Received April 6, 2019, accepted May 10, 2019, date of publication May 23, 2019, date of current version June 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2918433

A Network-Layer QoE Model for YouTube Live in Wireless Networks

LUIS ROBERTO JIMÉNEZ, MARTA SOLERA, AND MATÍAS TORIL
Communication Engineering Department, University of Malaga, E-29071 Málaga, Spain

Corresponding author: Luis Roberto Jiménez (lrjp@ic.uma.es)

This work was supported by the Spanish Ministry of Economy and Competitiveness under Grant TEC2015-69982-R and Grant RTI2018-099148-B-I00.

ABSTRACT YouTube Live is one of the most popular services on the Internet, enabling easy streaming of a live video with the acceptable video quality. Thus, understanding user perception of this service is of the utmost importance for network operators. As in other video streaming services, YouTube Live traffic is affected by delays and interruptions due to unfavorable network conditions, which translate into unacceptable initial reproduction times, image freezes, or abrupt changes in image quality. Detecting these events is key to ensure an adequate quality of experience (QoE). Unfortunately, data encryption makes it very difficult for operators to monitor the QoE from packet-level data collected in network interfaces. In this paper, an analytical model to estimate the QoE for encrypted YouTube Live service from packet-level data collected in the interfaces of a wireless network is presented. The inputs to the model are TCP/IP metrics, from which four service key performance indicators (S-KPIs) are estimated: initial video play start time, video interruption duration, video interruption frequency, and image quality. The model is developed with an experimental platform consisting of a live streaming server, a terminal agent, a radio access network (e.g., Wi-Fi access point), a network-level emulator, a probe software, and a man-in-the-middle proxy. Model assessment is carried out by comparing the S-KPI estimates with measurements from the terminal agent under different network conditions introduced by the network emulator.

INDEX TERMS HTTP adaptive streaming (HAS), encryption, modeling, network emulator, quality of experience (QoE), service key performance indicator (S-KPI).

I. INTRODUCTION

In the last decade, the exponential growth of traffic and the launch of new services have completely changed mobile communications networks and this trend will continue in the coming years. By 2023, it is estimated that there will be 7.3 million smartphone subscriptions connected to different networks [1]. Likewise, the deployment of future 5G networks will pave the way for new mobile use cases [2].

In parallel, technological advances have raised mobile user expectations, forcing operators to change the way they manage their networks. In a market where most operators provide a similar offer, service performance as perceived by the customer (a.k.a. Quality of Experience, QoE) has become the key differentiating factor. As a consequence, operators are changing legacy network management approaches, focused on network performance and Quality of Service (QoS) to a more modern approach focused on user opinion and QoE [3]. Customer experience management (CEM) will be even more important in 5th generation (5G) mobile networks, where services with very different requirements will coexist (e.g., social networks, virtual reality, autonomous vehicle...) [2]. Thus, QoE should be the main criterion for assigning radio resources [4], prioritizing service requests [5] or selecting paths in software-defined architectures [6] in future 5G systems.

At present, the most important service in mobile networks is video streaming, which is expected to generate 82% of all IP traffic volume by 2021 [7]. For this reason, operators focus their attention on understanding traffic from this kind of applications. In the past, video distribution was based on HTTP progressive download (HPD), where the server sends parts of the clip with the format specified by the client in the HTTP Request message [8]. Currently, the video distribution technique adopted by most service providers (e.g., YouTube, Netflix, Hulu...) is HTTP adaptive streaming (HAS) [9].

The associate editor coordinating the review of this manuscript and approving it for publication was Dongxiao Yu.

HAS breaks the video content into a sequence of small HTTP-based file segments (normally, from 2 to 10 seconds of video content), which is available to the user with different resolutions (i.e., at different coding bitrates). While reproduction takes place, the video client automatically selects the segment with the highest bitrate still avoiding client buffer underruns causing rebuffering events (a.k.a. video stalling). Thus, HAS adapts to changing network conditions while providing the highest possible image quality. Additionally, several social networks (e.g., YouTube Live, Facebook Live, PeriscopeĚ) now offer the possibility to provide live video streaming with small reproduction delays. All these changes make it necessary to understand the properties of the new video traffic to be able to monitor and control the QoE perceived by the users.

A key task in video QoE management is QoE assessment. The simplest approach is by performing subjective tests, where real observers judge subjective video quality provided under different network conditions in a controlled lab environment [10]. In parallel, service performance metrics (e.g., initial reproduction delay, stalling frequency/duration...) can also be measured with specialized client software, so that the impact of each metric on user experience can be isolated [11]. However, subjective tests are complex and time-consuming, and not valid for real-time large-scale monitoring. Objective methods are classified into media-based, bitstream-based or parametric models [12]. Media-based (a.k.a. signal-based) methods decoding the video content and bitstream-based methods that rely on the video elementary stream, both require access to the application layer and are therefore only suitable for service providers that have access to one side of the link. Alternatively, parametric packet-layer methods analyze protocol messages to identify the different stages of the session (e.g., start of video playback, stalling ...), from which service key performance indicators (S-KPIs) can be obtained for the connection (e.g., initial reproduction delay, stalling duration...). Then, S-KPIs are translated into mean opinion score (MOS) [13] values by formulas derived in subjective tests. In the past, S-KPIs could be directly measured by network operators using packet inspectors (a.k.a. software probes) in key network interfaces of the core network [14]. Note that congestion and flow control mechanisms in TCP ensure that network-level indicators reflect end-to-end service quality. Unfortunately, this approach is not possible anymore due to traffic encryption. Since 2016, 97% of YouTube traffic is encrypted using HTTPS connections with Transport Layer Security (TLS) and Secure Sockets Layer (SSL). In the absence of other methods, simpler parametric models can blindly relate basic network-level QoS measurements (e.g., average IP session throughput) to MOS figures. This approach is often followed by frameworks used for large-scale on-line passive monitoring on a per-connection basis [15], [16]. Recently, these platforms have been extended with data analytics capabilities to isolate the indicators that better reflect user experience and predict their trends [17].

Several QoE assessment models for video streaming have been proposed in the literature. Preliminary works analyze video streaming traffic, identifying differences between service providers [18], wired and wireless networks [19] and terminal operating systems [20]. In [21], a linear formula is derived by regression analysis to estimate the QoE of a video streaming session from S-KPIs, such as initial reproduction delay, stalling frequency, and stalling duration. In [22], a QoE model for the conventional (i.e., non-real time) YouTube service based on estimating the client buffer level is presented. Even if the analysis is focused on fixed-quality video streaming (i.e., without HAS), the authors justify the need for new service performance metrics to estimate the QoE with HAS. More focused on HAS, a QoE model for Apple HTTP Live video streaming solution with HAS is proposed in [23] to estimate the QoE in mobile devices from the linear combination of average image quality, standard deviation of image quality and frequency of switches between image qualities. As an alternative to network-based approaches, in [15], several QoE models for popular smartphone apps based on passive in-device measurements are proposed. Most of these works focus on conventional (i.e., video-on-demand) video streaming, whose QoE models may not be valid for live video streaming due to the different parametrization of system components. To the authors knowledge, no previous work has described a QoE model for YouTube Live service, including online transmission, HAS, and encryption, based on network-layer passive monitoring.

In this work, a data-driven analytical QoE model for YouTube Live video streaming service in wireless networks is proposed. The proposed parametric model relates basic QoS metrics measured in core network interfaces on a per-connection basis to S-KPIs reflecting end-to-end service performance, which can then be translated into MOS figures with classical packet-layer methods. The inputs of the model are common TCP/IP metrics (e.g., average session throughput, packet loss ratio and round-trip time), from which four of the most important video S-KPIs are estimated: initial video play start time, video interruption duration, video interruption frequency, and image quality. The model is developed by regression techniques on data collected with an experimental platform, consisting of a YouTube Live streaming server, a user terminal agent, a Wi-Fi wireless network, a network-level emulator, a probe software and a man-in-the-middle proxy. This platform allows to: a) automate the creation of live video streaming sessions in YouTube, b) emulate user interactions with the live video streaming application through a smartphone, c) modify network conditions with a network emulator, and d) intercept SSL and HTTPS traffic with a man-in-the-middle proxy [24] to decrypt protocol messages. With this platform, a preliminary analysis is carried out to identify differences between conventional and live video streaming that justifies the development of new QoE models. Then, model assessment is carried out by comparing S-KPI estimates with real measurements made by the user terminal agent and video resolution data in protocol messages under different network conditions. The rest of the paper is structured as follows. Section II reviews parametric QoE models for video streaming in the literature. Section III describes the experimental platform. Section IV discusses the main differences between conventional and live YouTube streaming service. Section V explains the performance model relating TCP/IP metrics with S-KPIs for YouTube Live service. Section VI presents the main contribution, consisting of the regression curves used to estimate the S-KPIs of the service from TCP/IP metrics. Finally, Section VII presents the conclusions of the work.

II. RELATED WORK

The first QoE models were designed for conventional video-on-demand streaming based on HPD, where stalling is the most critical factor. The simplest approach is to estimate user experience directly from network-layer metrics, such as packet loss/jitter/delay and bandwidth [25]. More refined approaches estimate user experience from application-layer metrics. In [21], a linear formula is derived by regression analysis to estimate the QoE of an HPD video streaming session from S-KPIs, such as initial reproduction delay, stalling frequency, and stalling duration. In [26], an exponential function is proposed to estimate MOS from the number of stalling events and their duration. In [22], a QoE model for the conventional YouTube service based on estimating the client buffer level is presented. Simpler metrics, such as the reception rate (i.e., download throughput vs video coding bitrate) [26] or the stalling duration ratio [27] can also give a hint of the video user experience. As an alternative to network-based approaches, in [15], a QoE model for YouTube based on passive in-device measurements is proposed.

Hoßfeld *et al.* [28] evaluated for the first time the factors affecting the QoE in HAS, suggesting a simplified model that estimates video MOS from the fraction of time that the highest quality level is viewed. A more comprehensive analysis of perceptual and technical factors affecting the QoE with HAS is presented in [29]. More refined models estimate the QoE from the linear combination of average image quality, standard deviation of image quality and frequency of switches between quality levels [23], combined with stalling statistics [30], [31]. With recent advances in big data analytics, the newest approaches apply machine learning to build sophisticated models that capture complex dependencies of MOS with a large number of predictors [32]–[35].

In [36], a comparison of several of the above-mentioned models is done based on a large video QoE database. As expected, results show that the simplest network-based and stalling-centric models do not perform well with HAS, and the most complex hybrid methods combining media-based and packet-based information (e.g., ITU-T P.1203 [34]) outperform the others.

In [37], a simple data-driven QoE model for conventional video streaming services in wireless networks is presented. The proposed model is built with fixed resolution videos, and cannot be used to estimate the impact of HAS on user perception.

