application or underlying OS native player design. CR accounts for 2.8% - 6.2% of the total network data for the VideoApp service. Furthermore, up to 35% of (Live, OS2) sessions have a CR overhead greater than 20%, which is a non-trivial overhead. This points to the need of looking into optimizing the trade-off between improved user experience and network overhead. However, the adaptation logic and CR behavior can be hidden within the native player design and not accessible to video content providers.

Bitrate switching: We compute bitrate switches per minute for each session and plot the CDF of its distribution in Figure 3.5. A large number of sessions have non-zero bitrate switches. This can be attributed to the player behavior during the startup phase of the video and later adaptation. The VideoApp player typically starts with a low bitrate level and switches up the bitrate as the playback progresses. We also observe high number of bitrate switches for Live sessions on OS2 and VoD sessions on OS1. We speculate this could be because of the differences in the player adaptation logic on these platforms.

3.4 Measurement study

We deploy MIMIC in a part of the network of a major U.S. MNO as a part of a video QoE inference system, called VideoNOC [69]. Using the data collected from the deployment, we characterize video services and their QoE from an MNO's perspective. Our measurement study is divided into three parts:

- We describe the relative video service usage and analyze its impact on the network.
- We provide insights into the design of a variety of streaming services and its implications on QoE and network usage.

- We then analyze the impact of network factors on video QoE, namely mobility and higher video demand per cell.

We also compare our findings to a 2011 study [68], to point out the evolution of video streaming in cellular networks over the years.

Data set: The data set from which we draw sample network locations used for this analysis comprises of logs spanning two weeks in 2017. The data consists of 272 million anonymized sessions from a sample of 15 different Content Providers (CPs) amounting to 5,070 TB of data. We filter out sessions shorter than 2 minutes in duration to remove potential skew due to auto-play feature, users that browse or sample videos, and overall startup effects (e.g., initial buffering, which we discuss separately in detail).

We group CPs by the estimated broad type of content served – user-generated content (UGC), premium video-on-demand (VoD) and live TV (Live) content. If a CP uses HTTPS, we refer to it by appending an 's' to the end of the name. UGC services are streaming predominantly user-generated videos with a varying proportion of commercial videos and advertisements in up to 4K resolutions. Most of them are considered online social networks. VoD CPs are large paid streaming services with movies and shows offered in various quality levels up to 4K resolution, either under independent subscription or as a part of the home TV services. Live CPs offer live TV streams from large TV and sports broadcasters.

3.4.1 Handling encrypted video

Three out of the 15 CPs we consider used end-to-end encryption. We use a coarse-granular metric, called *Session Throughput* (ST), as a proxy for a QoE metric in the analyses involving these services. Here we describe the manner in which ST is computed and its correlation with different QoE metrics.

For encrypted traffic, the web proxy provides TLS transaction statistics with a single TLS connection potentially corresponding to multiple HTTP requests. The QoE estimation

Table 3.7: Correlation between ST and ground truth.

Content type	$\rho_{(br,ST)}$		$\rho_{(rr,ST)}$	
	OS1	OS2	OS1	OS2
Live	0.89	0.88	-0.18	-0.10
VoD	0.76	0.80	-0.27	-0.37

methodology for unencrypted traffic can not be applied to encrypted traffic as per-HTTP transaction statistics are not available. Hence, for encrypted traffic, we calculate the session throughput (ST), which is defined as follows:

$$ST = \frac{\sum_{i=1}^{N} S_i}{T}, \tag{3.5}$$

where S_i is the content length of the i^{th} TLS transaction in a session of N transactions and T is the total session time as observed on the network. We rely on degradation in ST in order to detect degradation in streaming quality. For a session, we record the overall session throughput as well as per-minute session throughput which is calculated by considering transactions only in the current minute. The latter is done to detect temporary variations in the session QoE. Note that ST is also calculated for unencrypted video services with the TLS connection content length replaced by HTTP transaction content length.

We correlate the in-app SDK average bitrate and re-buffering ratio with the calculated ST for the matched sessions in VideoApp. Table 3.7 shows the Pearson's correlation coefficient (ρ) values between ST and average bitrate ($\rho_{(br,ST)}$) and between ST and re-buffering ratio ($\rho_{(rr,ST)}$) for each analyzed OS.

We find that average bitrate and ST are strongly positively correlated, while there is a weak negative correlation between re-buffering ratio and ST. This shows that session throughput can be used as an indicator of video QoE. Specifically, degradation in session throughput would indicate degradation in average bitrate in most cases and high re-buffering in some cases. Recent work also shows that average downlink throughput maps reasonably well to video QoE [70].

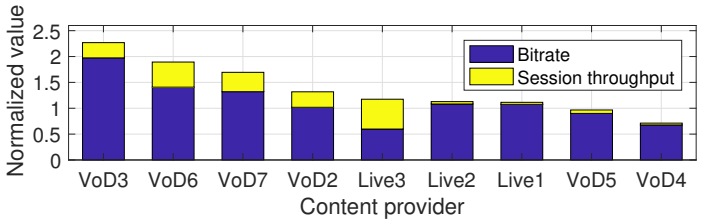


Figure 3.6: Normalized median per-session average bitrate and session throughput from a sample of CPs and network locations. A value of 1 corresponds to 1.5 Mbps.

3.4.2 Insights into relative video usage

As video services and their delivery evolved over the past several years, one example use case is to consider the relative demand, efficiency and popularity across CP categories and network locations.

Relative contribution to overall video demand: Considering the contribution to the overall number of video sessions streamed, we observe that the top 5 CPs from our sample set generate 95% of video sessions. They consist of UGC and VoD CPs, with majority being UGC. Other VoD CPs generate 4.5% of video sessions, and Live CPs remaining 0.5%. While Live video is currently a minor contributor to video sessions in cellular, this is an emerging category that may grow in the future. The relative contribution in terms of data demand is similar as the same top 5 CPs contribute 93.5% of the overall data demand. This is qualitatively similar to the 2011 study [68]. *Given the significant impact of these top 5 CPs, optimizing these services for cellular networks would have a large overall impact on the efficiency of network resource utilization.*

Understanding efficiency of video content delivery: We consider the delivered video bitrates and their relationship to the overall video session throughput (ST) to understand the efficiency of delivery vs. QoE. We focus on a subset of CPs for which we can estimate the available bitrates using request URIs. Figure 3.6 shows the medians of average per-session bitrate and ST from a sample of CPs and network locations. The bitrate ranges indicates significant increase compared to the 2011 study, which found that most video sessions were encoded at 255 Kbps. This improvement comes with transition from 3G to LTE and from

progressive download (PD) to adaptive bitrate (ABR) streaming technologies. Another aspect of video evolution compared to 2011 study is that nearly all detected video sessions use some form of ABR, with PD diminished.

However, the question arises whether the high bitrates are reasonable. Video sessions in cellular networks are typically streamed to small-screen devices, such as phones. Existing studies show that on small screens, bitrate levels of 1 to 1.5 Mbps with state-of-the-art encoding techniques can provide a high visual quality with a diminishing utility beyond that point [71]. Delivering excessive video bitrates wastes data allotments and network bandwidth, something of utmost importance under limited radio spectrum. The data reveals that most of the sample CPs actually exceed the median per-session bitrates of 1.5 Mbps, shown as the normalized value of 1 in Figure 3.6, making the excess bandwidth use more of a rule rather than an exception.

Further compounding the problem, ST often exceeds the bitrate by a significant margin, indicating that many services deliver much more data than the useful encoding bitrate, with overhead reaching up to 50% (Figure 3.6). This comes as a consequence of several possible issues: i) use of less efficient encoding, such as Constant Bit Rate (CBR), as opposed to Variable Bit Rate (VBR), ii) less efficient transport and packaging schemes, which in cases of separate audio and video streams could lead to more overhead, iii) various chunk duplication and replacement behaviors (discussed later in detail), etc. We find one or more of these issues present in many CPs.

On the other hand, we can see that some services have minimal overhead and tend to stream appropriate bitrates for small screens. It is encouraging to see that the service designs conscientious of cellular environment and device context are starting to appear among CPs. *These observations raise a question of whether the surging demand for video bandwidth could be significantly curbed by simply considering screen size of mobile devices, and carefully selecting more efficient methods, thereby reducing the obvious resource waste. The findings present a strong argument in favor of utility-based adaptation for more*

Table 3.8: Service design aspects across CPs

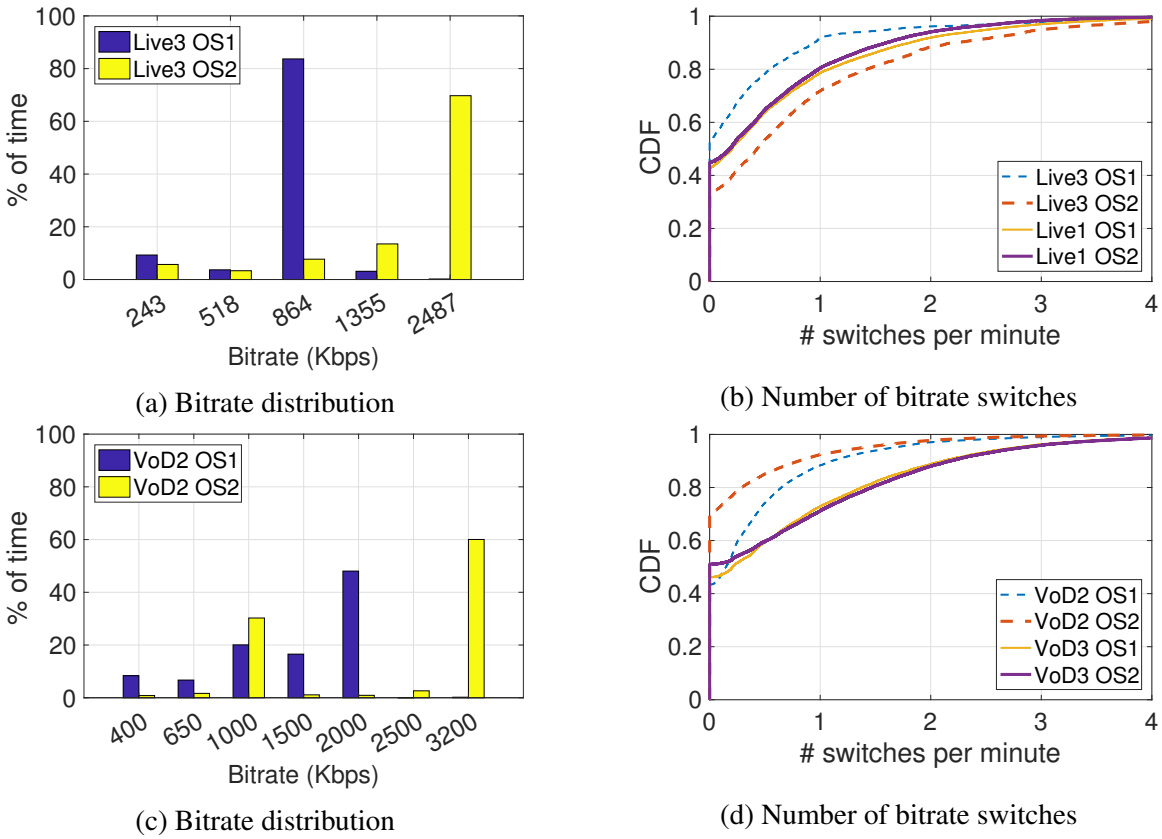
CP	Bitrate range (Kbps)	Number of bitrate levels	Chunk duration (s)	Chunk IAT (s)	Estimated buffer in OS1 (s)	Estimated buffer in OS2 (s)
UGC1S	-	-	-	4.7	35	35
UGC2	-	-	-	3.2	23	25
UGC3S	-	-	-	3.5	25	25
UGC4S	-	-	-	-	22	22
UGC3	254-1291	4	3.3	0.7	-	-
VOD1	-	-	-	-	60	27
VOD2	130-2664	11	10,5	7.8	20	52
VOD3	266-4681	8	4	3.6	10	16
VOD4	268-2408	6	4	3.9	14	25
VOD5	232-3219	7	4	3.5	45	29
VOD6	194-5901	11	3,10	8	20	35
VOD7	264-3875	8	9	8.7	12	25
LIVE1	175-3404	7	8	7.4	10	12
LIVE2	171-3402	7	8	7.7	13	13
LIVE3	153-2744	15	6	5.6	16	7

efficient use of network resources.

Cell coverage: We observe the proportion of cells with traffic from each CP, out of the total number of cells in our sample dataset. This offers an insight into the *network coverage* across CPs. We observe that a small number of CPs can be observed in a majority of radio cells, with the top top 5 CPs cumulatively covering 99% of cells. *This suggests that thoroughly understanding QoE for those CPs could provide a representative view of video QoE across different parts of the entire network. Combined with the highly skewed demand, it may be possible to use sampling to improve speed and reduce the processing cost of video QoE data*. However, it is still valuable to study other CPs, from the perspective of interaction with most popular ones and general traffic, and because their relative importance to users might be higher, as these are generally premium services.

3.4.3 Insights into video service design

We now consider the video service design parameters used by various CPs, focusing on those that have high impact on QoE and efficient use of resources in cellular network (Ta-



(a) Bitrate distribution

(b) Number of bitrate switches

(c) Bitrate distribution

(d) Number of bitrate switches

Figure 3.7: Variability of QoE metrics across OS for LIVE3 and VOD2

ble 3.8).