The above works are focused on video-on-demand (i.e., non-real-time) streaming service. The main contributions of this work are: a) a preliminary analysis showing the differences between conventional and live video streaming in wired and cellular networks, and b) a set of functions mapping TCP/IP metrics to S-KPIs for YouTube Live service, which allow mobile network operators to reuse existing packet-layer QoE models (e.g., [21], [34]).

III. EXPERIMENTAL TEST PLATFORM

Fig. 1 shows a diagram of the platform used to automate measurement collection. The platform is made up of two modules: a broadcast module (left part of the diagram) responsible for the live broadcast of a video by the YouTube Live platform, and a measurement module (right part of the diagram) responsible for the modification, decryption, collection, and analysis of measurements. The broadcast module consists of a PC running Wirecast application. The measurement module consists of a man-in-the-middle proxy (mitmproxy) recording all HTTP and HTTPS traffic and a mobile terminal connected to a Wi-Fi network using a PC as a gateway with direct output to the Internet. In the mobile terminal, a user terminal agent (TEMS Pocket) is running, which mimics user interactions during the video streaming session. TEMS application can also collect S-KPI measurements, as it has access to one of the communication endpoints. On the PC, a network emulator (NetEm) is executed to modify network conditions in a controlled way (e.g., available bandwidth, delay and/or packet loss ratio). Packet-level network measurements are collected at the network emulator interface to the Internet by a standard capturing tool. This data is then processed with a traffic monitoring and analysis application (Network Probe), with which basic QoS metrics for each connection are obtained. The resulting data is used to derive the QoE model relating QoS metrics to S-KPIs. All these processes are detailed in the following paragraphs.

A. LIVE VIDEO-STREAM BROADCASTING

In this work, a live video broadcast session is first created in the YouTube platform. For this purpose, a computer generates multimedia content to be distributed in real time. Specifically, a live streaming server consisting of a PC with two 2.4-GHz 8-core Intel (R) Xeon processors, 64GB of RAM, Microsoft Windows Server 2016 Datacenter ver. 10.0.14933 operating system and a Matrox G200 multimedia card are used. The server is connected to an isolated network point to avoid unwanted changes in the available bandwidth for uploading multimedia stream to YouTube (100 Mbps upload speed). Telestream Wirecast software and the web camera are installed on the server. Wirecast is a software for live video streaming that allows to produce and stream multimedia events in real time to YouTube from a workstation [38]. The version used is Wirecast Pro 7.3 64-bits, with high-definition video broadcast capability. Wirecast uses an external device

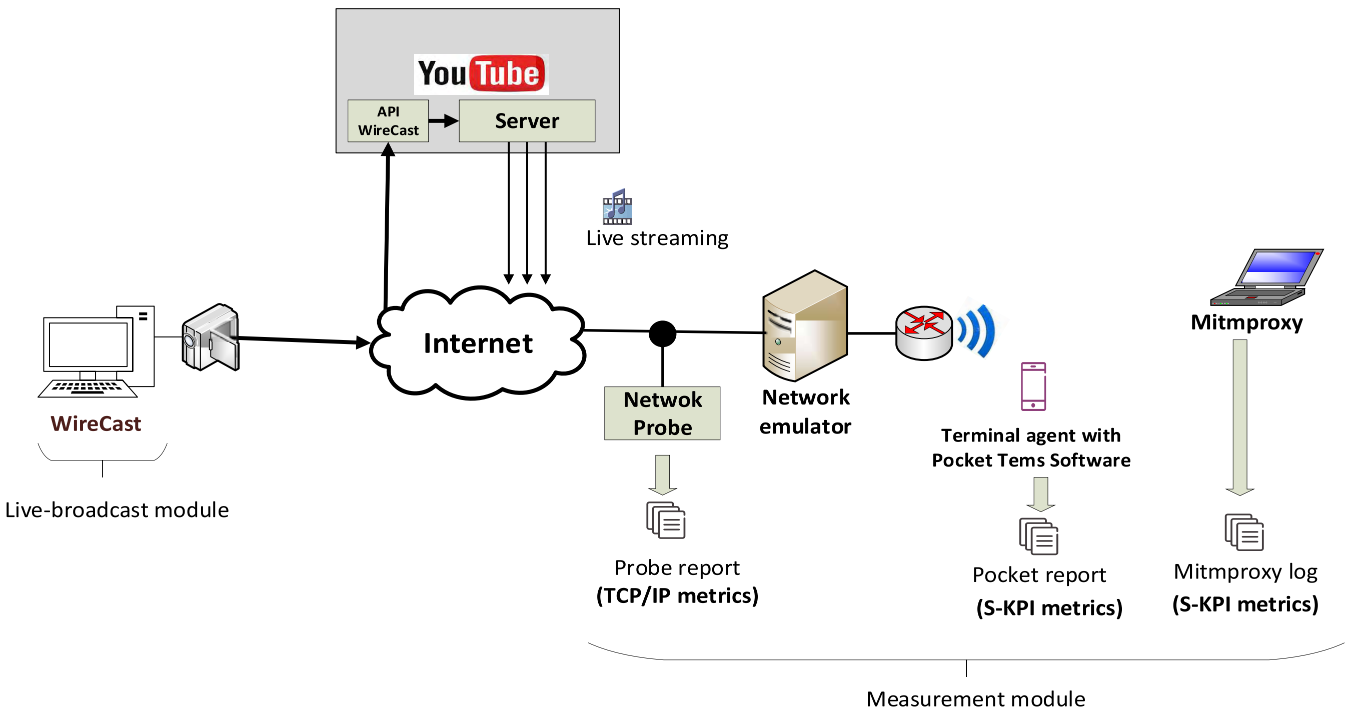


FIGURE 1. Experimental test platform.

to capture live video, a Logitech HD Pro C920 webcam, which allows high-definition recordings (HD 1080p). For this device, the video format, the video coder and the output bitrate must be configured. In this work, 1920x1080 resolution, progressive format, H.264 AVC encoder, and 3-9 Mbps bitrate (1080p profile) are selected. After setting up live video-stream upload, a stream event gateway on the YouTube website must be created by logging into the YouTube Creator Studio service. This service sets up the channel where transmission takes place, provides the video identifier (ID) and may change video stream attributes in real time (e.g., audio and video encoder) [39]. When creating the event, various parameters must be established: name, broadcast start time, download video stream characteristics and event type characteristics (Public, Private, Hidden). The Public event distributes the live streaming ID in the YouTube Live list at a global level, allowing any user to access the live stream; in contrast, in Private and Hidden modes, it is necessary to have the ID of the event to access the live stream. In the testbed, a hidden event is used. This event is linked to live capture. Subsequently, YouTube transcodes the content received by the upload link at a specific bitrate, creating a master stream with the bitrate set when selecting the encoder in the event configuration. Then, YouTube replicates the flow in different bitrates, so that the broadcast live video stream is available to all types of users [39].

B. LIVE VIDEO-STREAM RECEPTION

Live video streaming traffic is generated by sending display requests to a previously created public broadcast event. For this purpose, a Samsung Galaxy Note 4 smartphone with TEMS Pocket application v16.3 [40] is used. This terminal agent software is responsible for emulating user interaction with the live video streaming client and collecting S-KPI measurements. The terminal connects to the Internet via a standard Wi-Fi router model HG556a, with Wireless 802.11 b/g/n and Ethernet 802.3u interfaces, operating at 2.4 GHz and configured in bridge mode to interconnect Wi-Fi devices to the measurement subnet. The connection between the wireless access point and the subnet is through a twisted pair cable connected from the access point to the PC with the network emulator.

C. NETWORK EMULATION

To change network conditions, NetEm network emulator [41], included in Linux kernel since version 2.6, is used. NetEm can introduce controlled effects on the subnet, such as packet delay, loss, duplication, and reordering. Packet delay and jitter are described by mean value, standard deviation and correlation coefficient. By default, a uniform distribution is used for the delay, which can be replaced by other functions, such as Pareto, Pareto-normal, normal or custom distributions created from experimental or simulation data [42]. On the platform, NetEm is installed on a PC with an i5-750 processor at 3 GHz, 8 GB of RAM and Ubuntu 16.04 LTS 64-bit operating system. This PC includes two network cards linked by a routing table to provide Internet access to devices connected to the wireless subnet. In the tests, only delay, packet loss, and maximum throughput (throttling) parameters are adjusted.

D. TRAFFIC DECRYPTION

User traffic can only be decrypted if the SSL certificate and private key of the user are known. Generally, such information is only available at the ends of the communication (client and server). In the experiments, a mitmproxy is located in the same Wi-Fi network as the mobile terminal generating video traffic. The mitmproxy is configured in transparent mode by redirecting traffic into the proxy at the network layer without changing client settings. Mitmproxy can decrypt traffic on the fly, as long as the client trusts its built-in certificate authority. For this purpose, a self-signed SSL certificate generated by the mitmproxy must be pre-installed at the mobile terminal. In the platform, mitmproxy is in a laptop with two Intel Core i5 (6th Gen) 6200U @ 2.3GHz processors, 8 GB of RAM and Ubuntu 16.04 LTS 64-bit operating system.

E. MEASUREMENT COLLECTION

As shown in Fig. 1, two measurement points are configured in the platform, at the Internet output and the terminal. The former is devoted to network-layer measurements (TCP/IP metrics), whereas the latter is devoted to S-KPI measurements.

1) NETWORK-LEVEL MEASUREMENTS

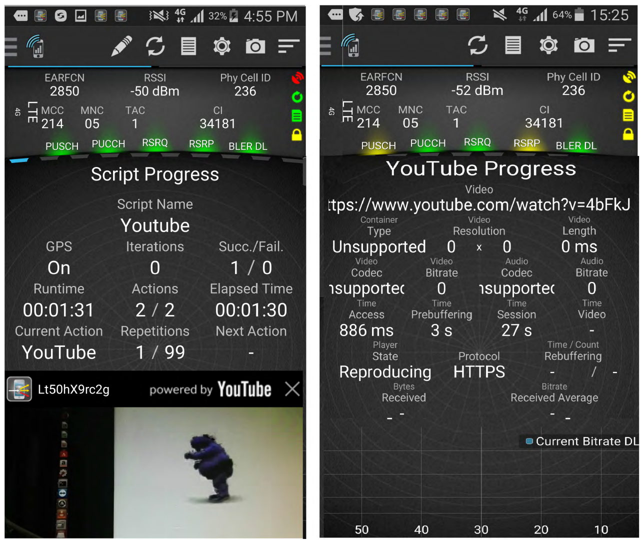
To obtain TCP/IP metrics, packet-level trace files are generated with the open source tool tcpdump [43]. The resulting ".pcap" files are then processed offline with proprietary traffic analysis and monitoring tool, Network Probe, whose output is a comma-separated values (CSV) file with basic QoS metrics per user, connection and packet burst.

2) S-KPI MEASUREMENTS

S-KPIs are obtained by two processes. On the one hand, TEMS terminal agent generates logs, which are then processed offline by a Python script to compute S-KPIs related to the client buffer status. On the other hand, mitmproxy performs traffic decryption in real time and generates logs with HTTP messages exchanged between the user terminal agent and YouTube server, from which image quality indicators can be derived later.

a: S-KPIs MEASURED BY THE TERMINAL AGENT

TEMS Pocket is a user terminal agent for verification, maintenance, and troubleshooting of mobile networks [40]. It allows the creation of scripts to automate the tests of services such as Facebook, Instagram, Twitter, WhatsApp, YouTube, etc. For YouTube Live service, YouTube option must be selected. Fig. 2 shows two screenshots when a measurement is taking place. On the lower left, a small screen displays the evaluated live video, while, on the right, the screen shows different S-KPIs values. In parallel, a report is generated with relevant S-KPI statistics. In this work, the S-KPIs analyzed by the terminal agent are initial video play start time, video interruption duration and video interruption frequency.



4:55 PM

EARFCN 2850 RSSI -50 dBm Phy Cell ID 236
LTE 4G MCC 214 MNC 05 TAC 1 CI 34181
PUSCH PUCCH RSRQ RSRP BLER DL

Script Progress

Script Name: Youtube
GPS: On
Iterations: 0
Succ./Fail.: 1 / 0
Runtime: 00:01:31
Actions: 2 / 2
Elapsed Time: 00:01:30
Current Action: YouTube
Repetitions: 1 / 99
Next Action: -

Lt50hX9rc2g powered by YouTube

15:25

EARFCN 2850 RSSI -52 dBm Phy Cell ID 236
LTE 4G MCC 214 MNC 05 TAC 1 CI 34181
PUSCH PUCCH RSRQ RSRP BLER DL

YouTube Progress

Video: ttps://www.youtube.com/watch?v=4bFkJ
Container Type: Unsupported
Video Resolution: 0 x 0
Video Length: 0 ms
Video Codec: ısupportec
Video Bitrate: 0
Audio Codec: ısupportec
Audio Bitrate: 0
Time Access: 886 ms
Time Prebuffering: 3 s
Time Session: 27 s
Time Video: -
Player State: Reproducing
Protocol: HTTPS
Time / Count Rebuffering: - / -
Bytes Received: - -
Bitrate Received Average: - -

FIGURE 2. TEMS Pocket screens while running YouTube script.

TABLE 1. YouTube itag mapping for live streaming [45].

Itag value	Video resolution [pixels]	Video encoding format
91	176 x 144	H.264 / MPEG-4 AVC Main Profile
92	320 x 240	H.264 / MPEG-4 AVC Main Profile
93	640 x 360	H.264 / MPEG-4 AVC Main Profile
94	854 x 480	H.264 / MPEG-4 AVC Main Profile
95	1280 x 720	H.264 / MPEG-4 AVC Main Profile
96	1920 x 1080	H.264 / MPEG-4 AVC High Profile

b: S-KPIs MEASURED BY THE MITMPROXY

YouTube server splits the live video sequence into segments (a.k.a. chunks), which are encoded with different algorithms, resolutions and containers. The combination of encoding algorithm, video resolution, and container type defines the video format, labeled with the parameter *itag* [19]. Table 1 shows some itag values for different live streaming videos resolutions and formats. Then, the video client changes the requested itag value based on its buffer status and central processing unit utilization. As explained in [44], YouTube encodes videos with Adobe Dynamic Streaming for Flash, which supports dynamic streaming over HTTP, but does not purely adopt the international standard MPEG-DASH. Instead, it generates a Uniform Resource Locator (URL) with the itag inside, which is exchanged between YouTube HTTP server and client in the video information in the HTML content.

When a user starts playing a live video stream, an HTTP video-playback request is sent to YouTube server. This message contains different parameters including video URL, codec, video live quality, resolution, etc. YouTube Server replies to the client with information on how the segmentation is done, which encoding and resolution are available for a particular video entry and other meta-data information. Thus, itag information can be obtained by decrypting HTTP messages and examining the recorded HTTP Archive (HAR) trace.

TABLE 2. YouTube video playback parameters: conventional vs live service.

Parameter	PC-conventional	mobile-conventional	PC-live	mobile-live
ms	au,rdu	au,rdu	lv	lv
source	YouTube	YouTube	yt_live_broadcast	yt_live_broadcast
mm	31, 29	31,29	32	32
initcwndbps	916250	763750	5750	7010
dur	60.727	60.727	5	5
clen	69298866	8237401	noclen	noclen
aitags	133,134,135,136,137,160,242, 243,244,247,248,271,278,313	N/A*	N/A	N/A
lmt	1488970281410920	1488967711147620	N/A	N/A
fvip	2	1	N/A	N/A
gcr	N/A	N/A	es	es
live	N/A	N/A	1	1
cmbypass	N/A	N/A	yes	yes
mpd_version	N/A	N/A	5	5
playlist_type	N/A	N/A	DVR	DVR

(*) N/A: not available.

3) AUTOMATION

The automation process consists of configuring user terminal agent and network emulator and capturing/decrypting packet-level traffic. Both processes must be synchronized to allow processing and interpretation of measurements.

a: USER TERMINAL AGENT

A script must be set to repeatedly download and reproduce the multimedia flow of the live video streaming session. Such a script consists of actions and settings. In this work, two actions are selected: YouTube script, in charge of initiating/terminating video session, and Log file recordings, which allows saving S-KPI measurements in a single file for later post-processing. The main settings are: a) Video, indicating the identifier (ID) of the streaming video channel to be reproduced; b) Streaming duration (SD), indicating the length of a measurement period (i.e., time to collect S-KPI statistics); c) pre-guard (PG) and post-guard (PTD) intervals, indicating guard periods inserted before and after measurements to ensure that the establishment and release of the video session is recorded in the log file (for YouTube, the recommended value for both is 10 seconds) [46]; d) repeat action (RA), representing the total number of times all script options are executed, and e) maximum iterations (MI), denoting the total number of times the script is executed. The total measurement period, E, is

$$E = MI \cdot RA \cdot (SD + PG + PTD). \qquad (1)$$

b: NETWORK EMULATOR

A script is used to configure delay, jitter, loss rate and maximum throughput parameter values in NetEm, based on the command line tool ‘tc’ [41]. The interface controlled by NetEm corresponds to the network emulator interface to the wireless access point.

IV. YOUTUBE FEATURES

A preliminary analysis is carried out to identify differences between conventional and live video streaming on YouTube based on session traces. The analysis covers both protocol messages and application behavior.

To that end, different combinations of client type (mobile phone or PC) and video service type (conventional or live) are tested. For each combination, a short video streaming session is established, where all HTTP messages (request/response) are captured between the YouTube client and server in the video playback sequence. As a result, 4 HAR traces of video streaming session of 1 minute are collected (PC-conventional, PC-live, mobile-conventional and mobile-live).

A. PROTOCOL MESSAGES

YouTube uses video playback request messages to grab media data from the server [47]. By inspecting the video URL in these messages, 39 parameters are found: *alr, c, gir, ms, mv, pcm2cms, pl, ratebypass, requiressl, source, cpn, cver, ei, expire, id, initcwndbps, ip, keepalive, key, mm, mn, mt, signature, clen, dur, itags, lmt, mime, range, rbuf, rn, aitags, cmbypass, fvip, gcr, ipbits, live, mpd_version and playlist type*. In this work, the analysis is focused on parameters differing between YouTube Live and conventional YouTube. The reader is referred to [47] and [48] for a description of the other parameters.

Table 2 shows the list of parameters of interest. For clarity, parameters are arranged in 3 groups, depending on whether they are available in both YouTube modes, only in conventional YouTube or only in YouTube Live. In the first group, it is observed that ***ms*** and ***source*** are static data showing if media stream comes from a conventional or live feed. ***mm*** includes as many integer values as media stream types. ***initcwndbps*** is the initial congestion window configured on the server side. It is observed that a few isolated live feeds, as the one in the table, show extremely low values of this parameter. Such low values are not observed in conventional YouTube sessions. ***dur*** denotes sequence length (in seconds) in conventional service, whereas, for live feeds, it denotes the length of downloaded video segments, between 1 and 5 seconds [49]. ***clen*** dynamically varies with *itag* in conventional service to reflect the total length of the video stream

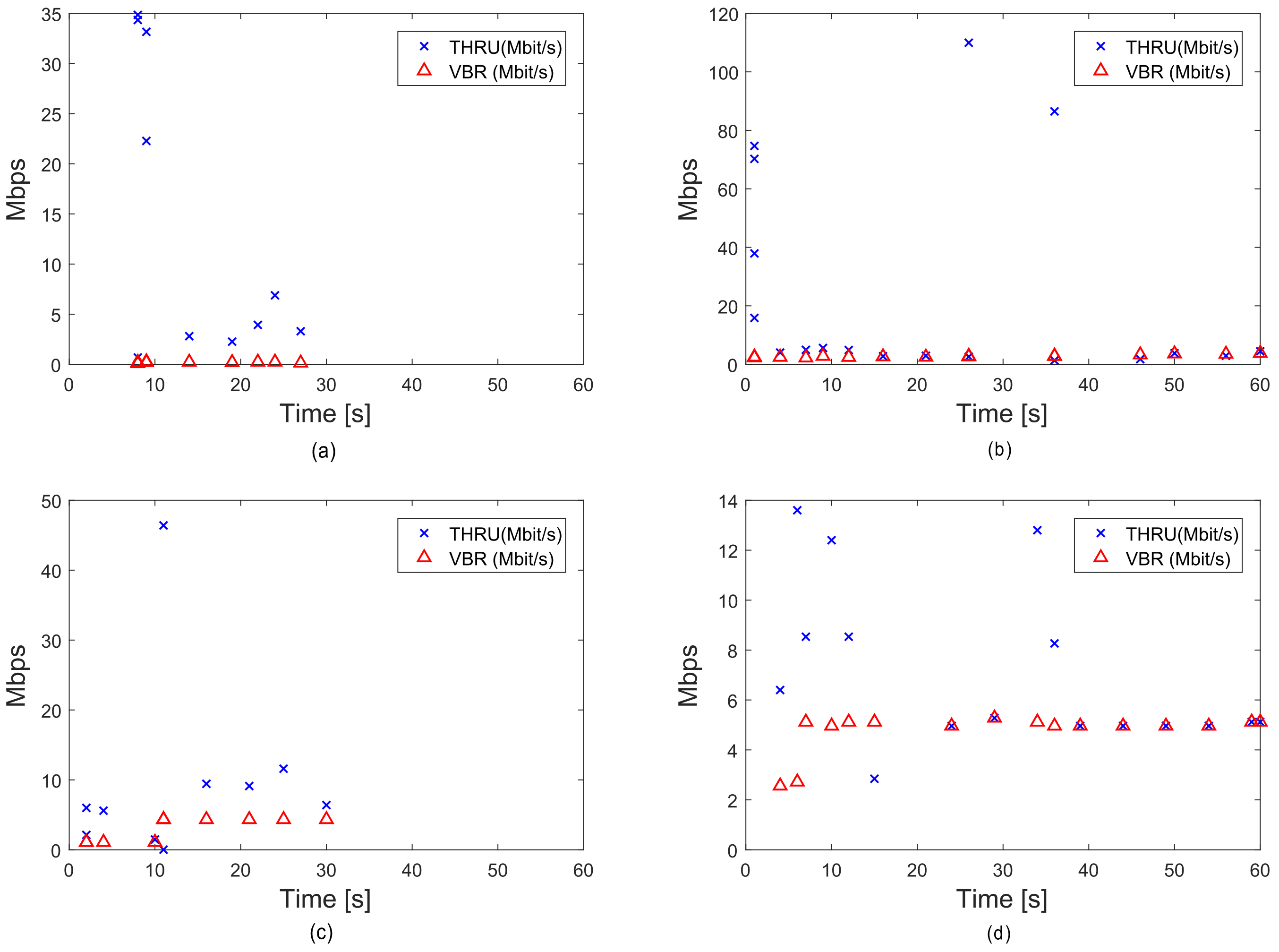


FIGURE 3. Video download rate and video bitrate versus time. (a) PC-conventional, (b) PC-live, (c) mobile-conventional, (d) mobile-live.