Bitrate range: We report detected *Bitrate range* across a *Number of bitrate levels* for CPs in the dataset. While most CPs provide a similar lowest bitrate (near 200 Kbps), the range drastically varies, with some CPs exceeding 4 Mbps at the high end. While some CPs deliver all available bitrates over cellular networks, others do not deliver the highest bitrates in cellular networks in any appreciable amount (e.g. LIVE1, LIVE2, VOD4, VOD5). We confirm that this is the CP design having nothing to do with network performance by finding that the highest requested bitrates across cells with low, medium, and high load, are generally the same. This points to a form of internal control or cap on bitrates in cellular, which we further confirm by manual inspection of mobile apps. Such controls are implemented either using *CP-tailored manifest files for cellular or application settings (e.g. by settings in user preferences).*

We further find it peculiar that there is no apparent rule followed by CPs on the number and granularity of bitrates offered. For approximately similar range, there could be a large difference in number of bitrates. One reason could be differences in streaming technologies across devices for the same CP leading to different encoding requirements and CDN storage management. Another reason could be differences in bitrate adaptation strategies where more bitrates may achieve some design objective for the CPs, such as smoother visual quality transitions during adaptation.

Chunk duration: *Chunk duration* in Table 3.8 shows the duration of a video chunk obtained by manually inspecting the manifest file for some of the services. It appears that there is a tendency where Live services use longer chunks than VoD services. These choices might be the remnants of the legacy recommendations in HLS, however, it is not possible to determine the true reason from network traffic. Longer chunks might, however, reduce the flexibility and agility of the player to adapt to rapidly changing network conditions, often found in cellular networks. We also compare these values to the median *Chunk inter-arrival time (IAT)* observed on the network for each video service. The values are very close to the actual chunk duration for the services. This indicates a viable possibility for automatically detecting chunk durations from their inter-arrival times, thereby removing the need to manually inspect these services.

Buffer levels: Another source of network overhead in video streaming can be inadequately sized video buffer. In HAS, players usually fill the video buffer and then enter into the steady state phase where they maintain the buffer level by fetching another chunk once there is space in the buffer. Downloading a large amount of video, especially during the startup buffering phase, can lead to waste of network bandwidth in case of abandonment. Past work has indicated that a large number of users tend to sample video content [72], or do not watch the entire content [68].

To assess the buffering behavior at large, we use the following methodology, which is intended to estimate average buffering across sessions, as opposed to precisely determine

the maximum buffer size in each player. For each CP, we estimate and compare the average amount of content (in seconds) buffered in the first minute of the video session. The buffered content ($\hat{B}_{startup}$) is estimated by dividing the amount of data downloaded in the first minute ($D_{startup}$) of the video session by the average bitrate ($\bar{br}$) of the entire session.

$$\hat{B}_{startup} = \frac{D_{startup}}{\bar{br}} \tag{3.6}$$

In case of CPs for which we do not have the bitrate, we use ST as an estimate of the bitrate. The median of $\hat{B}_{startup}$ across all sessions of a CP is calculated and shown as *Estimated buffer* (Table 3.8) for OS1 and OS2. For encrypted services, we are not able to detect the OS type and hence the same values are shown for both OS. We exclude UGC3 from this analysis as we found that this CP pre-loads multiple videos in background before users specifically select to play a title.

For the same CP, we find differences in the video buffer content across OS's. This indicates different design choices across OS's. The video buffer for *Live* content is smaller than for *VoD* and *UGC*, which is to some degree expected, as *Live* services are streaming content generated in real time and it is not desirable to have a long lag to real time. For most CPs, the buffer is less than 30s. However, we also find larger buffers for some CPs (e.g., 60s for VOD1 on OS1), indicating non-trivial network overhead due to abandonment in these CPs. *This points to the need for exploring smarter strategies to determine the buffer size, especially in the beginning of the video playback, in order to minimize network overhead due to users sampling videos.*

Bitrate usage and switching variability across OS: Different buffering strategies by OS prompt a closer examination whether other design aspects differ across OS-s, resulting in different QoE. We look deeper into variability in bitrates and switching using two example services, LIVE3 and VOD2.

Figure 3.7a shows the distribution of bitrates played by LIVE3 on OS1 and OS2. Nearly 70% of LIVE3 content on OS2 is streamed at an average bitrate of 2,487 Kbps, whereas

Table 3.9: Network overhead due to chunk replacement

CP	% sessions w/ non-zero CR overhead	mean CR overhead (% bytes)	% sessions w/ $\geq$ 20% CR overhead	CR overhead (% data)
UGC3	79.99%	13.68%	26.48%	9.14%
LIVE1	73.29%	8.68%	13.42%	5.42%
LIVE2	63.89%	7.75%	12.79%	4.95%
VOD5	60.52%	6.38%	12.66%	7.80%
LIVE3	60.07%	6.55%	9.30%	3.58%
VOD6	41.14%	8.37%	17.30%	4.62%
VOD7	38.13%	5.46%	10.06%	5.11%
VOD2	25.93%	4.03%	7.65%	4.54%
VOD3	20.07%	2.75%	4.79%	4.75%
VOD4	3.02%	0.31%	0.55%	1.30%

84% of the content on OS1 is streamed at an average bitrate of 864 Kbps. This suggests that sessions on OS2 may experience higher video quality than OS1[2]. However, LIVE3 sessions on OS2 have a higher number of bitrate switches per minute as compared to OS1. *This seems to suggest that higher bitrate switching on OS2 could be because trying to stream video at higher bitrate leads to instability as available bandwidth varies.* We also plot the bitrate switches per minute for LIVE1 in Figure 3.7b to show a video service in which the switches per minute are similar on both OS platforms.

Interestingly, we observe a contrasting example in the case of VOD2, where there is similar variation in streamed bitrates across the two OS, but streaming at higher bitrate does not lead to higher bitrate switches. As shown in Figure 3.7c, VOD2 sessions on OS2 are streamed at higher bitrate than sessions on OS1. However, unlike LIVE3, the bitrate switches per minute for VOD2 sessions are lower on OS2 as compared to OS1 (see Figure 3.7d), despite streaming at a higher bitrate on OS2.

These examples suggest that it is not necessarily the network conditions alone, but also player-specific design choices such as default bitrate levels and bitrate adaptation algorithm that significantly impact the QoE of a video session in practice.

Chunk replacement (CR) overhead: We now characterize *Chunk replacement* over-

[2]Video quality is also impacted by the encoding technology used. Some encoding technologies are more efficient than others and can provide higher quality at the same bitrate level

head in different services. Note that chunk replacement can happen due to several reasons, including (i) player trying to improve user experience when network conditions allow, (ii) recovering from various errors or overly aggressive but aborted attempts for high-quality chunks, (iii) ensuring consistent transition between bitrates (frame alignment), and possibly others. This behavior has been observed in previous studies [33], but has not been quantified at large scale. Note that replaced, or otherwise duplicated or repeated chunks, are important from the perspective of both the MNO (represent wasted network resources) and end user (represent waste of a limited data plan). We define CR overhead as the percentage of data in a video session transmitted due to replaced chunks.

Table 3.9 shows the CR overhead for video services in which we could read the chunk identifier and hence detect chunk replacement. A large number of sessions, as high as 80% for UGC3 have non-zero chunk replacement overhead. A non-zero chunk replacement overhead can still be explained by the fact that video players typically start by downloading lower quality chunks to quickly fill up the video buffer and replace them later with high quality chunks, given sufficient network bandwidth. However, a significant number of sessions (26% for UGC3) have chunk replacement overhead greater than 20% which is non-trivial. The overall network overhead due to chunk replacement is quite high (up to 9% for UGC3). This points to the need of looking into optimizing the trade-off between improved user experience and network overhead.

3.4.4 Impact of mobility and demand

The final example use case shows the impact of two important factors on video QoE, namely mobility and per-cell demand, as both are highly relevant to cellular networks.

Impact of user mobility: In a cellular network, user mobility often leads to hand-offs between cells and eNodeBs. While the duration of the hand-off itself is short, the reduced radio signal strength at the edges of a cell can cause degradation in the video QoE. We study the impact of user mobility on video session throughput (ST), which is most directly

Table 3.10: Impact of mobility and demand on session throughput for a subset of network locations

CP	Impact of mobility				Impact of demand	
	$\Delta_{mobility}$		% mobile sessions		Δ_{higher}	
	OS1	OS2	OS1	OS2	OS1	OS2
UGC1S	-6.5%	-6.5%	13.7%	13.7%	-27.8%	-27.8%
UGC2	2.6%	-0.6%	7.4%	5.7%	-16.2%	-23.2%
UGC3S	-1.9%	-1.9%	7.8%	7.8%	-24.1%	-24.1%
UGC4S	13.8%	13.8%	4.4%	4.4%	-43.6%	-43.6%
UGC3	-1.4%	-1.4%	6.6%	6.6%	-19.7%	-19.7%
VOD1	-7.6%	-2.2%	11.3%	14.4%	-7.2%	-2.9%
VOD2	-1.2%	5%	10.8%	15.5%	-32.6%	-24.6%
VOD3	-5.6%	-5.2%	13.6%	17.1%	-41.2%	-35.3%
VOD4	-1.9%	-0.2%	12.3%	17%	-6.2%	-3.5%
VOD5	0.4%	-1.6%	13.7%	17.6%	-1.4%	-3.1%
VOD6	-5.9%	-6.1%	11.7%	18.6%	-19.6%	-17.8%
VOD7	-1%	-2%	9%	15.3%	-2.5%	-22.3%
LIVE1	0.2%	0%	10.9%	20.5%	-2.4%	-3.2%
LIVE2	-0.3%	-0.6%	14.4%	20.3%	-9.2%	-10.4%
LIVE3	0.2%	-13.8%	10.9%	16.2%	-12.4%	-25.8%

affected by varying radio signal.

We infer mobility in a video session by observing the number of eNodeBs in the session. We label a session *mobile* if there were at least three different eNodeBs encountered during that session. We use three eNodeBs as an indicator of mobility instead of two since a stationary UE could be handed-off to a cell in an adjecent eNodeB due to variation in received signal strength. We label the sessions with a single eNodeB as *stationary*, and ignore sessions with two eNodeBs in our analysis. We then calculate the relative change in median session throughput of *stationary* and *mobile* video sessions, referred to as $\Delta_{mobility}$, for every CP:

$$\Delta_{mobility} = \frac{\hat{ST}_{mobile} - \hat{ST}_{stationary}}{\hat{ST}_{stationary}} \times 100\% \tag{3.7}$$

Impact of mobility is shown in Table 3.10 as $\Delta_{mobility}$ with percentage of mobile sessions for the CPs on the two OS platforms. The key high level observation is that the average values of $\Delta_{mobility}$ are low, indicating that user mobility does not significantly impact video QoE. The reduction in ST predominantly in the range of up to 7% should not impact QoE

as the distance between adjacent bitrates is typically by the factor of 1.5 to 2. However, the non-negligible percentage of mobile sessions (up to 20%) might warrant closer inspection to determine precise impact. The relatively low impact (and in some cases improvement) in ST could be attributed to a combination of potential reasons: (i) the video buffer can compensate for the temporary degradation in throughput during hand-off, and (ii) mobile devices are outdoors or in vehicles where radio signal typically has better quality than indoors.

Impact of higher video demand: The last-mile cellular radio link is one of the most challenging network environments, with frequent fluctuation in signal strength, quality and available bandwidth. Therefore, contention for resources is expected, and we examine its impact on video QoE, again considering change in ST ia subset of cells. We use the following heuristic to infer *higher video demand* in a cell. For every cell, we use 15-minute time bins and label the video demand in the cell as either *higher* or *lower*. A cell is considered under higher demand in the current time bin, if:

- *there are at least 10 video sessions during that time bin*, and
- *at least 50% of the video sessions have ST less than the median ST for the corresponding CP*

We compute the relative change in median session throughput of sessions under *higher demand* cells ($\hat{ST}_{higher}$) and sessions under *lower demand* cells ($\hat{ST}_{lower}$).

$$\Delta_{higher} = \frac{\hat{ST}_{higher} - \hat{ST}_{lower}}{\hat{ST}_{lower}} \times 100\% \tag{3.8}$$

This heuristic allows us to consider relative impact of video sessions on each other, and alleviates the need to consider busy hours, radio characteristics, or resource utilization. However, the total load of those cells is not driven by detected video streams, but overall traffic, Table 3.10 shows the *Impact of demand* as Δ_{higher} values for different CPs. Most of the CPs are significantly impacted by higher load with ST dropping by as high as 40%

for UGC4S, but generally in the 20%-30% range. We also observe that the impact is different across CPs. This can be attributed to the difference in encoding bitrates, adaptation algorithms, presence of bitrate control, and known instability issues when adaptive players compete [73]. There is a strong indication that CPs that normally limit the bitrates in cellular such as VOD5 and LIVE1 are not highly impacted by increased demand.

3.5 Summary

In this chapter, we presented MIMIC, a methodology to estimate unencrpyted video QoE metrics from passive network measurements. The results from large-scale validation show that MIMIC can provide a very accurate view of key QoE metrics, namely average bitrate and re-buffering ratio, to a network operator. We also present a measurement study using data collected from deployment of MIMIC in a major U.S. MNO. The measurement study analyzed a variety of content providers on multiple aspects including, relative usage and demand, video service design and its impact on QoE and network usage, and impact of network factors on QoE.