(content length) in bytes for the current video resolution. In contrast, in live feeds, ***clen*** reflects "noclen". This is consistent with the fact that the total duration of the live video session is unknown a priori. The second group of parameters only appear in conventional YouTube. ***aitag*** is the set of itags for the video sequence available on the server. ***lmt*** is a timestamp with the file creation date of the video content at the resolution currently selected by the client. ***fvip*** takes integer values and its name suggests its use for virtual IP forwarding to balance server load. The third group of parameters only appear in YouTube Live. ***grc*** is a two-letter code with the country, ***live*** is 1 for live videos tested, ***cmbypass*** reflects the activation of some bypass scheme and ***mpd_version*** takes integer values showing the Media Presentation Description (MPD) version. Finally, ***playlist_type*** denotes the type of playlist from a set of values (e.g., DVR, EVENT, VOD, LIVE ...), restricting how the playlist can be updated [50].

Figures 3 (a)-(d) show the average downlink throughput (THRU) and video bitrate (VBR) of the different video segments downloaded by the client, for each combination of client and service type. Specifically, (a) and (b) reflect the results of conventional and live service in PC, and (c) and (d) do the same for the mobile phone. In the figures, each point corresponds to a video playback request message (i.e., HTTP GET/request), represented in time when the request is sent from the client to the server. Thus, each message results in 2 points in the figure (1 cross for THRU and 1 triangle for VBR) aligned in time. Note that bursts consisting of several consecutive messages may see time aligned in the figure, even if they take place at different instants. Time origin is when the play button is pressed by the user and time span is limited to 1 minute for space reasons. Note that video sequence duration in this experiment is 1 minute in both the conventional YouTube and the YouTube Live session.

In Figs. 3 (a) and (c), corresponding to conventional YouTube in PC/mobile, it is observed that the download of the 60-s video file is completed in less than 30 s (less than half the video duration). In both cases, a burst of consecutive messages is generated at the beginning of the session to fill the client buffer as soon as possible, causing that

THRU>>VBR. As a result, video file download terminates well before reproduction ends. In the example, the 60-s video is downloaded with only 10 request messages. In contrast, in Figs. 3 (b) and (d), corresponding to their live counterparts, it is observed for a 60-s video playback live session, media is gradually downloaded (THRU $\approx$ VBR), as live content is available in the video server. In this case, for 60 s of live video, 16 and 18 packets are needed in the PC and mobile cases, respectively. A closer inspection of Fig. 3.(b) shows 2 samples of very large THRU in the middle of the session (seconds 26 and 36). These are due to the way THRU is calculated here on a per-message basis, which gives inaccurate estimates when the client sends 2 request messages at the same time. Likewise, a closer analysis of VBR values in Figs. 3 (c) and (d) shows a sudden increase of video bit rate (i.e., itag change) in the middle of the session, triggered by the client buffer state [47].

From these results, it is concluded that the client buffer state is completely different in conventional and live YouTube service, which has a strong impact on initial reproduction times and rebuffering statistics. This justifies the need for new ways to estimate service performance indicators from TCP/IP metrics for YouTube Live.

V. EXPERIMENTAL METHODOLOGY

This section describes the process to capture data to estimate the S-KPIs of an encrypted YouTube Live video streaming session from TCP/IP metrics. Firstly, the exact formula for each S-KPI is presented. Secondly, model construction is explained. Thirdly, model assessment is detailed.

A. S-KPI SELECTION

End-user QoE in HAS video streaming is mainly given by three basic criteria: initial buffering time, number and duration of stallings and delivered image quality [29]. For the latter, a recent study [36] shows that a combination of the average image quality and the average magnitude of image quality switches performs better than the time spent on the highest quality level. Unfortunately, in most cases, only a histogram of quality levels for the whole video streaming session is available, making it impossible to compute the magnitude of changes. Consequently, only four S-KPIs are defined here: video play start time, video interruption count, video interruption duration and image quality distribution. These S-KPIs are included in most packet-layer models (e.g., [21], [23], [30], [31]). Their definition is:

- Video play start time (SPT): time since the user sends the request to start the video streaming (click on the ID) until the first video frame appears on the screen.
- Video interruption frequency (IF): frequency with which video playback in a streaming session is interrupted for rebuffering reasons, computed as

$$IF = IC/SET \ \ [1/\text{min}], \tag{2}$$

where IC is the total number of stallings during the session and SET is the duration of the live reproduced video in minutes.
- Video interruption duration ratio (IDR): ratio reflecting the relative duration of stallings, computed as

$$IDR = ID/(ID + SET). \tag{3}$$

where ID is the total stalling duration.
- Image quality distribution (IQ): histogram reflecting the ratio of chunks downloaded with a certain itag value.

B. MODEL CONSTRUCTION

A measurement campaign was conducted from September 13th to September 22th, 2018. The test battery consists of a live video streaming broadcast with 1920x1080 resolution, progressive format, 25 frames/s, and 4.5-9 Mbps maximum bitrate. The local server is connected to the Internet by a 100/300 Mbps upload/download speed link, which guarantees enough bandwidth to transmit live streaming video without interruption. Video content is generated with an HDTV camera in front of a monitor displaying an endless sequence of a single scene with constant and moderate motion so that the original video bitrate does not vary much and changes in S-KPIs are only due to network conditions.

Once the live video streaming session is launched, the network emulator and the client are synchronized. The network state is modified by changing NetEm configuration. To reduce the number of experiments, the jitter parameter is set to 0. The tested combinations, selected according to values experienced by users in a live LTE downlink [37], are:

- Packet loss ratio [%]: 0, 0.75, 1.5, 3.
- Packet delay [ms]: 0, 50, 100, 200, 400.
- Maximum throughput limit [kbps]: 500, 1000, 2000, 4000.

Hence, 80 different network emulator settings (network states) are emulated, each of which is maintained for 33 minutes. Thus, the total time of the test battery is 44 hours.

For each network state, several live video streaming sessions are initiated. Maximum session length is set to 200 seconds (3.33 minutes) so that several session attempts are carried out per network state (at least, 10). For each session, the following data is captured:

- TCP/IP report, with TCP/IP metrics computed offline from pcap files by the Network Probe,
- S-KPI report, with video play start time, video interruption count and video interruption duration, measured by TEMS terminal agent, and
- Itag report, containing the histogram of itag values in each live video streaming segment, obtained by processing mitmproxy log.

Based on the previous data, a preliminary linear regression analysis is carried out to identify the most relevant TCP/IP metrics used as predictors. Then, a more refined regression analysis is performed to build the models to predict the four S-KPIs selected.

TABLE 3. Selected live video streams.

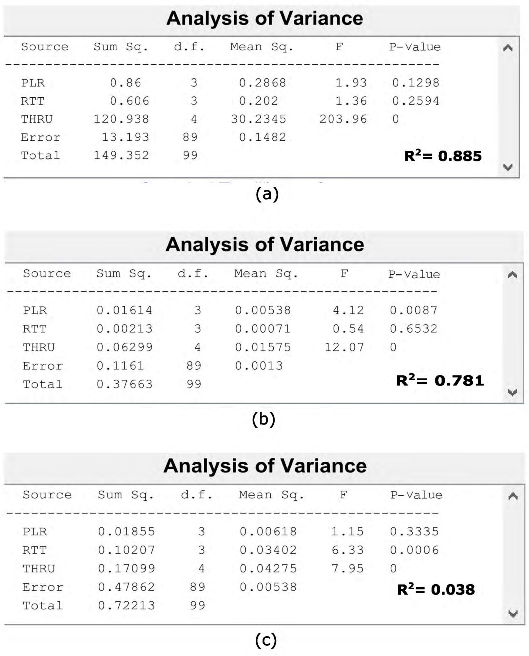
Live Video Url	Content	Motion Speed
https://www.youtube.com/watch?v=7dDnjTblFrI	Cartoon English songs live channel	High
https://www.youtube.com/watch?v=hDzrpintkxY	Caribbean beach live cam	Medium
https://www.youtube.com/watch?v=93MsUfNcnOI	News and politics live channel	High
https://www.youtube.com/watch?v=WslbFqATXtk	Hamptoms Island Ferry live cam	Medium
https://www.youtube.com/watch?v=BPXtn15_9qo	Venice Italy - Rialto Bridge live cam	High
https://www.youtube.com/watch?v=WjOGhNDX51M	Venice Italy - Campo Santa Maria live cam	Medium
https://www.youtube.com/watch?v=31ce1XGcGqw	Tucson Arizona - 4th Avenue live cam	High
https://www.youtube.com/watch?v=LgFJcJVy8GU	Netherlands - Rail Road live cam	High
https://www.youtube.com/watch?v=NuIAYHVeFYs	Nature videos live channel	Slow
https://www.youtube.com/watch?v=qRrr0QgemLk	Spanish cartoons live channel	High

1) IDENTIFICATION OF RELEVANT TCP/IP METRICS

In regression analysis, minimizing the number of variables (predictors) often leads to more robust models (i.e., with less overfitting). In this work, a feature selection process is carried out to identify the TCP/IP metrics with the largest impact on each video S-KPI. For this purpose, a classical Analysis of Variance (ANOVA) technique [51] is used. A regression model is built first with all indicators (referred to as full model). Then, an iterative process starts where the least significant variable is eliminated in each step. The more steps are executed, the simpler the model is, but the less accurate. The selection of the least significant variable in each step is done by computing Student's t and p-value statistics, T and P, for every variable across iterations. Roughly speaking, the p-value of a predictor represents the probability that its regression coefficient is 0. Thus, a predictor (i.e., TCP/IP metric) with large p-value tends to be less important for predicting the dependent variable (S-KPI).

The candidate TCP/IP metrics selected a priori for the full model are average downlink throughput (THRU), overall downlink packet loss ratio (PLR) and average round trip time (RTT). These indicators are closely related to effective transmission rates [52]. All of them are measured at IP layer and collected by the Probe software.