One major limitation of MIMIC is that it is useful mainly for unencrypted video. In the following chapters, we address this limitation by developing inference approaches that work on encrypted traffic. In doing so, we present two complimentary inference approaches with one approach focusing on providing fine-granular and accurate QoE estimation and the other approach focusing on scalability by considering coarse-granular and light-weight network data.

CHAPTER 4

EMIMIC: QOE INFERENCE FOR ENCRYPTED VIDEO

4.1 Introduction

In the last chapter, we presented a highly accurate and practical QoE inference approach 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. We address this challenge in this chapter and present a QoE inference approach, called eMIMIC, for encrypted HAS video. eMIMIC 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 [74]. The key objective of this chapter is to demonstrate feasibility and accuracy of the cross-layer approach to infer service-level QoE metrics from network-level passive measurements.

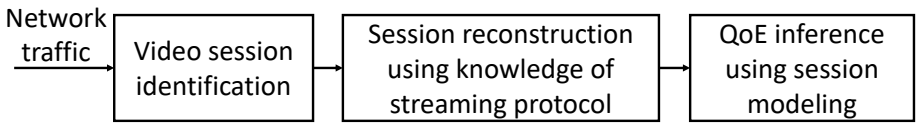
To facilitate the QoE inference from encrypted video sessions, we develop an experimental framework with automated streaming and collection of network traces and ground truth of video sessions, as well as QoE metric estimation. We use this framework to do an extensive evaluation of eMIMIC with three 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 [39], 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.

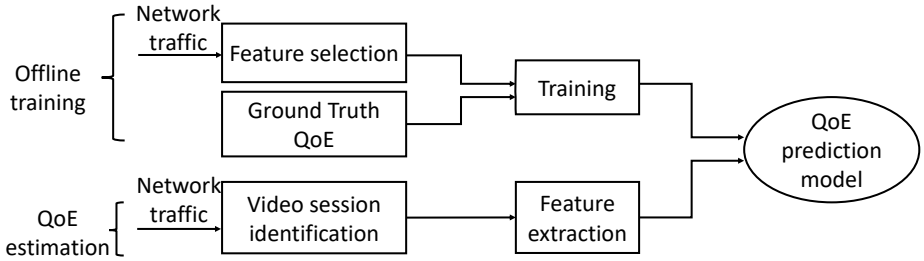
Our contributions are summarized as follows:

- 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 [39] 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 chapter is organized as follows. We begin by describing different categories of QoE inference approaches and design requirements of an ideal approach in Section 4.2. Section 4.3 presents our QoE inference methodology, followed by its evaluation in Section 4.4. Section 4.5 discusses some of the outstanding issues in video QoE inference from passive network measurements, while Section 4.6 concludes the chapter.



(a) Overview of SM-based approach



(b) Overview of ML-based approach

Figure 4.1: Overview of QoE inference approaches

4.2 Background and Design Requirements

4.2.1 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 4.1a). 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 [46].

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 4.1b 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 Chapter 2)

4.2.2 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 [38, 41], logging unencrypted versions of the traffic [39] or using a trusted proxy [40]. 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 4.4.5. 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 [75, 56], 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.

4.3 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.

4.3.1 Chunked video delivery in HAS

The network traffic corresponding to the media chunks in a 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 [19], and this process repeats (Figure 4.2). *The video session at the client can be modeled using this strong serial*

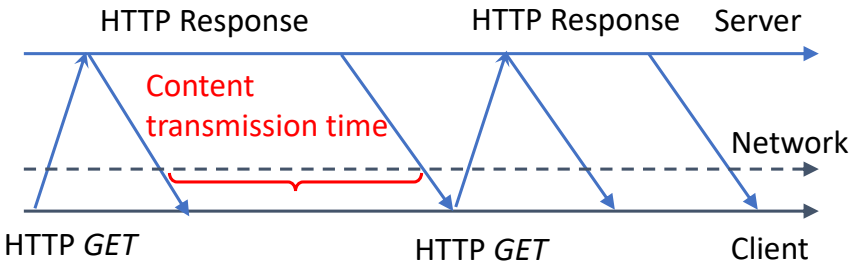


Figure 4.2: Data flow of chunk requests and responses

request-response pattern corresponding to chunk downloads observed on the network.

4.3.2 Challenges in designing eMIMIC

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 4.2 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 [76].

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

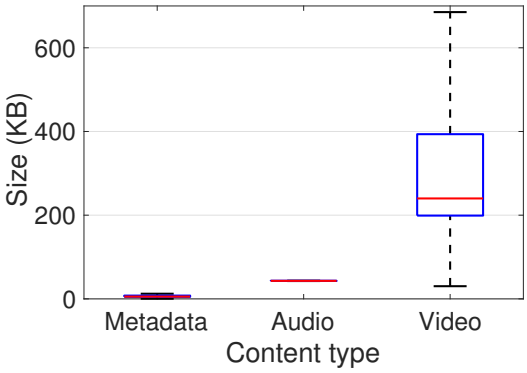
chunks, potentially causing head-of-line blocking, as well as diminishing the ability of the player to quickly adapt to changing network conditions.

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.

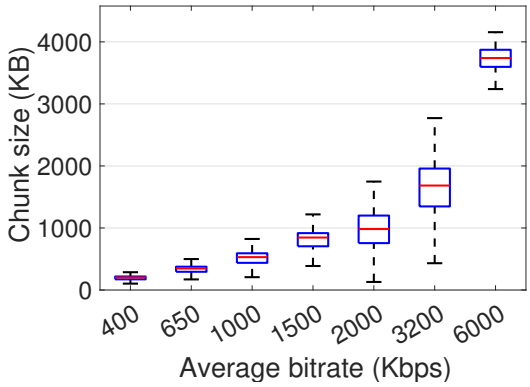
Applying this approach to all TCP flows in a video session, we can reconstruct HTTP transactions, along with the size (S_i) and download start time (ST_i) and end time (ET_i) for every chunk i. This approach can also be applied to UDP-based transport such as QUIC, assuming the same request-response sequence,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

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.



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



(b) Chunk size vs. average bitrate

Figure 4.3: Chunk and bitrate characterization for VoD2.

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 4.3a 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 4.4 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 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.

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 4.3b 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 down-

loaded chunks can improve the accuracy of bitrate estimation of a chunk.

Solution: We first estimate the bitrate of a chunk i using its size blue(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.

4.3.3 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.

$$\hat{BR} = \frac{\sum_{i=1}^{N} \hat{Q}_i}{N} \tag{4.1}$$

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 bluevideo 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:

$$B_i = \max((i - 1 - j + CTS) \times L - (ET_i - ET_j), 0) \tag{4.2}$$

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:

$$b_i = (ET_i - ET_j) - (i - 1 - j + CTS) \times L \tag{4.3}$$

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:

$$\hat{RR} = \frac{\sum_{k=1}^{N} b_k}{N \times L + \sum_{k=1}^{N} b_k} \tag{4.4}$$

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 bluethe total video streamed in minutes and estimate bitrate Switches Per Minute (SPM).

$$\hat{SPM} = \frac{\sum_{i=2}^{N} I(\hat{Q}_i \neq \hat{Q}_{i-1}) \times 60}{N \times L} \tag{4.5}$$

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:

$$ST = T_{loading} + TTNC + T_{decode} \tag{4.6}$$

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.

4.4 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.

4.4.1 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 4.4). We use Java implemen-

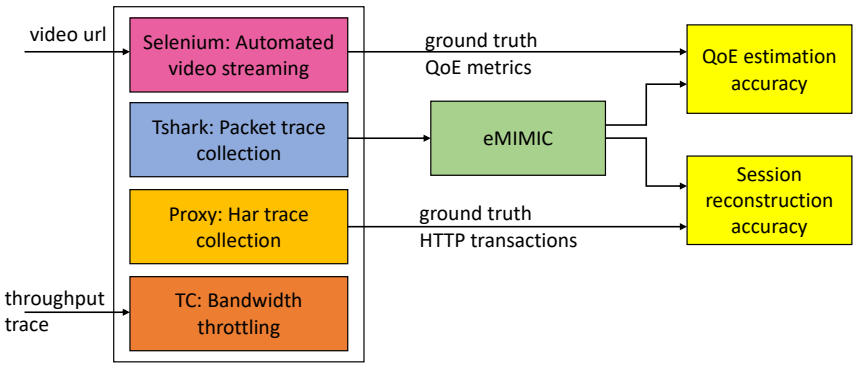


Figure 4.4: Experiment framework and evaluation methodology

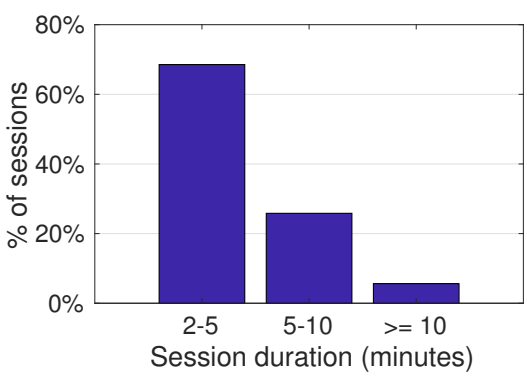
tation of a popular browser automation framework, known as *Selenium*[1]. The HTTP logs of encrypted sessions are collected using a trusted proxy, *BrowserMob proxy*[2], that is easy to integrate with *Selenium*. We use *TShark*[3] for capturing packet-level network traffic and Linux Traffic Control (*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.

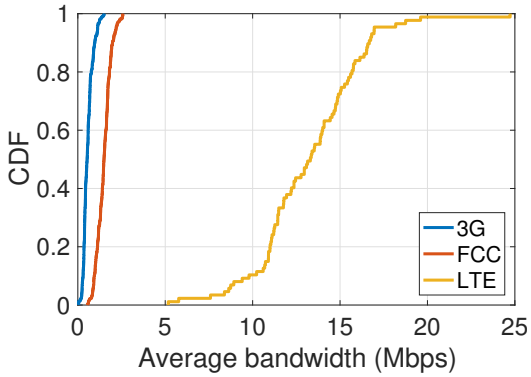
We collected URIs of 100 videos each from both services, covering different genres such as animated videos, talk shows and action movies. The intent was to capture a diversity of content complexities and encoding bitrates. The duration of each session is based on a distribution obtained from the video network dataset collected in [46] and is shown in Figure 4.5a. 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

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



(a) Session duration distribution



(b) CDF of average bandwidth of three dataset

Figure 4.5: Bandwidth traces and session duration

realistic network conditions:

- Norway 3G dataset [77] consists of per-second throughput measurements from mobile devices streaming videos while connected to a 3G/HSDPA network.
- Belgium LTE dataset [78] is similar to Norway 3G but the network is LTE, resulting in higher throughput.
- FCC dataset [79] 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 4.5b shows the CDF of average bandwidth of these traces.

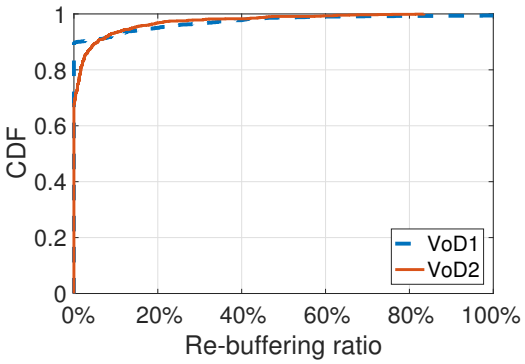
Ground truth QoE metrics: These metrics in video streaming are available within the video player itself. We monitor the player buffer using the JavaScript API exposed by the *Video* element of the HTML5 MSE-based video players of these services. We found two functions, `buffered` and `played`, that return the range of video content that has been buffered and played, respectively. Calling them together enables us to infer the size of buffer. However, it still does not give any information about other QoE metrics such as video bitrate.

Table 4.1: Design parameters of VoD1 and VoD2

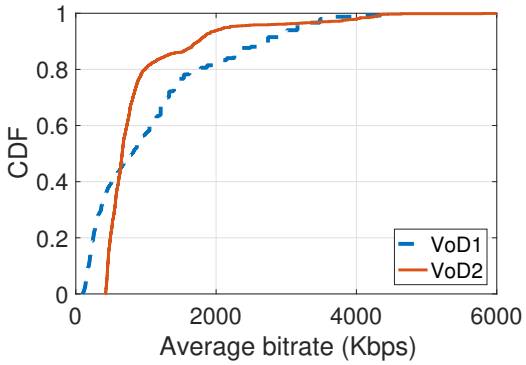
Design parameter	Audio		Video	
	VoD1	VoD2	VoD1	VoD2
Bitrate levels	2	1	10	7
Bitrate range (kbps)	64 - 96	64	100 - 4000	400 - 6000
Chunk duration (s)	16	5	4	5
Chunks to start	1	1	2	1

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.