Figs. 4 (a)-(c) breaks down ANOVA statistics for three of the four considered S-KPIs, namely initial video play start time, video interruption frequency and video interruption duration. These statistics are obtained by analyzing the residuals from the regression analysis. It is observed that, for the three S-KPIs, THRU is the most significant predictor, as it has the lowest p-value for all of them. A closer analysis proves that the accuracy of the model with a single predictor (THRU) is nearly the same as with the full model. Specifically, the difference in the determination coefficient, R^2, between the full model and the simplified model based on THRU is less than 0.005 for any of the S-KPIs. From these results, it can be concluded that THRU is the most significant factor, providing enough information to estimate S-KPIs.



Analysis of Variance

```
Source   Sum Sq.   d.f.   Mean Sq.     F      P-Value
------------------------------------------------------
PLR        0.86      3     0.2868     1.93   0.1298
RTT        0.606     3     0.202      1.36   0.2594
THRU     120.938     4    30.2345   203.96   0
Error     13.193    89     0.1482
Total    149.352    99                    R²= 0.885
```

(a)

Analysis of Variance

```
Source   Sum Sq.   d.f.   Mean Sq.     F      P-Value
------------------------------------------------------
PLR      0.01614     3    0.00538     4.12   0.0087
RTT      0.00213     3    0.00071     0.54   0.6532
THRU     0.06299     4    0.01575    12.07   0
Error    0.1161     89    0.0013
Total    0.37663    99                    R²= 0.781
```

(b)

Analysis of Variance

```
Source   Sum Sq.   d.f.   Mean Sq.     F      P-Value
------------------------------------------------------
PLR      0.01855     3    0.00618     1.15   0.3335
RTT      0.10207     3    0.03402     6.33   0.0006
THRU     0.17099     4    0.04275     7.95   0
Error    0.47862    89    0.00538             R²= 0.038
Total    0.72213    99
```

(c)

FIGURE 4. ANOVA statistics. (a) Initial video play start time. (b) Video interruption duration. (c) Video interruption frequency.

to be tested with other videos. For this purpose, the measurement campaign is repeated with 10 different YouTube Live feeds from different streaming servers. Streams are carefully selected so as to ensure that video URLs are active during the whole measurement campaign and the video content is diverse enough to be representative of all conditions. Table 3 details the content and motion speed for the selected live videos. As observed in the table, their content covers very different scenes, from live video cameras in nature and public places to animated videos. The validation script reproduces sequentially the 10 live streams under the 80 different NetEm configurations. The total time is 2640 minutes (44 hours), split into 33 min per NetEm configuration and 3.3 minutes per live video.

C. MODEL ASSESSMENT

The proposed mapping from TCP/IP metrics to S-KPIs is built with measurements from a single video. To check the validity of these mapping functions, the resulting model has

D. QoE MODEL

To check the impact of errors when S-KPIs are estimated from TCP/IP metrics, QoE is also measured. To avoid the need for time-consuming subjective tests, a simple utility model is used to map S-KPI values to MOS figures. The model is based on the video QoE measurement standard U-vMOS (User/Unified/Ubiquitous video Mean Opinion Score), proposed by Huawei in 2016 [53]. U-vMOS model takes into account the dissatisfaction caused by initial reproduction time, stallings and poor image quality. Similarly, an overall MOS associated to the video experience of a session c is calculated here per live video stream and scenario as

$$vMOS(c) = 1 + (sQuality(c) - 1) \times \left(\frac{\beta_1(sInteraction(c) - 1) + \beta_2(sView(c) - 1)}{4(\beta_1 + \beta_2)} \right), \quad (4)$$

where $sQuality$ is the maximum MOS due to image quality (i.e., provided that the initial delay is 0 and there is no stalling), $sInteraction$ is the maximum MOS related to the loading time, $sView$ is the maximum MOS due to stall frequency and stall duration, and β_i are regression constants. As in [53], $\beta_1 = 0.71$ and $\beta_2 = 0.77$. Likewise, $sInteraction$ and $sView$ are computed from SPT, IF and IDR measurements with the mapping functions defined in [53]. Unlike in [53], $sQuality$ is not computed for a fixed image resolution, but from itag statistics. In [23], MOS is computed from itag statistics neglecting stallings. Following the same approach, MOS due to image quality is estimated from the histogram of itag values per session, obtained from HAR files generated by processing mitmproxy logs. For this purpose, a limited set of 20 live video sessions are established with very different NetEm settings. For each of these video sessions, a center of gravity (COG) of itag value is computed as

$$Cog(c) = \sum_{itag=91}^{96} W_{itag} R_{itag}(c)), \quad (5)$$

where W_{itag} is a weight proportional to the image quality for an itag value, ranging from 1 (lowest) to 6 (highest) as shown in Table 4, and $R_{itag}(c)$ is the relative frequency of each itag value for session c. Thus, a larger ratio of itag 96 in a session should translate into a larger $Cog(c)$ (i.e., closer to 6), denoting a higher image quality. Then, the MOS due to image quality is calculated as a simple linear Cog-to-MOS mapping as

$$sQuality = MOS_{min} + \left(MOS_{max} - MOS_{min}\right) \times \left(\frac{Cog(c) - Cog_{min}}{Cog_{max} - Cog_{min}} \right), \quad (6)$$

TABLE 4. Weights for itag values.

itag	W_{itag}
91	1
92	2
93	3
94	4
95	5
96	6

$$Cog_{min} = \min(Cog(c)), \quad (7)$$
$$Cog_{max} = \max(Cog(c)), \quad (8)$$

where MOS_{min} and MOS_{max} are the expected maximum and minimum MOS values considering all criteria (in this work, 1.5 and 4.5, respectively, taken from [23], [53]), and Cog_{min} and Cog_{max} are the maximum and minimum Cog values of the 20 sessions used to derive the Cog-to-MOS mapping.

VI. RESULTS

For clarity, the regression analysis performed to build the different S-KPI models is described first and model validation is presented later.

A. MODEL CONSTRUCTION

First, Section V-B1 justified why S-KPIs are estimated only from THRU measurements. Then, the regression models reflecting the impact of THRU on each S-KPI are presented.

1) VIDEO PLAY START TIME

Fig. 5 shows a scatter plot of the video play start time (SPT) versus THRU. Each of the 80 points corresponds to the average of 10 video sessions of the same live feed for the same network state (NetEm configuration). The segmented regression curve that best fits the data is superimposed, resulting in a determination coefficient of $R^2 = 0.93$. In the figure, it is observed that points are grouped in columns that match the bandwidth limitations (throttling values in NetEm). The regression curve shows how SPT decreases when THRU increases. As explained in [20], video reproduction starts when the client buffer is filled up to a certain threshold. The time to fill the buffer increases as the transmission rate decreases. In [11], the average initial video play start time reported for conventional (i.e., non-real time) YouTube services was 2.6 s, with a maximum of 11.9 s. Results here show an average initial delay between 3 and 6.8 s for YouTube Live.

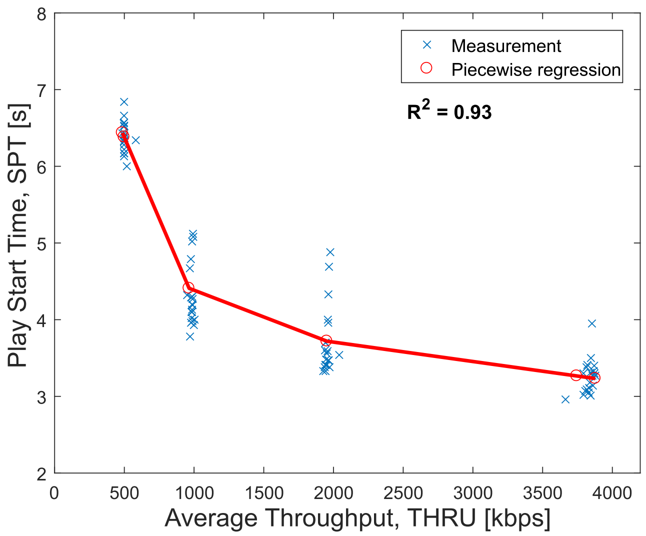


FIGURE 5. Video play start time versus average session throughput.

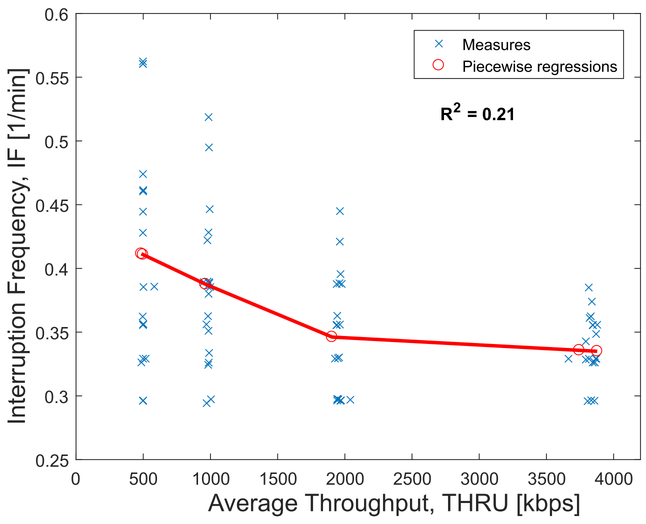


FIGURE 6. **Interruption frequency versus average session throughput.**

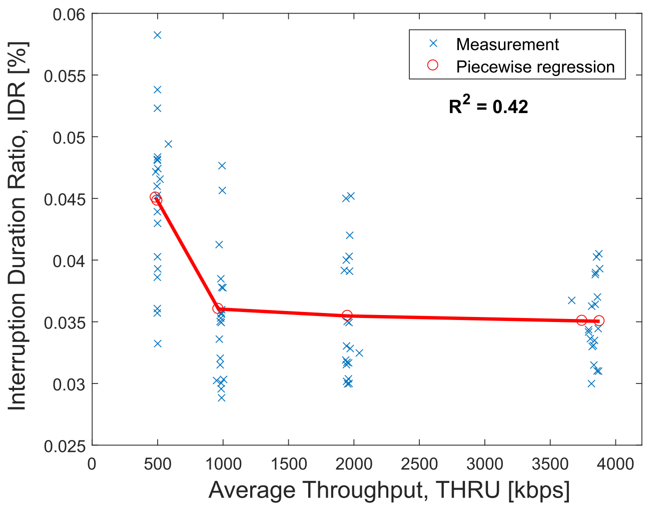


FIGURE 7. **Interruption duration ratio versus average session throughput.**

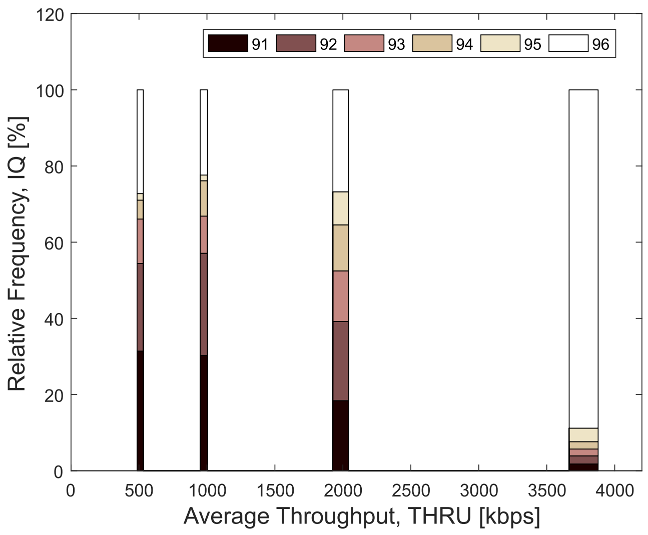


FIGURE 8. **Distribution of itag values versus average session throughput.**

The larger variability for THRU = 2000 kbps is due to stochastic noise.

2) VIDEO INTERRUPTION FREQUENCY

Fig. 6 shows a scatter plot of video interruption frequency (IF) and THRU. As in Fig. 5, each point corresponds to the average of 10 video sessions for the same network state. The segmented regression curve that best fits the data is superimposed. A quick look at the y-axis reveals medium IF values (lower than 0.6 rebuffering events per minute for the smallest THRU values). Unexpectedly, IF does not tend to zero for large THRU values. A closer analysis of HAR traces reflects the existence of one rebuffering event in all live sessions coincident with the video playback sequence start, which is the reason for the non-negligible IF value for large THRU (>0.3 = 1stalling / 3.3 min video duration). In the rest of the session, HAS prevents buffer underruns, as inferred from the small IF increment for small THRU (i.e., 0.41 for 500 kbps). It should be pointed out that, even if model accuracy is low ($R^2 = 0.21$) and IF is not negligible, it will be shown next that the total interruption duration is short, having a limited impact on QoE. Thus, IF is discarded as a relevant metric, since, in this case, it is not a good indicator of QoE.

3) VIDEO INTERRUPTION DURATION RATIO

Fig. 7 shows a scatter plot of video interruption duration ratio (IDR) and THRU, together with the segmented regression curve. Again, it is observed that, with HAS, the percentage of time the user is in the rebuffering state is small (less than 0.06%) even for low THRU values. Only for 500 kbps, the average IDR increases up to 0.045%. The regression determination coefficient is $R^2 = 0.42$. A closer analysis (not shown here) reflects that the maximum total interruption duration is 160 ms (equivalent to only 4 frames with a video frame rate of 25 Hz).

4) IMAGE QUALITY DISTRIBUTION

Fig. 8 represents a stacked bar graph with the itag distribution for the different session throughputs. Column width reflects THRU ranges for each throttling value. The y-axis shows the percentage of itags of each class within each column. First, it can be observed that only 6 itags values appear (91, 92, 93, 94, 95, 96), with 91 and 96 the minimum and maximum resolutions and also the most frequent values. As expected, the share of itag 96 (high image quality) increases as THRU increases. For low THRU values, an alternating behavior between the maximum and minimum itag values is observed, although low-resolution itags prevail.

5) QoE

Fig. 9 shows the vMOS estimated by the QoE model for a live video streaming session depending on the average session throughput resulting from the 80 different network states.

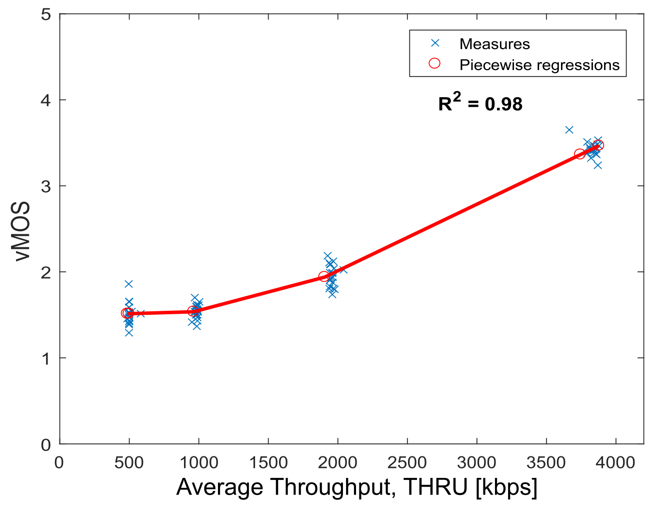


FIGURE 9. Estimated QoE versus average session throughput.

Each point corresponds to the average of 10 video sessions for the same network state. The segmented regression curve is also superimposed. As expected, QoE decreases as THRU decreases. Specifically, a THRU of 4 Mbps results in a vMOS close to 3.5, below the maximum of 4.5. An inspection of Fig. 4 shows that values of THRU above 4 Mbps lead to $SPT \approx 3$ s, $IF = IDR \approx 0$ and $Cog(c) \approx 6$, resulting in $sInteraction(c) \approx 2.86$, $sView(c) \approx 4.98$ and $sQuality(c) = 4.28$. Thus, it is concluded that the 3-second initial delay limits the maximum QoE to 3.5. In contrast, a THRU of 0.5 Mbps leads to vMOS slightly larger than 1, mainly due to a low image quality (on average, $sQuality(c) = 1.92$ versus $sInteraction(c) = 1.51$ and $sView(c) = 4.85$). Also note that the coefficient of determination is $R^2 = 0.98$.

B. MODEL ASSESSMENT

The above-presented regression models, derived for a single live video stream, are tested over 10 different streams. For each S-KPI, the regression curve is first superimposed with real measurements to check if the model captures the trend with THRU. Then, a scatter plot compares measurements and estimates.

1) VIDEO PLAY START TIME

Fig. 10.(a) plots the SPT for the 10 live video streams with the 80 different network states (SPT-10). The four center points (circles) correspond to the vertical (SPT) and horizontal (THRU) mean of measurements for each throttling value (0.5/1/2/4 Mbps). The error bars show the 10% and 90% confidence interval of that mean. The segmented regression curve obtained in section VI-A1 is superimposed. It is observed that the considered model captures the dependence on THRU very well. Fig. 10.(b) compares SPT estimates and measurements. SPT estimates are obtained from THRU measurements by using the piecewise regression curve in Fig. 10.(a). In the figure, it is observed that estimation is accurate for most of the samples since the regression equation is reasonably close to the identity (specifically, y = 0.96x+0.19), with a coefficient of determination $R^2 = 0.92$.

2) VIDEO INTERRUPTION FREQUENCY

Fig. 11.(a) plots the interruption frequency measured for the 10 live video streams (IF-10). Again, each circle represents the mean IF and THRU of measurements for each throttling value, while the error bars show the 10% and 90% confidence intervals. The segmented regression curve obtained in section VI-A2 is overlapped. It is observed that measurements and model estimates (regression curve) fit reasonably well.

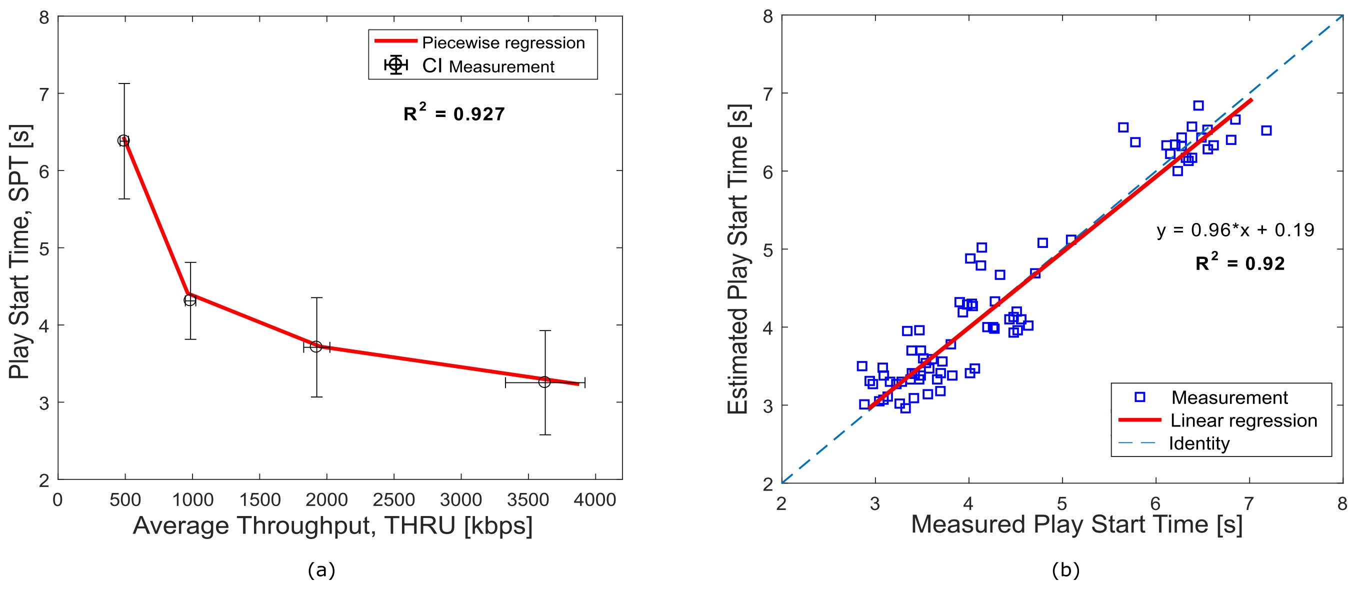
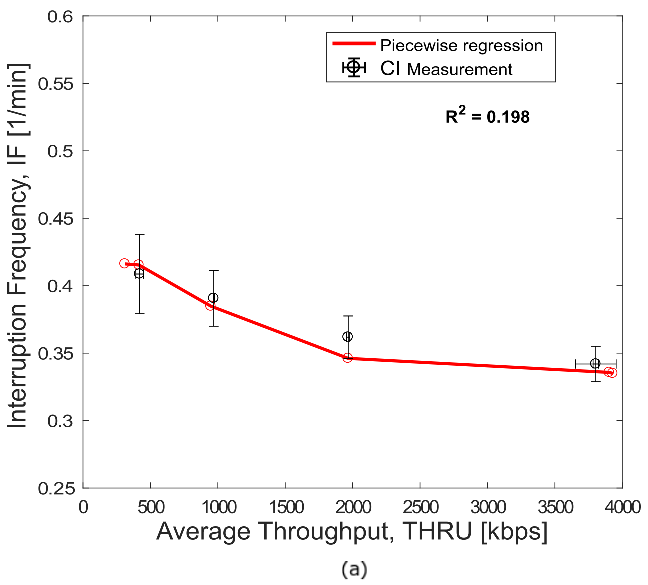


FIGURE 10. Video play start time of 10 live video streams. (a) Dependence on THRU. (b) Measurements vs estimates.