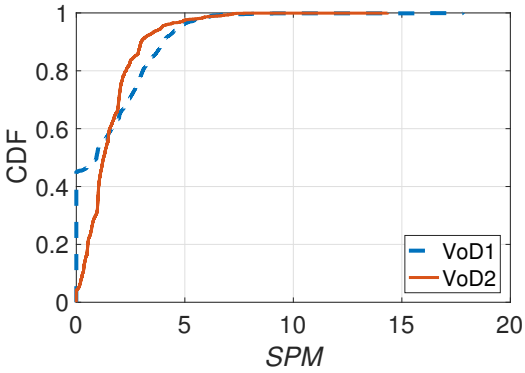
Obtaining video service design parameters: eMIMIC needs to know a few design parameters of a video service. The chunk duration is estimated by playing several video sessions completely and determining the number of chunks downloaded from HTTP logs. Video play time divided by the number of chunks gives 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 4.1



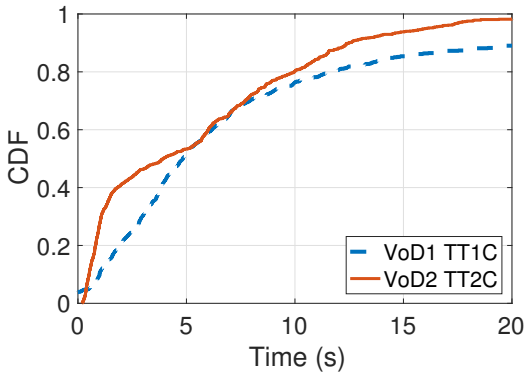
(a) Ground truth re-buffering ratio



(b) Ground truth average bitrate



(c) Ground truth number of bitrate switches per minute



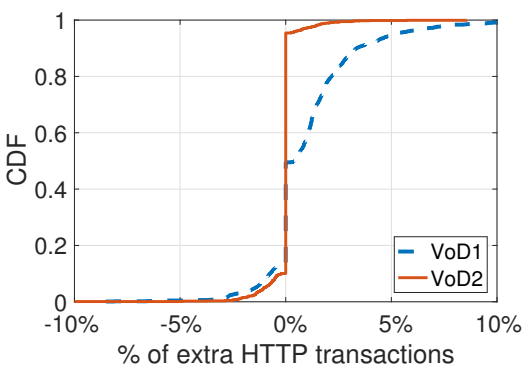
(d) Ground truth $TTNC$

Figure 4.6: CDF of ground truth QoE metrics

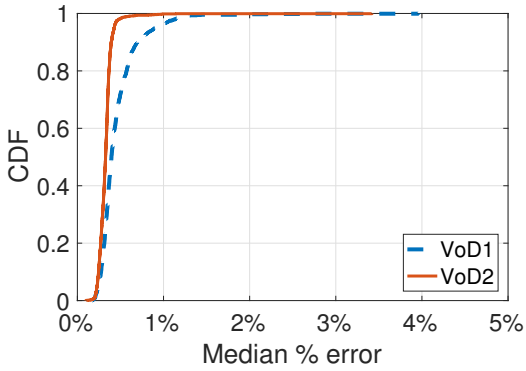
summarizes the values of these design parameters. We note that these design parameters are prone to change for a service which can impact eMIMIC performance. In future, we plan to devise methods to automatically detect these changes.

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



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



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

Figure 4.7: HTTP request reconstruction accuracy

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 4.6 shows the CDF of different ground truth QoE metrics for these sessions.

4.4.2 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.

Table 4.2: Media classification confusion matrix for VoD1

(a) With A/V buffer tracking

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

(b) Without A/V buffer tracking

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

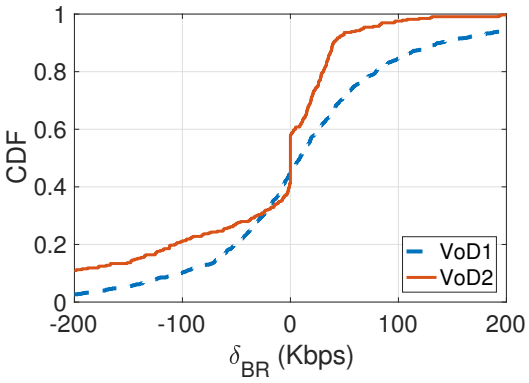
Figure 4.7a shows a CDF of percentage of extra transactions (negative value denotes missing transactions) in a session. We find that the accuracy of reconstruction is high with 80% of sessions from VoD2 reconstructed with 100% accuracy. The lower accuracy of reconstruction for VoD1 is because few *metadata* transactions in VoD1 are comparable in size to *video* and get misclassified as *video*.

Figure 4.7b 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.

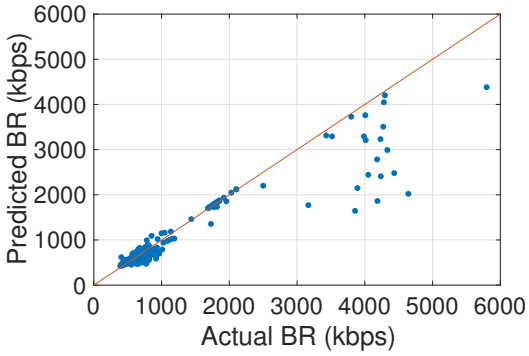
4.4.3 Media type classification accuracy

Table 4.2a 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 4.2b) when the A/V classification is done only using the size of the HTTP transaction. Tracking A/V buffer (Table 4.2a) helps in reducing the error of misclassifying video chunks as audio by 1.26% without significantly



(a) CDF of error in average bitrate estimation



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

Figure 4.8: Error in average bitrate estimation

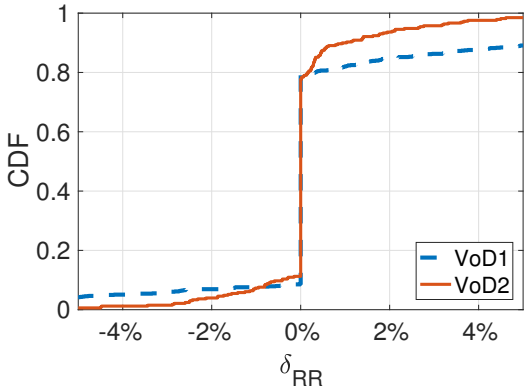


Figure 4.9: CDF of error in re-buffering ratio estimation

impacting the error in misclassifying audio chunks as video.

4.4.4 QoE inference accuracy

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

Average bitrate: Figure 4.8a 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 zero error in VoD1 partially because we do not know the exact values of bitrate levels and use approximate values instead.

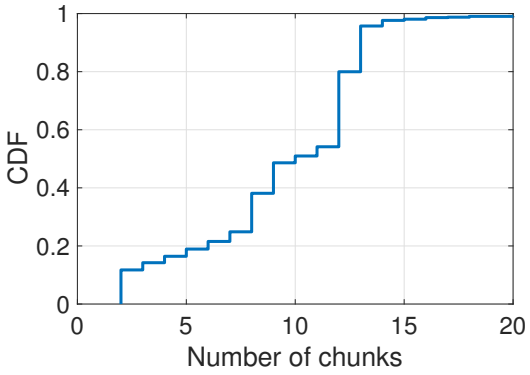


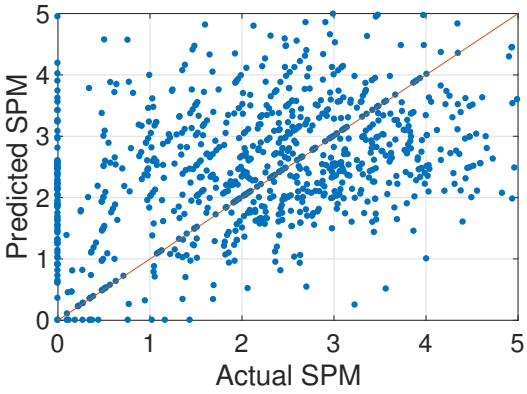
Figure 4.10: CDF of of chunks in the buffer at startup

Figure 4.8b 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.

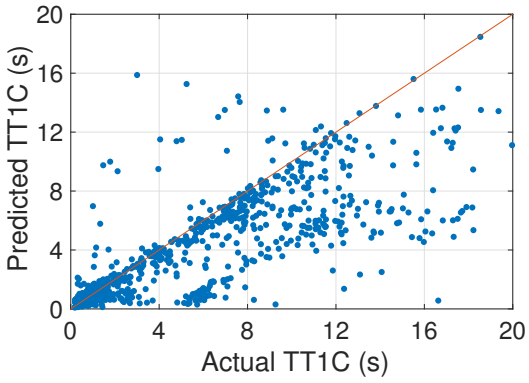
Re-buffering ratio: We calculate the difference (δ_{RR}) between the estimated re-buffering ratio and ground truth re-buffering ratio. Figure 4.9 shows a CDF of δ_{RR} for VoD1 and VoD2. We see that eMIMIC can predict re-buffering ratio with a high overall accuracy, i.e., within an error of 1% for around 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 4.10 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 ($48s$ 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.

Bitrate switches: Figure 4.11a 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



(a) VoD1: Scatter plot of SPM



(b) VoD2: Scatter plot of TT1C

Figure 4.11: Error in estimating the bitrate switches and startup time

Table 4.3: 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%

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 4.11b 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 [40] 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.

Table 4.4: 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%

Table 4.5: 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%

4.4.5 Comparison with ML-based approach

Here, we compare eMIMIC with ML16, an ML-based approach described by Dimopoulos et al. [39]. 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 [80] 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 [81].

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

Table 4.6: 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

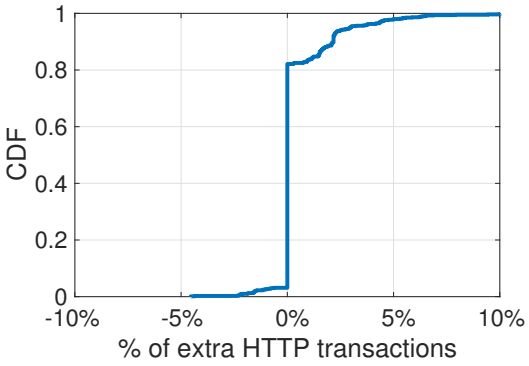
kbps and 1400 kbps are chosen to classify sessions of VOD1 into *low*, *med* and *high*. The overall classification accuracy of eMIMIC is slightly higher (around 3%) than ML16 (row 1 of Table 4.3). Table 4.4 shows the confusion matrix of bitrate classification for VOD1. eMIMIC identifies *low* and *med* 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 [39]): *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 4.3 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 4.5 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.

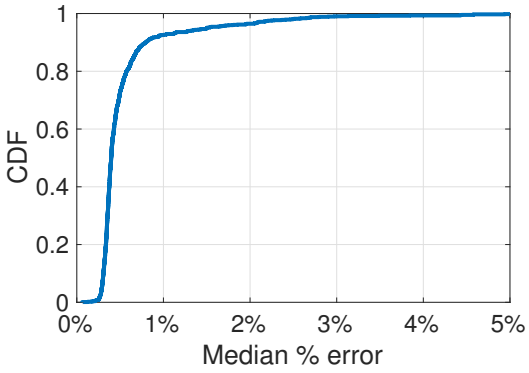
Finally, we test if ML16 generalizes across services by using the ML16 model learned for VOD2 to estimate re-buffering ratio for VOD1. The classification accuracy of the model dropped to 31% on VOD1 from 61% on VOD2. This shows that ML16 does not generalize and needs separate training for each service whereas eMIMIC faces no such issues.

4.4.6 QoE inference accuracy for a Live service

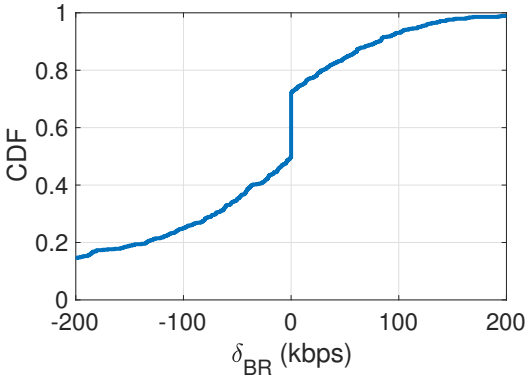
Live streaming has been growing over the last few years [27]. Live video differs from VoD in terms of few key design parameters, i) the buffer in Live streaming is small to reduce the



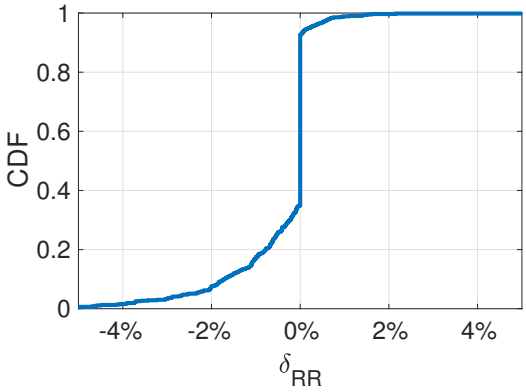
(a) CDF of % extra requests in the reconstruction



(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

Figure 4.12: LIVE1: Session reconstruction and QoE metrics estimation error

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. This may 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 4.6 shows the design parameters for LIVE1, obtained by the methodology described in Section 4.4.1. 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 that displays a box with different session 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 4.12a shows the CDF of percentage of extra transactions (calculation methodology described in Section 4.4.2) in a session. The accuracy of reconstruction is high in general with 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 4.12b).

QoE estimation accuracy: Figure 4.12c 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 4.12d. 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.