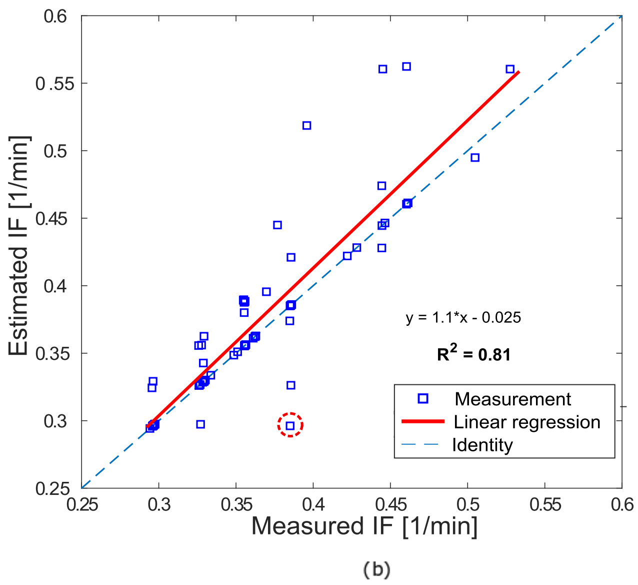
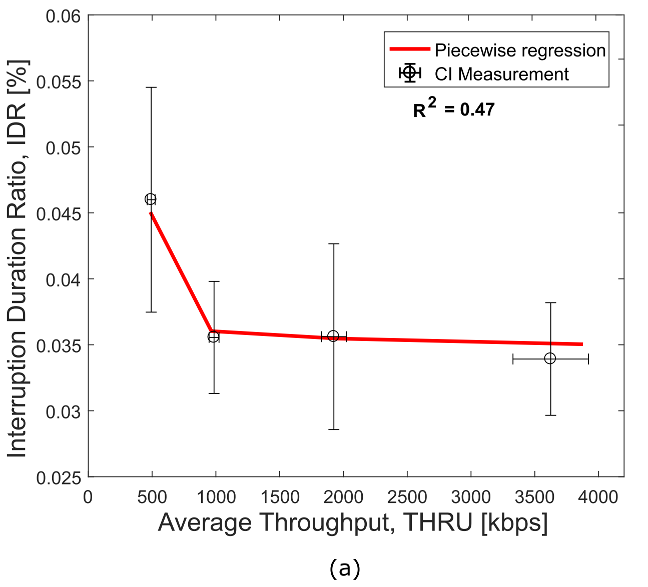


FIGURE 11. Video interruption frequency of 10 live video streams. (a) Dependence on THRU. (b) Measurements vs estimates.



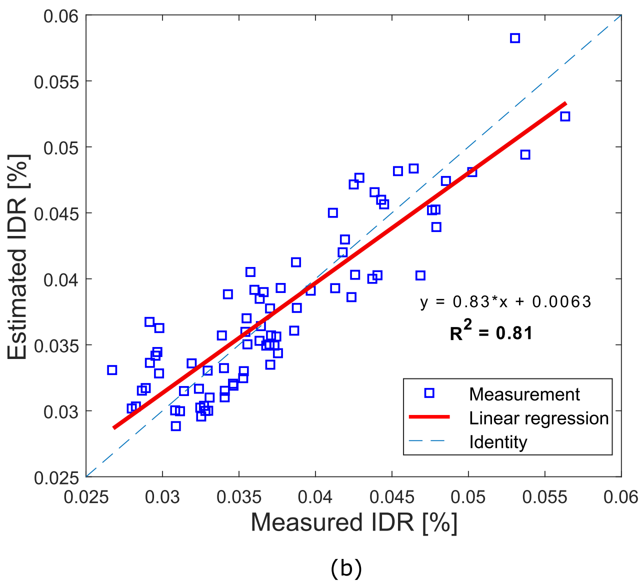


FIGURE 12. Video interruption duration ratio of 10 live video streams. (a) Dependence on THRU. (b) Measurements vs estimates.

Fig. 11.(b) compares IF measurements vs estimates obtained with the piecewise regression curve in Fig. 11.(a). It is observed that line equation is close to the identity ($y = 1.1x + 0.025$) and $R^2 = 0.81$. A more detailed analysis shows that the problematic sessions show large fluctuations around their average THRU value. In particular, the outlier highlighted by a dotted circle corresponds to a session that experienced an interruption of 7.4 seconds in video transmission due to a brief change of content server.

3) VIDEO INTERRUPTION DURATION RATIO

Fig. 12.(a) plots the IDR for the 10 live video streams with the different network states (IDR-10). The segmented regression curve obtained in section VI-A3 is superimposed. Fig. 12.(b) compares IDR measurements and estimates obtained with the piecewise regression curve in Fig. 12.(a). In the figure, it is observed that estimation is accurate for most of the samples, since the regression equation is reasonably close to the identity ($y = 0.83x + 0.0063$), with a determination coefficient of $R^2 = 0.81$.

4) IMAGE QUALITY

Fig. 13 compares the distribution of itag values observed in the 10 live video streams against estimates raised with the model derived from a single video. Measurements are represented with a trend line and error bars showing confidence

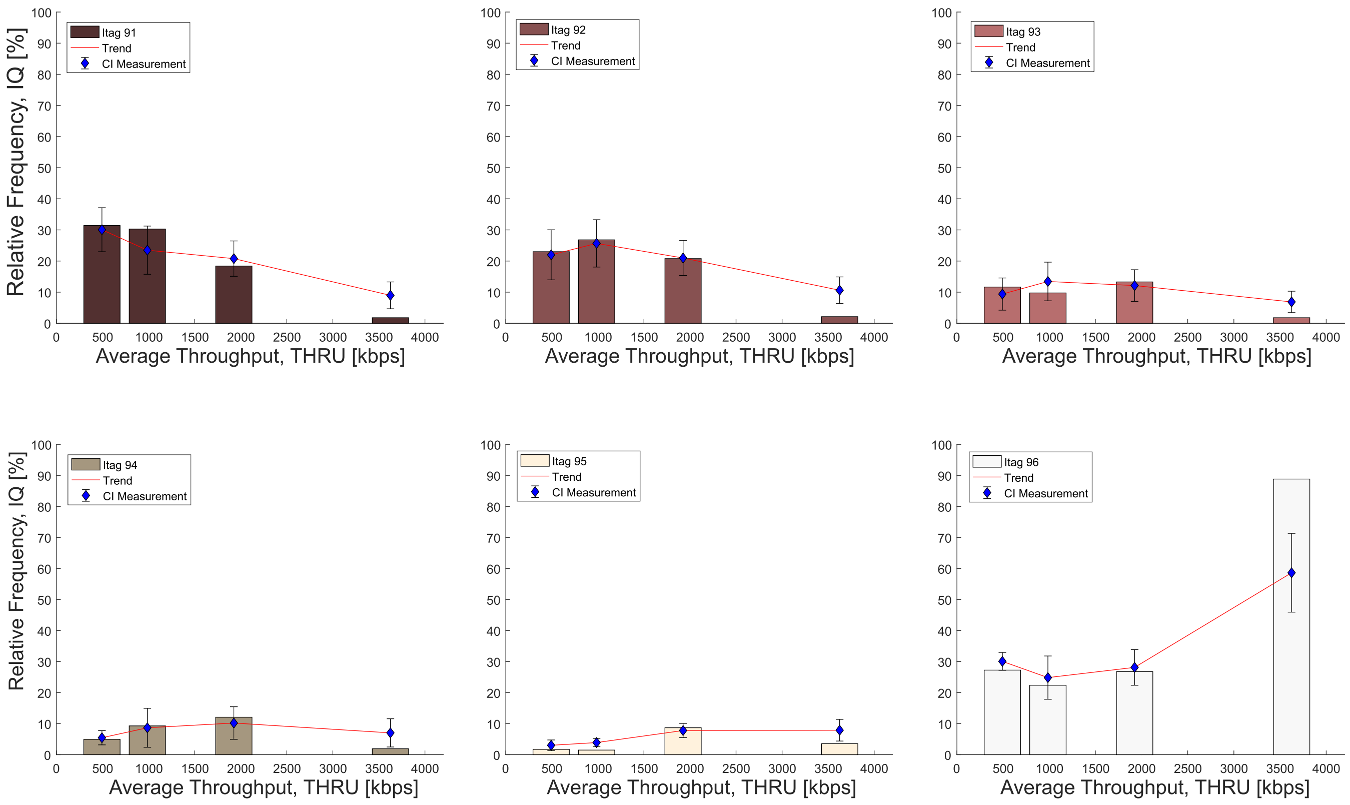


FIGURE 13. Itag distribution versus throughput for 10 live video streams.

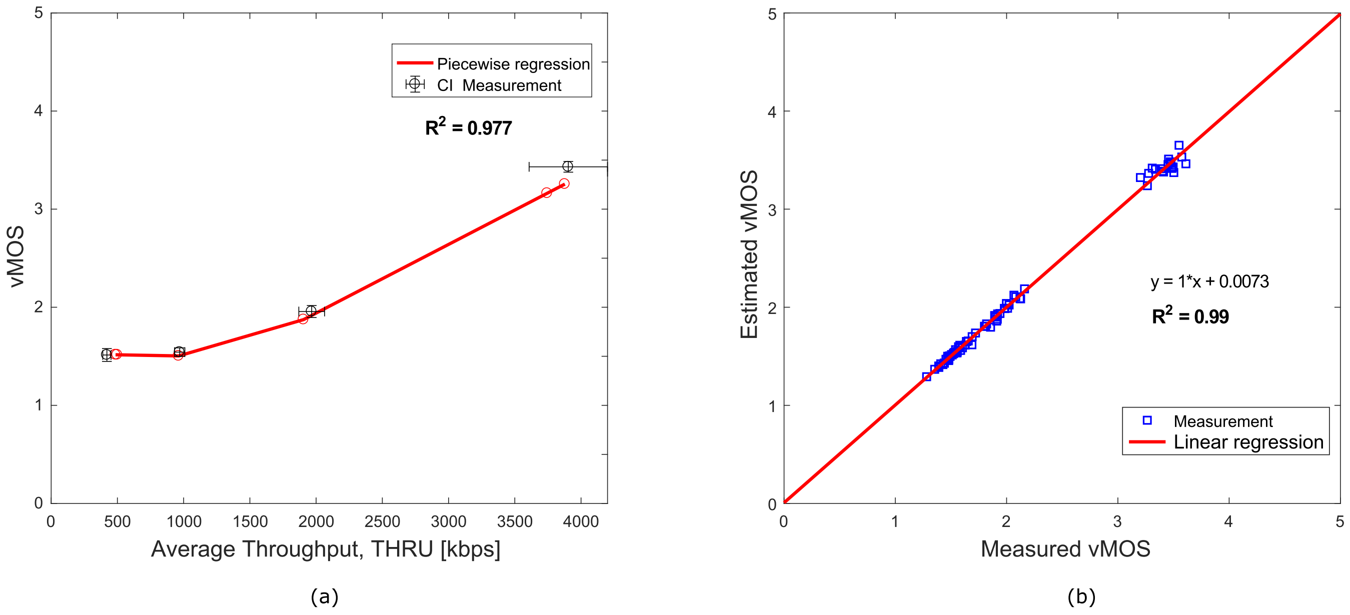


FIGURE 14. QoE for 10 live video streams. (a) Dependence on THRU. (b) Measurements vs estimates.

intervals, and estimates are shown by bars. For clarity, each itag value is shown in a different figure. Again, it is observed that, for the 10 videos, the most frequent itags are 91, 92 and 96. With higher bitrates, more itag 96 values appear, corresponding to better image quality; with lower bitrates, the itag values are distributed between 91, 92 and 96.