Table 4.7: 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 4.8: 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%

4.4.7 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 [82]. In this section, we evaluate the accuracy of eMIMIC in making such real-time prediction of QoE metrics. We limit out analysis to sessions from VoD1. Our methodology, however, works for other services as well.

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.

Table 4.9: 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%

We first evaluate the impact of varying T_{class} using 20 seconds as the buffer occupancy threshold. Table 4.7 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. The overall 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 (56.5% - 82.9%), while the recall is high (97.3% - 98.0%). 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 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 4.8). 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

Table 4.10: 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%

(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 streaming *low* quality video chunks. We evaluate the accuracy of eMIMIC in classifying the average bitrate category 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 4.9 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 of classification increases as the number of most recently downloaded chunks considered are increased. 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 4.10). 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, the error in individual chunk bitrate estimation reduces as the chunk quality increases, ultimately leading to an increase in classification accuracy when a higher bitrate threshold is used. This shows that eMIMIC can estimate average bitrate class of most recently downloaded chunks with high accuracy, precision, and recall.

4.5 Discussion and Future work

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

4.5.1 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 [83]. 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 net-

work 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.

4.5.2 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 re-calibration 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 [84], a UDP-based protocol, requiring re-calibration 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 [85] 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 4.3.2). 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.

4.5.3 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.

4.6 Conclusion

In this chapter, we presented 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 also 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. Finally, we highlight some of the outstanding issues and challenges in QoE estimation. In the next chapter, we focus on one of these challenges, i.e., scalability, and present an approach that uses light-weight network data for QoE inference.

CHAPTER 5
INFERENCE USING COARSE-GRAINED DATA

5.1 Introduction

Video QoE estimation using network data primarily consists of three steps: i) collecting network data using a monitoring tool, ii) identifying video traffic and sessions from collected data, and iii) estimating session QoE metrics using methods designed for this purpose (see Figure 5.1). In the previous chapter, we mainly focused on designing QoE estimation mechanism (step 3 in Figure 5.1) for encrypted traffic with a goal to improve inference accuracy. In doing so, we assume access to packet traces, the most granular network data. However, collecting and processing packet-level data from the entire network can be challenging because of the scale of ISP networks. At the same time, it is important for ISPs to understand network-wide video performance for efficient management and provisioning, especially in the case of capacity-constrained and highly heterogeneous cellular networks. This makes it challenging to use existing QoE estimation mechanisms in practice.

One possible approach is to develop flexible telemetry systems that provide the most useful metrics (e.g., HTTP transactions) required for inference by in-network processing of the packet data [49, 86, 48]. While this is a viable approach, it involves significant modifications to the existing measurement systems and has the following practical challenges, i) limited measurement resources and budget with the constraint that the same network data is often used for multiple purposes (e.g., security, performance), and ii) limited flexibility as the monitoring tools are provided by vendors [87].

Given these challenges, we ask: "Is it feasible to detect video performance issues with lightweight, readily-available but coarse-grained network data?" Our question is motivated by the fact that ISPs already collect coarse-grained data using standard telemetry systems

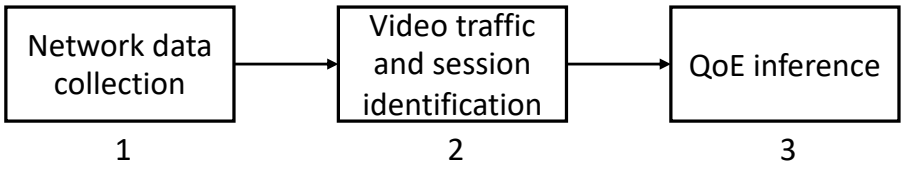


Figure 5.1: QoE inference steps

for different network management functions [88, 89, 87]. We consider whether such data can be used by ISPs to estimate coarse-grained QoE metrics (e.g., *low*, *high*) and thus to identify parts of the network that underperform in a lightweight manner. Ultimately, this approach can enable adaptive video performance monitoring wherein an ISP collects fine-grained data only from the problematic locations for further diagnosis.

We specifically consider coarse-grained network data in the form of Transport Layer Security (TLS) transactions. The data is clearly lightweight as number of TLS transactions in a video session are significantly smaller (by a factor of 1400 in our dataset) as compared to packets. The data is also readily available as TLS transactions can be collected using a transparent proxy (e.g., Squid [90]). Moreover, video traffic can be easily identified (step 2 in Figure 5.1) using the headers from TLS transaction data. Prior work has used similar data to infer QoE for web traffic [91] and unencrypted video[1] [69]. A major challenge, however, is that the TLS transaction data is coarse-grained. Thus, existing inference techniques will not work on this form of data. Another challenge in using this data is to delimit sessions[2] when a user watches back-to-back videos from the same service. Accurate session identification is important for accurate QoE estimation due to changes in streaming patterns and the corresponding traffic within a session as it progresses (see Section 5.2).

Therefore, we analyze the feasibility of using TLS transaction data to detect video performance issues. Specifically, we consider categorical estimation of key video QoE metrics [16, 2], namely, video quality, re-buffering ratio and a combined QoE metric that jointly considers the two individual metrics (Section 5.2). We first develop a machine learn-

[1]For unencrypted video, a proxy provides HTTP transactions
[2]Our definition of a session consists of streaming a single video

ing (ML)-based approach that builds on previous work by adapting ML-based techniques to TLS transaction data. We evaluate our methodology using data collected under diverse emulated network conditions from three streaming services, namely, YouTube, Netflix, and Hulu (anonymized in the paper). We also compare the QoE estimation accuracy using TLS transaction data against packet traces. Finally, we present a simple heuristic to distinguish consecutive sessions from the same video service leveraging TLS transaction arrival and server access patterns.

Our key findings are summarized below:

- The TLS transaction data can be used to estimate combined QoE metric (Section 5.2) with an accuracy of up to 72% and detect *low* QoE (*low* video quality or *high* re-buffering) instances with a recall of 73%-85%.

- Compared to packet traces with an existing ML-based approach [39], estimation using TLS transaction data has up to 7% (9%) lower accuracy (recall), but it has 1400x lower memory overhead and 60x lower computation overhead.

- The session identification heuristic can accurately identify 89% of the consecutive sessions.

The rest of the chapter is organized as follows: Section 5.2 describes the QoE metrics that we infer and the network data used for their inference. Section 5.3 presents the QoE estimation methodology with Section 5.4 presenting the results. Section 5.5 discusses some outstanding issues and Section 5.6 concludes the chapter.

5.2 Target QoE and Network data

Here we describe the HAS QoE metrics we estimate and the network data used for their inference.

5.2.1 Target QoE metric

Existing approaches estimate the objective video QoE metrics using network data in two different ways : *fine-granular* and *per-session*. The former estimates QoE metrics within a session at periodic intervals while the latter provides estimates only once for the entire session. The estimation granularity of an approach is clearly impacted by the granularity of the input network data. Our goal in this work is to enable ISPs to identify video performance issues in a light-weight manner. Given the coarse-granular nature of the data that we use for this purpose, we estimate categorical values (e.g., low, medium, and high) of *per-session* video QoE metrics. We envision an adaptive video performance monitoring, wherein an ISP can collect fine-granular network data for further diagnosis from the locations identified using coarse-granular but light-weight estimation. Thus, we estimate the two most important video QoE metrics, namely *rebuffering ratio* and *video quality* [16], defined as follows:

Re-buffering ratio: We use *re-buffering ratio* (*rr*) to measure the severity of video stalls. It is defined as the stall time in proportion to the total playback time. We categorize *rr* into the following three categories: i) *zero*, if there are no stalls, ii) *mild*, if $0 < rr < 2\%$, and iii) *high*, otherwise.

Video quality: In HAS, videos are typically encoded into discrete quality levels which tend to be the same for a video service (e.g., Netflix, YouTube) and streaming protocol (e.g., HLS, DASH) combination with some minor exceptions[3]. We set thresholds and categorize the quality levels to *low*, *medium*, and *high* (see Section 5.4). The *video quality* of a session is simply the majority category of the video played in a session [61]. In case of a tie, we select the lower category.

Combined QoE: We also estimate the *combined QoE* of a session by jointly considering the impact of individual QoE metrics. There are a number of ways to combine the individ-

[3]Some videos may not be available at all (especially higher) quality levels. A service may use different quality levels depending on the content type (e.g, live vs on-demand video).

ual metrics [72, 19]. Our methodology can work for multiple such combinations. In this work, we use a simple approach of using the minimum category of the two QoE metrics. For instance, if a session had *zero* re-buffering but *low* video quality, its overall QoE is assigned to *low*.

Thus, for each session we estimate the categorical values of *video quality*, *re-buffering ratio*, and *combined QoE*.

5.2.2 Network data

ISPs typically collect different kinds of data from within their network using standard monitoring tools which is then used for various network management functions [87]. We now consider the suitability of different network data for video QoE inference. The collected data includes network device-level data (e.g., SNMP logs [92] or radio-level data in cellular networks [93]) and passive traffic monitoring data (e.g., packet-level traces or aggregate traffic statistics). Clearly, device-level data cannot be used to even identify video traffic, let alone assess end-user video QoE. We now consider data obtained from passive monitoring.

With passive monitoring, packet-level data is the most detailed data available to ISPs. However, ISPs typically do not collect packet traces on a network-wide scale due to high processing and storage overhead. It is important for ISPs to understand network-wide video QoE for network management functions like capacity planning. This is especially required in the case of cellular networks which tend to be capacity constrained and highly heterogeneous. Therefore, we consider using light-weight network traffic data that can be collected with standard monitoring tools for QoE inference.

Specifically, we consider network traffic data in the form of TLS transactions. For encrypted traffic, a standard web proxy similar to Squid [90] can be used for logging the TLS transactions by inspecting the unencrypted TLS headers. The TLS transaction data is clearly light-weight as the number of TLS transactions are significantly lower compared to packet traces. A major challenge, however, is that the TLS transaction data is coarse-

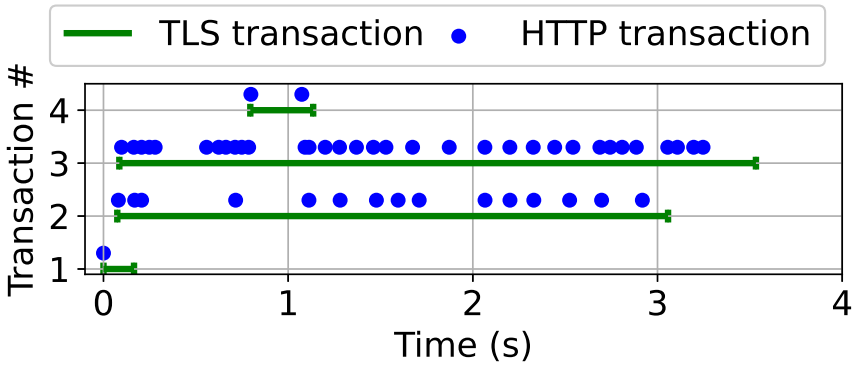


Figure 5.2: TLS transactions with the corresponding HTTP transactions within first 5 seconds of a Svc1 session. For clarity, only start of the HTTP transactions is shown.

granular. Figure 5.2 shows the TLS transactions within the first 5 seconds of a sample session from Svc1 with the corresponding HTTP transactions[4]. Note that the HTTP transactions are derived from packet traces [40]. Clearly, a single TLS transaction contains multiple and variable number of HTTP transactions. We observed an average of 12.1 HTTP transactions corresponding to every TLS transaction for the Svc1 sessions in our dataset (see Section 5.4). *Our goal is to analyze the feasibility of using this coarse-granular but readily-available and light-weight data to estimate video QoE.* We consider two kinds of information available in a TLS transaction: i) timings and size information, and ii) Server Name Indicator (SNI) field indicating the server's hostname. We use the former for QoE estimation and the latter for identifying the video traffic.

We note that flow-level monitoring (e.g., NetFlow [94]) is another popular measurement technique. In our dataset, we observed a single TLS transaction for every TCP connection. Thus, collecting flow records with size counters in NetFlow can effectively provide TLS transaction data. In addition, flow-level monitoring also provides the option of obtaining periodic summaries from long flows. A major challenge, however, with flow-level moni-

[4]Features derived from HTTP transactions are typically the most important features in the related work for QoE inference.

toring is identification of video traffic as it lacks application-layer data. Existing work has suggested solutions such as augmenting flows with DNS information [95]. We consider using such light-weight flow data as a part of future work and focus here on understanding feasibility of TLS transaction data for inference.

5.3 Methodology

We formulate the QoE estimation problem as a supervised machine learning problem. We assume that the TLS transactions corresponding to video traffic have already been identified (e.g., using SNI field) and grouped into sessions. Later, we also present a heuristic to delimit TLS transactions corresponding to consecutive sessions from the same service. Here, we describe the features that we extract from the sequence of TLS transactions corresponding to a session.

For every transaction in the session, we have its downlink data size, uplink data size, start time, and end time. We use this data to construct the following three kinds of features:
Session-level: These features consist of metrics calculated for the entire session. We calculate the session data rate, which is the total data divided by the session duration, in both downlink (SDR_DL) and uplink (SDR_UL) directions. In addition, we also log the session duration (SES_DUR) and the number of TLS transactions per second ($TRANS\_PER\_SEC$).
Transaction statistics: For a transaction, we already have its downlink size (DL_SIZE), uplink size (UL_SIZE), and duration (DUR). We collect some additional features for each transaction. First, we calculate Transaction Data Rate (TDR), which is obtained by dividing the downlink data size by the transaction duration. Note that TDR is not the same as network throughput as there can be intervals (e.g., *steady state* in HAS [73]) in a TLS transaction with no network activity. However, it is still an indicator of the network quality as, intuitively, TDR, is high if the available bandwidth was high. Second, we calculate Downlink-To-Uplink ($D2U$) Ratio, which is the ratio of the downlink data to the uplink

data. In HAS, the uplink data is typically an indicator of the number of video segments requested [42]. Hence, $D2U\ ratio$ represents the amount of data downloaded per segment. This can be a useful indicator of the video quality. Finally, we calculate the Inter-arrival time (IAT) of the transactions to capture patterns in arrival of transactions. Thus, we have 6 features for each transaction. From these features, we generate summary statistics, namely, minimum, median, and maximum value leading to 18 features in total [5].

Temporal Features: These features capture the temporal progress of data transfer during a session. We divide the session into pre-determined intervals each starting from the beginning of the session and calculate the cumulative downlink (CUM_DL_XXs) and uplink data (CUM_UL_XXs) during each of these intervals. For transactions that only partially overlap with an interval, we get its the share of downlink and uplink data based on the extent of the overlap with the interval[6]. This set of features can be useful in uncovering any temporal variations which may have been masked out in the aggregate transaction statistics.

We consider the following end-points for the intervals (in seconds): $\{30, 60, 120, 240, 480, 720, 960, 1200\}$. We use a maximum value of 1200 seconds as this is the maximum session duration in our dataset (see Section 5.4). The rationale behind using fine-granular intervals in the beginning is that a session is more likely to be impacted by poor network quality in the beginning because of empty video buffer. We explored other intervals (omitted due to lack of space) but found the above to yield the highest accuracy. Regardless, we consider these intervals as one of the hyperparameters of our model and an ISP can determine the intervals based on the data observed on their network for a service.

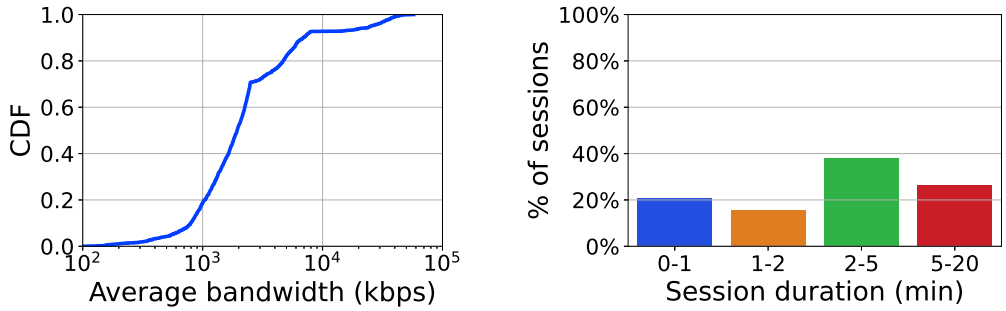
Thus, we have a total of 38 features for each session (summarized in Table 5.3). We use these features to estimate the QoE metrics of the session.

[5]We considered other statistics such as standard deviation and mean, but found them to be highly correlated to one of the existing statistics.

[6]This is an approximation as it is not possible to figure out the data transmission pattern within a transaction

Table 5.1: Summary of features

Type	Statstic	Features
Session level	single value	SDR_DL, SDR_UL, SES_DUR, TRANS_PER_SEC
Transaction Statistics	min, median, max	DL_SIZE, UL_SIZE, DUR, TDR, D2U, IAT
Temporal Statistics	interval based	CUM_DL_XXs, CUM_UL_XXs



(a) Average bandwidth

(b) Duration

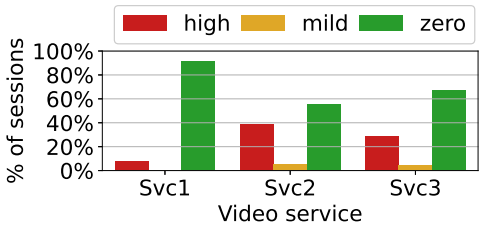
Figure 5.3: Bandwidth traces statistics

5.4 Evaluation

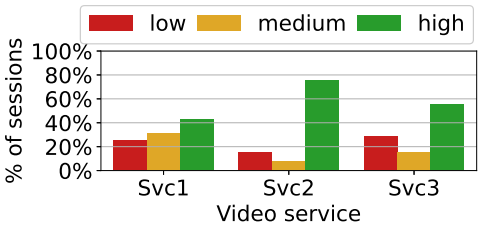
In this section, we describe the results evaluating the estimation accuracy using TLS transaction data and comparing it with packet traces. We first describe the methodology to collect the dataset used for evaluation.

5.4.1 Data collection

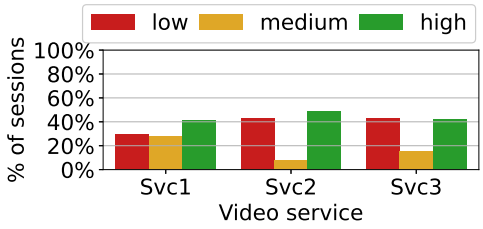
We use a browser-based automation framework to collect data for training and testing. The framework streams video sessions under emulated network conditions and collects network data in the form of packet traces and TLS transactions. We emulate network conditions using publicly available bandwidth traces representing a diversity of network environments including 3G, fixed broadband, and LTE [77, 78, 79]. Each session is streamed for a predetermined duration which ranges from 10 seconds to 20 minutes. Figure 5.3a and 5.3b show the distribution of average bandwidth and duration of the traces, respectively.



(a) Re-buffering ratio



(b) Video quality



(c) Combined QoE

Figure 5.4: Distribution of QoE metrics across services

Using the above methodology, we collect data for three popular streaming services, denoted as Svc1, Svc2, and Svc3 (anonymized for confidentiality). We curate a list of 50-75 videos for each service that includes content from a variety of genres such as animation, sports, and news, if available. The ground truth video QoE metrics are collected by injecting Javascript functions that utilize the HTML5 VIDEO API for monitoring buffer level and service-specific functions (determined manually) for monitoring video quality [48]. We then classify the video quality levels into one of the three categories. We use resolution-based thresholds in Svc1 and Svc2 as these services had a unique resolution for each quality level. For Svc2, we classify video resolution of *360p* or lower as *low*, *480p* as *medium*, and *720p* or higher as *high*. The thresholds for Svc1 were *288p* for *low*, *480p* for *medium*, and the remaining levels were classified as *high*. For Svc3, we observed only three quality levels in our dataset and classify them into *low*, *medium*, and *high*. In practice, these thresholds can be set by the ISP based on its target quality. We use the per-second QoE

Table 5.2: Confusion matrix: Svc1, Combined QoE

Actual	# sessions	Predicted		
		low	med	high
low	632	72%	21%	8%
med	599	25%	43%	32%
high	880	5%	12%	84%

Table 5.3: Accuracy (A), Recall (R), and Precision (P) values for different feature sets

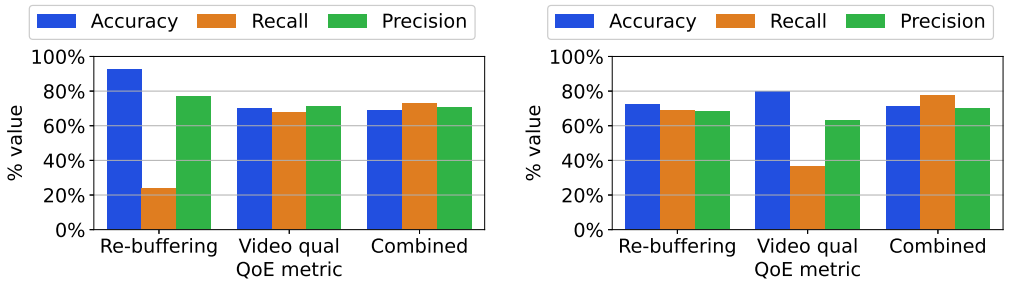
Feature set	Svc1			Svc2			Svc3		
	A	R	P	A	R	P	A	R	P
Only Session-level (SL)	58%	61%	60%	66%	68%	63%	66%	77%	66%
SL + Transaction Stats (TS)	65%	72%	67%	69%	77%	68%	71%	84%	74%
SL + TS + Temporal Stats	69%	73%	71%	71%	78%	71%	73%	85%	75%

information to obtain categorical values of per-session video quality, re-buffering ratio, and combined QoE.

Overall we had $2,111$ sessions for Svc1, $2,216$ sessions for Svc2, and $1,440$ sessions for Svc3. Figure 5.4 shows the distribution of ground truth QoE metrics for the three services. We observe difference in QoE metrics across services streamed under similar network conditions. This can be attributed to differences in service design. We found that Svc1 uses a larger video buffer (240s) as compared to the other two services. Furthermore, Svc1 player attempts to avoid re-buffering by quickly filling the buffer at the expense of streaming at low video quality. However, the other two services, especially Svc2, switch video quality only when the video buffer runs low. Therefore, poor network conditions led to low video quality in Svc1, whereas in Svc2 and Svc3 (although to a lesser extent), it led to re-buffering.

5.4.2 Results

We use the Python Scikit library to train and test different machine learning models. We use 5-fold cross validation for evaluating the accuracy of the models. We explore different ML-based models, namely SVM, k-NN, XGBoost, Random Forest, and Multi-layer Perceptron.



(a) Svc1 (b) Svc2

Figure 5.5: Accuracy for different QoE metrics

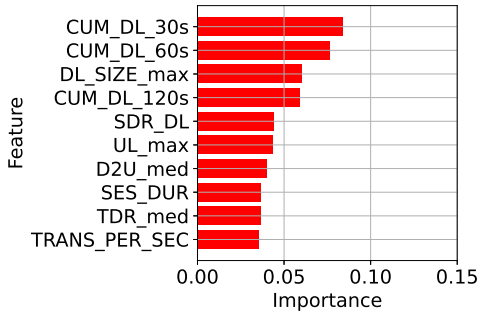
Here, we present results using Random Forest which yielded the highest accuracy.

Accuracy for different QoE metrics: Figure 5.5 shows the classification accuracy of different QoE metrics in Svc1 and Svc2. We particularly focus on the recall value of *low* QoE metric class as one of our main goals is to detect video performance issues. For Svc1, our model has 70% recall value in identifying *low* video quality sessions, while the recall is only 21% in identifying *high* re-buffering sessions (see Figure 5.5a). For Svc2, the trend is reversed and our model can detect *high* re-bufferring with a 71% recall and *low* video quality with only 40% recall (see Figure 5.5b). The results are similar for Svc3 with a recall of 63% in detecting *high* re-buffering and only 58% in detecting *low* video quality. We find that the classification accuracy is high for QoE metrics that are more likely to degrade with poor network conditions in a video service.

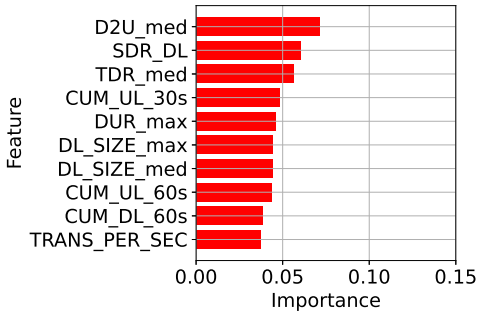
We now focus on the classification accuracy for combined QoE. Our model can detect sessions with a *low* combined QoE with high recall (73%-85%) across all three services. Table 5.2 shows the confusion matrix for classifying the combined QoE in Svc1. Most of the mis-classifications happen between neighboring QoE classes (e.g., *low* classified as *med*). This is most likely due to the model's inability in classifying instances that are closer to the class thresholds. Naturally, the error is higher for sessions with *medium* QoE, while the sessions with *low* or *high* combined QoE can be classified with a high accuracy across all three services.

Table 5.4: Accuracy metrics using packet traces and ML16. The numbers in paranthesis report the gain as compared to the TLS transaction data.

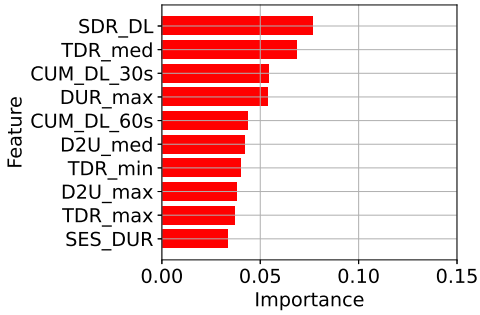
Service	Accuracy	Recall	Precision
Svc1	74% (+5%)	82% (+9%)	73% (+2%)
Svc2	78% (+7%)	85% (+7%)	76% (+5%)
Svc3	78% (+5%)	89% (+4%)	78% (+3%)



(a) Svc1



(b) Svc2



(c) Svc3

Figure 5.6: Top 10 important features across three services

Takeaway: The coarse-granular but light-weight TLS transaction data can enable ISPs to detect video performance issues *aka low* combined QoE sessions with a high accuracy. ISPs can then collect fine-granular data for parts of the network with issues for a fine-granular QoE estimation and troubleshooting. In the remaining chapter, we focus on results pertaining to combined QoE.

Feature importance: We next evaluate the impact of different kinds of feature on model accuracy. Table 5.3 shows the model accuracy for different sets of features. The accuracy (recall) is lowest when only session-level features are used and it improves by 6%-11%

Table 5.5: Transaction identification accuracy into Existing or New session

Actual	# Transactions	Predicted	
		Existing	New
Existing	13269	98%	2%
New	1545	11%	89%

(6%-12%) as features capturing the transaction statistics and temporal distribution of data are added to the model. This shows that despite being coarse-granular, TLS transactions within a session can provide useful information about the QoE of a session.

Figure 5.6 shows the 10 most important features as reported by the Random Forest model across the three services. There are 4 features that appear in the top 10 list of all three services. These features are downlink session data rate (SDR_DL), median transaction data rate (TDR_MED), Median D2U ratio ($D2U\_MED$), and the cumulative downlink data in the first minute (CUM_DL_60s). TDR_MED and SDR_DL represent the downlink data rate which is likely to correlate with the available bandwidth. $D2U\_MED$ represents the downlink to uplink data ratio and is likely to be higher when the video quality is high and vice-versa. Finally, CUM_DL_60s represents the data downloaded in the beginning of the session when the video buffer is usually low and when a session is more likely to suffer if the network conditions are poor. We also observe differences across services with 10 features that appear in only one out of the three services. This is likely due to the differences in TLS transaction mechanisms across services.

Takeaway: The analysis shows that in addition to session-level metrics such as duration and data rate, there are also patterns in the TLS transactions within a session that differ based on the session QoE. An ML-based approach can learn these patterns to identify *low* QoE sessions.

Comparison with packet traces: We now compare the QoE estimation accuracy from TLS data against packet traces. There are many ML-based algorithms that estimate QoE using packet traces. However, most of them estimate fine-granular QoE metrics within a session. While it is possible to get per-session QoE metrics from fine-granular estimation,

we decided to use a heuristic that was designed specifically to estimate per-session QoE metrics. Specifically, we implement the algorithm proposed by Dimopoulos et al. [39], called ML16. The algorithm uses features corresponding to video segments as well as traditional network metrics such as re-transmissions and RTT. ML16 uses different sets of features for estimating re-buffering ratio and video quality. We use the feature set corresponding to video quality in ML16 for estimating the combined QoE as it is a super-set of the features used to estimate re-buffering.

Table 5.4 shows the accuracy metrics and the respective gains in comparison to TLS transaction data. Using features derived from packet traces with ML16 results in overall accuracy improving by 5%-7%, recall by 4%-9%, and precision by 2%-5%. This is intuitive as packet traces are highly fine-granular. Moreover, they can be used to derive information about video segments downloaded in a session which are fundamental to HAS and its QoE. However, using packet traces has an associated memory and computation overhead [7]. E.g., the average number of packets across sessions in Svc1 are $27,689$ as compared to only 19.5 TLS transactions. The total computation time of extracting relevant features from all sessions for the packet data is around 503 seconds as compared to 8.3 seconds using TLS transaction data, a difference of factor of 60.

Takeaway: Packet traces provide higher accuracy than the TLS transaction data but with a significant computation and memory overhead. Therefore, ISPs can implement adaptive network monitoring, wherein fine-granular network data is collected only once any performance issues are detected.

Session identification heuristic: Recall that identifying video traffic and sessions is an important step prior to QoE estimation. However, it can be challenging to correctly delimit session boundaries using TLS transaction data if multiple videos from the same service are watched back-to-back by a user. Even a timeout-based approach, wherein a session boundary is detected if there is no more video traffic for a certain time, may not always

[7] A significant portion of this overhead can be mitigated by deriving relevant features via in-network processing. However, this requires ISPs to deploy new network monitoring tools.

work because the active TLS transactions do not always immediately end once the player or the web page is closed. Instead, they time out after some period leading to overlapping transactions for back-to-back sessions. Existing work suggest heuristic based on the packet or segment-arrival pattern which will not work with TLS transaction data. We develop a simple heuristic for session detection which is based on the following two insights: i) The beginning of a session is characterized by more than one TLS transaction, and ii) More often that not, the set of servers serving content change when a new session begins. Thus, for each transaction we consider the set of succeeding transactions with a start time within W seconds. Using these set of transactions, we calculate N, the number of transactions in the set, and δ, the percentage of transactions with a different server than the set of servers seen for the current session. A transaction is considered to start a new session, if N and δ are greater than N_{min} and δ_{min}, respectively. We use the following parameter values, W = 3 seconds, N_{min} = 2, and δ_{min} = 0.5.

Table 5.5 shows the confusion matrix for Svc1 sessions. The session identification accuracy is high with 89% sessions are identified correctly. For about 3% of the incorrectly classified *new-session* transactions, the heuristic missed by only one transaction. In comparison, a timeout-based heuristic would have considered all of them as a single session as all of these sessions are streamed back-to-back. We note that this is an extreme case in comparison to real-world scenario.

Takeaway: Session identification is an important step for QoE estimation and needs to be designed for the specific network data. The difference in transaction arrival and server request pattern can enable session identification with high accuracy for TLS transaction data.

5.5 Discussion

Machine learning techniques: We use supervised machine learning along with input features engineered based on the knowledge of HAS protocol. There can also be other ways

of using machine learning techniques. One potential approach within supervised machine learning can be to use deep learning. One of the advantages of using deep learning is that it does not require feature engineering. Deep learning-based models can be easier to adapt to changes in streaming service protocols as the adaptation would involve only re-training the models using new ground truth QoE data. However, there are also challenges in using deep learning. First, it generally requires more data with ground truth QoE for training compared to classical supervised ML techniques which can be challenging for ISPs. Second, deep learning-based models generally lack interpretability which may be useful for ISPs.

Comparison using packet traces: We use packet traces along with an ML-based approach that provides per-session categorical estimates of QoE metrics for comparison against TLS transaction data. There are other approaches that use packet traces for QoE estimation including eMIMIC that can provide more fine-grained QoE estimation. One of the reason we did not use eMIMIC is because one of the streaming services used for evaluation requests variable number of video segments in a single HTTP transaction, thus making it challenging to model the session. Using the ML16 approach allows us to compare accuracy for all three services.

Flexible monitoring systems: Using flexible monitoring systems can potentially allow ISPs to use more accurate inference techniques such as eMIMIC. Existing work proposed flexible monitoring frameworks that provides ISPs with option of collecting different forms of aggregate traffic data [96, 97]. However, it is challenging to scale such systems to the entire network traffic because of the high processing overhead [98]. Furthermore, the traffic in the ISP network is growing, especially in cellular ISPs with the advent of 5G networks, making it more challenging for such systems to catch up with the scale of ISP networks in the near future. Until then, an adaptive monitoring approach using lightweight inference for detecting video performance issues can enable ISPs to monitor network-wide video performance.

Video service diversity and machine learning: Ideally, ISPs would like to monitor

video performance for all services streamed over their networks with the following goals: i) understand the ability of different parts of the network to support video, and ii) uncover service-specific design issues arising due to network and application-layer interactions in HAS. Monitoring all services, however, require obtaining ground truth QoE data for all of these services to train the ML models. One potential solution could be to develop generalizable ML models that can infer QoE for a service with high accuracy even if it was not used in training the model. Related work shows that this can be challenging even with packet traces, let alone with coarse-grained TLS transaction data. At the same time, we note that most of the contribution to the network demand is by a handful of services [69]. More specifically, the top 5 video services have been shown to contribute 95% of the video demand and have high network coverage (e.g., 99% of the cells in cellular network [69]). Thus, it may suffice for ISPs if they develop ML models that infer QoE for these top services to get a representative view of video performance across the entire network. ISPs can then also infer QoE for the remaining services using fine-grained network data from only a part of the network to uncover any service-specific design issues.

5.6 Conclusion and Future Work

In this chapter, we demonstrated that coarse-grained measurement data can be used for detecting performance issues related to video streaming with reasonable accuracy. This coarse-grained data is light-weight and does not require additional efforts by ISP in terms of network monitoring. The predictive nature comes from mainly two factors: i) data-related features being able to capture the network quality ii) difference in patterns of TLS transactions for low and high QoE sessions. Furthermore, the estimation using TLS transaction data has a comparable accuracy with respect to packet traces and has a significantly lower overhead. Our approach can enable an ISP to develop adaptive monitoring wherein it can collect more information for low QoE sessions such as packet traces for diagnosis.

CHAPTER 6

SUMMARY OF CONTRIBUTIONS AND FUTURE WORK

ISPs need an understanding of end-user video QoE for an efficient network management and provisioning. This thesis is a step in enabling ISPs to use passive network measurement data for QoE inference. We develop three inference approaches each for a different *streaming context* and network measurement data. The first two inference methods, MIMIC and eMIMIC, are designed with a primary focus on inference accuracy for two different streaming contexts. The third inference method addresses scalability challenges of inference by considering coarse-granular but light-weight network measurement data. The contributions of the thesis are summarized below:

The MIMIC methodology: This work presents an inference methodology for unencrypted video leveraging the semantics of HAS protocol and the corresponding network traffic pattern. We evaluate MIMIC at a large-scale using ground truth data from a popular streaming service. We conduct a measurement study by deploying MIMIC inside the network of a major U.S. mobile network operator. The measurement study characterizes a variety of video services and their QoE from the unique perspective of an MNO. We present insights on the relative video demand, video service design and its impact on network usage and QoE, and impact of network factors on video QoE.

The eMIMIC methodology: This work provides a methodology to infer QoE for encrypted video. eMIMIC reconstructs the chunk-based delivery sequence of a video session by using packet traces for QoE inference. We develop an experimental framework with automated streaming and collection of network traces and ground truth of video sessions and use it to evaluate eMIMIC with three popular commercial video streaming services. Our evaluation shows that eMIMIC can accurately estimate both per-session as well as fine-granular QoE metrics without requiring any training on ground truth QoE metrics.

Inference using coarse-grained network data: This work develops a QoE inference method that uses light-weight and readily available network measurement data in the form of TLS transactions. We use machine learning for inference in this work given the coarse-granular nature of the data. The features used in machine learning model are developed based on our prior experience in designing session modeling-based methods. We also develop a simple heuristic to distinguish consecutive video sessions from the same service. The heuristic leverages the server access patterns at the beginning of the session to identify transactions corresponding to a new session. We extensively evaluate the machine learning model for three popular streaming services. We show that coarse-granular data can be used to detect video performance issues with a reasonably high accuracy. We compare the estimation accuracy with that obtained using packet-level traces as well as the associated computation and memory overhead to demonstrate the accuracy and scalability trade-off.

6.1 Bringing it together

In this thesis, we provide approaches for QoE inference for different kinds of streaming context and network data. These approaches differ in terms of scalability, ease of deployment, and granularity of QoE inference. An operator may be able to combine one or more of these techniques for a more effective inference system. Here we provide two examples of such a combination.

Adaptive video performance monitoring: We envision an adaptive video performance monitoring wherein ISPs collect more fine-grained network data as and when they detect video performance issues in the network. Thus, ISPs could use TLS transaction data for a network-wide coarse-grained video QoE estimation. Once an issue has been detected in a particular network location, an ISP can collect packet traces only from this location and use eMIMIC for a fine-grained QoE estimation. Such selective fine-grained QoE inference can also be used whenever an ISP changes the network configuration to gauge the impact of the change on video performance.

Training ML-based approach: In order to use the ML-based technique proposed in Chapter 5, ISPs first need to build QoE inference models using training data. We use a browser-based automated framework that collects ground truth QoE metrics using the Javascript APIs provided by the browser platform and some service-specific APIs. It could be difficult to obtain ground truth QoE metrics using this methodology for some services or under some platforms like iOS or smart TVs. In such a case, ISPs can use QoE estimates from MIMIC for training the ML-based models as it is easier to collect HTTP transaction data by using a trusted proxy in a controlled setting.

6.2 Future Work

We now present some potential future directions for this work:

- **Scalability of inference approaches**: This thesis takes a step in addressing scalability challenges by considering light-weight network data for inference and exploring the scalability-accuracy trade-offs. A potential future direction can be to explore this trade-off in greater detail. There are alternative forms of network data that ISPs can typically collect, such as flow-level summaries using NetFlow [94] and network Key Performance Indicators (KPIs) reflecting the health and performance metrics of the network elements. Each of these data sources would occupy a different position in the scalability vs accuracy spectrum. Understanding these trade-offs can inform ISPs with best practices in network data collection.

- **Handling streaming protocols and design evolution**: QoE inference using network data is a moving target. This is because networks as well as the video service implementations evolve over time. For instance, there is an increasing interest in deploying new application or transport protocols in support of video streaming such as QUIC and HTTP/2. Both QUIC and HTTP/2 allow request multiplexing over the same connection which makes it challenging to use session modeling-based technique de-

veloped in Chapter 4. A possible solution is to consider estimating the number of segments by monitoring the data in upstream direction. One can then design techniques that consider the multiplexed requests together as a single chunk but with a duration determined based on the number of upstream requests observed on the network. Similarly, there are differences in the aggregate forms of data available from UDP-based QUIC protocol as it uses different mechanisms from TLS for encrypting network data. Therefore, ML-based approach designed in Chapter 5 need to be adapted for the aggregate forms of data available from QUIC.

There are also interesting research challenges in designing systems for handling streaming service evolution. One potential approach can be to design a continual validation framework. The framework could automatically detect changes in services over time using statistical techniques such as *concept drift* [99]. It also needs to have mechanisms that re-calibrate existing inference techniques with minimal effort once a change has been detected. One possible approach to minimize the training effort could be to use techniques from machine learning such as *transfer learnin*g [100].

- **QoE diagnosis**: Once ISPs obtain video QoE estimate, they would also be interested in diagnosing poor video QoE. There are several factors that can lead to degradation of video QoE, such as congestion in the ISP network or interdomain connection, or an issue at the end-user's device or the server. An accurate diagnosis of these issues can inform ISPs' own network management in terms of resource allocation (e.g. re-configuration of network scheduling to maximize QoE [55]) and long-term network provisioning (e.g. installing new capacity in persistently *congested* areas). In some cases, the diagnosis step may also help uncover issues caused by application and network-layer interactions that require collaboration between ISPs and content providers. Clearly, the video QoE estimation step by itself is not sufficient for identifying the root cause of QoE degradation. Thus, we need additional methods to localize and diagnose issues leading to poor QoE. A promising direction could be

to use techniques from network tomography. More specifically, ISPs could correlate QoE metrics obtained from inference methods and augment them with network path across users to localize issues.

REFERENCES

[1] *Impact of web QoE on revenues*, https://www.portent.com/blog/analytics/research-site-speed-hurting-everyones-revenue.htm, 2020.

[2] 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.

[3] C. S. INC, *Cisco VNI: Forecast and methodology, 2017-2022*, 2017.

[4] MUVI. (2019). New trends in internet video.

[5] *Open Connect: Netflix partnering with ISPs to deliver content*, https://openconnect.netflix.com/Open-Connect-Overview.pdf.

[6] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide QoE fairness using openflow-assisted adaptive video streaming," in *Proc. of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*.

[7] D. Soldani, M. Li, and R. Cuny, *QoS and QoE management in UMTS cellular systems*. John Wiley & Sons, 2007.

[8] J. Zhang and N. Ansari, "On assuring end-to-end QoE in next generation networks: Challenges and a possible solution," *IEEE Communications Magazine*,

[9] *Conviva: in-app QoE monitoring*, https://www.conviva.com/.

[10] New Relic, *Mobile APM features*, https://newrelic.com/mobile-monitoring/features, 2018.

[11] J. Jiang, X. Liu, V. Sekar, I. Stoica, and H. Zhang, "EONA: Experience-oriented network architecture," in *Proc. ACM HotNets*, 2014.

[12] M. Wichtlhuber, R. Reinecke, and D. Hausheer, "An SDN-based CDN/ISP collaboration architecture for managing high-volume flows," *IEEE TNSM*, 2015.

[13] K. Brunnström, S. A. Beker, K. De Moor, A. Dooms, S. Egger, M.-N. Garcia, T. Hossfeld, S. Jumisko-Pyykkö, C. Keimel, M.-C. Larabi, *et al.*, "Qualinet white paper on definitions of Quality of Experience," 2013.

[14] *ITU-T P.1203: Objective video QoE standard*, https://www.itu.int/rec/T-REC-P.1203, 2018.

[15] *VQEG: Objective video quality assessment*, https://www.its.bldrdoc.gov/vqeg/projects/audiovisual-hd.aspx, 2018.

[16] 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.

[17] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "A quest for an Internet Video Quality-of-Experience metric," in *Proc. ACM HotNets*, 2012.

[18] J. Jiang, V. Sekar, and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE," in *Proc. ACM CoNEXT*, 2012.

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

[20] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *Proc. of IEEE INFCOM*, 2016.

[21] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service," in *Proc. ACM SIGCOMM*, 2014.

[22] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. ACM SIGCOMM*, 2017.

[23] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, "Oboe: Auto-tuning video ABR algorithms to network conditions," in *Proc. ACM SIGCOMM*, 2018.

[24] A. Zambelli, "IIS smooth streaming technical overview," *Microsoft Corporation*, 2009.

[25] Apple, *HLS*, https://developer.apple.com/streaming.

[26] *DASH*, http://dashif.org/mpeg-dash/.

[27] *Live Stream growing as big as VOD*, https://www.muvi.com/blogs/live-stream-growing-big-vod.html.

[28] Macromedia, *RTMP*, https://en.wikipedia.org/wiki/Real-Time_Messaging_Protocol, 2019.

[29] A. Jain, A. Terzis, H. Flinck, N. Sprecher, S. Arunachalam, and K. Smith. (2019). RTSP.

[30] *HTTP Adaptive Streaming*, https://en.wikipedia.org/wiki/Adaptive_bitrate_streaming, 2019.

[31] Y.-T. Lin, T. Bonald, and S. E. Elayoubi, "Impact of chunk duration on adaptive streaming performance in mobile networks," in *Wireless Communications and Networking Conference (WCNC), 2016 IEEE*, 2016.

[32] S. Akhshabi, A. C. Begen, and C. Dovrolis., "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proc. ACM MMSys*, 2011.

[33] A. Mansy, M. Ammar, J. Chandrashekar, and A. Sheth, "Characterizing client behavior of commercial mobile video streaming services," in *Proc. ACM MoVID*, 2013.

[34] Y. Chen, K. Wu, and Q. Zhang, "From QoS to QoE: A tutorial on video quality assessment," *IEEE Communications Surveys & Tutorials*, 2014.

[35] A. Raake, M.-N. Garcia, W. Robitza, P. List, S. Göring, and B. Feiten, "A bitstream-based, scalable video-quality model for http adaptive streaming: Itu-t p. 1203.1," in *Proc. of IEEE Quality of Multimedia Experience (QoMEX)*, 2017.

[36] K. T. Chen, C. C. Tu, and W. C. Xiao, "OneClick: A Framework for Measuring Network Quality of Experience," in *Proc. IEEE INFOCOM*, 2009.

[37] 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.

[38] 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.

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

[40] 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.

[41] 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.

[42] D. Tsilimantos, T. Karagkioules, and S. Valentin, "Classifying flows and buffer state for Youtube's HTTP adaptive streaming service in mobile networks," in *Proc. of ACM MMSys*, 2018.

[43] M. H. Mazhar and Z. Shafiq, "Real-time Video Quality of Experience Monitoring for HTTPS and QUIC," in *IEEE INFOCOM*, 2018.

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

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

[46] 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.

[47] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "eMIMIC: Estimating HTTP-Based Video QoE Metrics from Encrypted Network Traffic," in *Proc. of IEEE/IFIP TMA*, 2018.

[48] F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster, "Inferring streaming video quality from encrypted traffic: Practical models and deployment experience," *Proc. of the ACM on Measurement and Analysis of Computing Systems*, 2019.

[49] F. Loh, F. Wamser, C. Moldovan, B. Zeidler, D. Tsilimantos, S. Valentin, and T. Hoßfeld, "Is the uplink enough? estimating video stalls from encrypted network traffic," in *IEEE/IFIP NOMS*, 2020.

[50] M. Seufert, P. Casas, N. Wehner, L. Gang, and K. Li, "Features that matter: Feature selection for on-line stalling prediction in encrypted video streaming," in *Proc. of IEEE INFOCOM Computer Communications Workshops*, 2019.

[51] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, 2008.

[52] W. M. Shbair, T. Cholez, J. Francois, and I. Chrisment, "A multi-level framework to identify HTTPS services," in *Proc. of IEEE/IFIP NOMS*, 2016.

[53] J. Garcia, T. Korhonen, R. Andersson, and F. Västlund, "Towards Video Flow Classification at a Million Encrypted Flows Per Second," in *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, 2018.

[54] *The Transport Layer Security (TLS) Protocol Version 1.2*, IETF RFC 3954, 2008.

[55] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang, "A Scheduling Framework for Adaptive Video Delivery over Cellular Networks," in *Proc. of ACM MobiCom*, 2013.

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

[57] 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. of IEEE MASCOTS*, 2013.

[58] S. Benno, J. O. Esteban, and I. Rimac, "Adaptive streaming: The network has to help," *Bell Labs Technical Journal*, 2011.

[59] 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. of Software, Telecommunications and Computer Networks (SoftCOM)*, IEEE, 2012.

[60] I. Ben Mustafa, T. Nadeem, and E. Halepovic, "FlexStream: Towards flexible adaptive video streaming on end devices using extreme SDN," in *Proc. of ACM MM*, 2018.

[61] S. Xu, S. Sen, Z. M. Mao, and Y. Jia, "Dissecting VOD services for cellular: Performance, root causes and best practices," in *Proc. of ACM IMC*, 2017.

[62] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, Z. L. Zhang, M. Varvello, and M. Steiner, "Measurement Study of Netflix, Hulu, and a Tale of Three CDNs," *Proc. of IEEE/ACM Transactions on Networking*, 2015.

[63] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z. L. Zhang, "Unreeling Netflix: Understanding and improving multi-CDN movie delivery," in *Proc. of IEEE INFOCOM*, 2012.

[64] V. K. Adhikari, S. Jain, Y. Chen, and Z. L. Zhang, "Vivisecting youtube: An active measurement study," in *Proc. of IEEE INFOCOM*, 2012.

[65] J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "Shedding Light on the Structure of Internet Video Quality Problems in the Wild," in *Proc. of ACM CoNEXT*, 2013.

[66] H. Yin, X. Liu, F. Qiu, N. Xia, C. Lin, H. Zhang, V. Sekar, and G. Min, "Inside the bird's nest: Measurements of large-scale live vod from the 2008 olympics," in *Proc. of ACM IMC*, 2009.

[67] M. Ghasemi, P. Kanuparthy, A. Mansy, T. Benson, and J. Rexford, "Performance characterization of a commercial video streaming service," in *Proc. of ACM IMC*, 2016.

[68] J. Erman, A. Gerber, K. K. Ramakrishnan, S. Sen, and O. Spatscheck, "Over the top video: The gorilla in cellular networks," in *Proc. of ACM IMC*, 2011.

[69] 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.

[70] P. Casas, B. Gardlo, R. Schatz, and M. Mellia, "An educated guess on qoe in operational networks through large-scale measurements," in *Proc. of ACM Internet-QoE*, 2016.

[71] D. D. Vleeschauwer, H. Viswanathan, A. Beck, S. Benno, G. Li, and R. Miller, "Optimization of HTTP adaptive streaming over mobile cellular networks," in *Proc. of IEEE INFOCOM*, 2013.

[72] 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.

[73] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?" In *Proc. of ACM NOSSDAV*, 2012.

[74] *Video QoE metrics*, `https://mux.com/blog/the-four-elements-of-video-performance/`, 2018.

[75] 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.

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

[77] 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.

[78] 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.

[79] *FCC dataset*, `https://www.fcc.gov/measuring-broadband-america`, 2017.

[80] *Scikit: Python library*, `scikit-learn.org/stable`, 2018.

[81] 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.

[82] V. Krishnamoorthi, N. Carlsson, and E. Halepovic, "Slow but steady: Cap-based client-network interaction for improved streaming experience," in *Proceedings of the IEEE/ACM International Symposium on Quality of Service (IEEE/ACM IWQoS)*, 2018.

[83] *ECOMP*, `https://about.att.com/content/dam/snrdocs/ecomp.pdf`, 2018.

[84] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," Tech. Rep. IETF draft-ietf-quic-transport-16.

[85] M. Belshe, R. Peon, and M. Thomson, "Hypertext transfer protocol version 2 (HTTP/2)," Tech. Rep., 2015.

[86] M. Seufert, P. Casas, N. Wehner, L. Gang, and K. Li, "Stream-based machine learning for real-time QoE analysis of encrypted video streaming traffic," in *Proc. of IEEE ICIN*, 2019.

[87] M. Yu, "Network telemetry: Towards a top-down approach," *Proc. of ACM SIGCOMM CCR*, 2019.

[88] X. Xu, J. Jiang, T. Flach, E. Katz-Bassett, D. Choffnes, and R. Govindan, "Investigating transparent web proxies in cellular networks," in *Proc. PAM*, 2015.

[89] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, "An untold story of middleboxes in cellular networks," in *Proc. of ACM SIGCOMM*, 2011.

[90] *Squid: Optimising web delivery*, `http://www.squid-cache.org/`, 2017.

[91] M. Trevisan, I. Drago, and M. Mellia, "PAIN: A Passive Web Speed Indicator for ISPs," in *Proc. of ACM Internet QoE*, 2017.

[92] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[93] 3GPP. (2017). Key Performance Indicators (KPI) for Evolved Universal Terrestrial Radio Access Network (E-UTRAN).

[94] B. Claise, *Cisco Systems NetFlow Services Export Version 9*, IETF RFC 5246, 2004.

[95] I. N. Bermudez, M. Mellia, M. M. Munafo, R. Keralapura, and A. Nucci, "DNS to the rescue: Discerning content and services in a tangled web," in *Proc. of ACM IMC*, 2012.

[96] C. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk, "Gigascope: A stream database for network applications," in *Proc. of ACM SIGMOD*, 2003.

[97] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," in *NSDI*, 2013.

[98] A. Gupta, R. Harrison, M. Canini, N. Feamster, J. Rexford, and W. Willinger, "Sonata: Query-driven streaming network telemetry," in *SIGCOMM*, 2018.

[99] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, 2014.

[100] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, 2009.