5) QoE

Finally, Fig. 14.(a) shows the average vMOS for the 10 live video streams and 80 network conditions (QoE-10). The segmented regression curve obtained in VI-A5 is superimposed. Fig. 14.(b) compares QoE measurements vs estimates. QoE measurements are obtained by applying the U-vMOS model with real S-KPI measurements taken by the terminal agent and the analysis of protocol messages. In contrast, QoE estimates are obtained by applying the U-vMOS model with S-KPI estimates derived from average THRU measurements taken by the network probe. Such S-KPI estimates are obtained with the piecewise regression models shown in Fig. 5 (SPT), 7 (IDR) and 13 (IQ).

In the figure, it is observed that estimation is very accurate since the regression equation is close to the identity and the coefficient of determination $R^2 = 0.99$.

VII. CONCLUSIONS

In this article, a model to estimate the QoE of encrypted YouTube Live service from packet-level data collected in the interfaces of a wireless network has been presented. The model has been derived using an experimental platform consisting of a live video streaming server, a user terminal agent, a Wi-Fi wireless network, a network-level emulator, a probe software and a man-in-the-middle proxy. The latter is used to obtain itag statistics, from which to infer image quality. With that platform, a preliminary analysis of the differences between conventional and live YouTube services has been carried out to justify the need for new performance models. The analysis of collected data has shown the strong correlation of most S-KPIs with average session throughput. Likewise, measurements have confirmed the ability of adaptive streaming to significantly decrease the number of video interruptions, converting image quality as a more meaningful indicator of the user experience. These results are consistent with QoE statistics of the conventional YouTube service with HAS over mobile networks presented in [22] . An important difference observed in live streams is the existence of an initial rebuffering event just after video play start.

The provided regression curves can be used as a black box model to monitor the QoE of encrypted video streaming services on a session basis on a large scale. Such an approach is suitable for network operators that do not have access to application layer measurements. The model can easily be integrated into a centralized big data platform with access to key network interfaces (e.g., S11 and S1-U in 4G/5G non-standalone systems). The proposed methodology can be extended to different radio access technologies. Nonetheless, it might have to be updated as new transport-layer schemes are introduced (e.g., HTTP/2 protocol [54]). The analysis of the uplink from the video source to the server is left for future work.

REFERENCES

[1] Ericsson. *Ericsson Mobility Report*. Accessed: Nov. 2018. [Online]. Available: https://www.ericsson.com/assets/local/mobility-report/documents/2018/ericsson-mobility-report-june-2018.pdf/

[2] R. El Hattachi and J. Erfanian, "Next generation mobile networks (NGMN) alliance: 5G white paper," no. 1, 2015.

[3] A. Banerjee, "Revolutionizing CEM with subscriber-centric network operations and QoE strategy white paper accanto systems," Heavy Reading, no. 1, 2014.

[4] V. F. Monteiro, D. A. Sousa, T. F. Maciel, F. R. M. Lima, E. B. Rodrigues, and F. R. P. Cavalcanti, "Radio resource allocation framework for quality of experience optimization in wireless networks," *IEEE Netw.*, vol. 29, no. 6, pp. 33–39, Nov./Dec. 2015.

[5] P. Oliver-Balsalobre, M. Toril, S. Luna-Ramírez, and R. G. Garaluz, "Self-tuning of service priority parameters for optimizing quality of experience in LTE," *IEEE Trans. Veh. Technol.*, vol. 67, no. 4, pp. 3534–3544, Apr. 2018.

[6] H. Nam, K.-H. Kim, J. Y. Kim, and H. Schulzrinne, "Towards QoE-aware video streaming using SDN," in *Proc. IEEE Global Commun. Conf.*, Dec. 2014, pp. 1317–1322.

[7] *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021*. Accessed: Nov. 2018. [Online]. Available: https://https://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/white-paper-listing.html

[8] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "YouTube traffic characterization: A view from the edge," in *Proc. 7th ACM SIGCOMM Conf. Internet Meas.*, 2007, pp. 15–28.

[9] O. Oyman and S. Singh, "Quality of experience for HTTP adaptive streaming services," *IEEE Commun. Mag.*, vol. 50, no. 4, pp. 20–27, Apr. 2015.

[10] A. Diaz, P. Merino, and F. J. Rivas, "Customer-centric measurements on mobile phones," in *Proc. IEEE Int. Symp. Consum. Electron.*, Apr. 2008, pp. 1–4.

[11] M. Seufert, N. Wehner, F. Wamser, P. Casas, A. D'Alconzo, and P. Tran-Gia, "Unsupervised QoE field study for mobile YouTube video streaming with YoMoApp," in *Proc. 9th Int. Conf. Qual. Multimedia Exper. (QoMEX)*, Jun. 2017, pp. 1–6.

[12] A. Raake, J. Gustafsson, S. Argyropoulos, M.-N. Garcia, D. Lindegren, G. Heikkilá, M. Pettersson, P. List, and B. Feiten, "IP-based mobile and fixed network audiovisual media services," *IEEE Signal Process. Mag.*, vol. 28, no. 6, pp. 68–79, Nov. 2011.

[13] *Subjective Video Quality Assessment Methods for Multimedia Applications*, document ITU-T P.910, Apr. 2008.

[14] F. Ricciato, "Traffic monitoring and analysis for the optimization of a 3G network," *IEEE Wireless Commun.*, vol. 13, no. 6, pp. 42–49, Dec. 2006.

[15] P. Casas, M. Seufert, and R. Schatz, "YOUQMON: A system for on-line monitoring of YouTube QoE in operational 3G networks," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 2, pp. 44–46, Dec. 2013.

[16] A. Baer, P. Casas, A. D'Alconzo, P. Fiadino, L. Golab, M. Mellia, and E. Schikuta, "DBStream: A holistic approach to large-scale network traffic monitoring and analysis," *Comput. Netw.*, vol. 107, pp. 5–19, Oct. 2016.

[17] L. Skorin-Kapov, M. Varela, T. Hoßfeld, and K.-T. Chen, "A survey of emerging concepts and challenges for QoE management of multimedia services," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 14, no. 2s, p. 29, 2018.

[18] A. Rao, A. Legout, Y.-S. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proc. 7th ACM Conf. Emerg. Netw. Exp. Technol.*, 2011, p. 25.

[19] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "YouTube everywhere: Impact of device and infrastructure synergies on user experience," in *Proc. ACM SIGCOMM Conf. Internet Meas.*, 2011, pp. 345–360.

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

[21] R. K. Mok, E. W. Chan, and R. K. Chang, "Measuring the quality of experience of HTTP video streaming," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, May 2011, pp. 485–492

[22] F. Wamser, P. Casas, M. Seufert, C. Moldovan, P. Tran-Gia, and T. Hossfeld, "Modeling the YouTube stack: From packets to quality of experience," *Comput. Netw.*, vol. 109, pp. 211–224, Nov. 2016.

[23] J. De Vriendt, D. De Vleeschauwer, and D. Robinson, "Model for estimating QoE of video delivered using HTTP adaptive streaming," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, May 2013, pp. 1288–1293.

[24] Mitmproxy. *An Interactive HTTPS Proxy*. Accessed: Nov. 2018. [Online]. Available: https://mitmproxy.org/

[25] H. J. Kim, D. G. Yun, H.-S. Kim, K. S. Cho, and S. G. Choi, "QoE assessment model for video streaming service using QoS parameters in wired-wireless network," in *Proc. 14th Int. Conf. Adv. Commun. Technol. (ICACT)*, 2012, pp. 459–464.

[26] T. Hoßfeld, R. Schatz, E. Biersack, and L. Plissonneau, "Internet video delivery in YouTube: From traffic measurements to quality of experience," in *Data Traffic Monitoring and Analysis*. Springer, 2013, pp. 264–301.

[27] P. Casas, R. Schatz, and T. Hoßfeld, "Monitoring YouTube QoE: Is your mobile network delivering the right Experience to your customers?" in *Proc. WCNC*, 2013, pp. 1609–1614.

[28] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner, "Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming," in *Proc. IEEE 6th Int. Workshop Qual. Multimedia Exper. (QoMEX)*, Sep. 2014, pp. 111–116.

[29] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP adaptive streaming," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 469–492, 1st Quart., 2015.

[30] M. Claeys, S. Latre, J. Famaey, and F. De Turck, "Design and evaluation of a self-learning HTTP adaptive video streaming client," *IEEE Commun. Lett.*, vol. 18, no. 4, pp. 716–719, Apr. 2014.

[31] W. Huang, Y. Zhou, X. Xie, D. Wu, M. Chen, and E. Ngai, "Buffer state is enough: Simplifying the design of QoE-aware HTTP adaptive video streaming," *IEEE Trans. Broadcast.*, vol. 64, no. 2, pp. 590–601, Jun. 2018.

[32] E. Demirbilek and J.-C. Grégoire, "Machine learning–based parametric audiovisual quality prediction models for real-time communications," *ACM Trans. Multimedia Comput.*, vol. 13, no. 2, pp. 16:1–16:25, Mar. 2017.

[33] P. Casas, A. D'Alconzo, F. Wamser, M. Seufert, B. Gardlo, A. Schwind, P. Tran-Gia, and R. Schatz, "Predicting QoE in cellular networks using machine learning and in-Smartphone measurements," in *Proc. 9th IEEE Int. Conf. Qual. Multimedia Exper. (QoMEX)*, Jun. 2017, pp. 1–6.

[34] W. Robitza, M.-N. Garcia, and A. Raake, "A modular HTTP adaptive streaming QoE model—Candidate for ITU-T P.1203 ('P.NATS')," in *Proc. 9th IEEE Int. Conf. Qual. Multimedia Exper. (QoMEX)*, Jul. 2017, pp. 1–6.

[35] W. Pan and G. Cheng, "QoE assessment of encrypted YouTube adaptive streaming for energy saving in smart cities," *IEEE Access*, vol. 6, pp. 25142–25156, 2018.

[36] Z. Duanmu, A. Rehman, and Z. Wang, "A quality-of-experience database for adaptive video streaming," *IEEE Trans. Broadcast.*, vol. 64, no. 2, pp. 474–487, Jun. 2018.

[37] P. Oliver, M. Toril, S. Luna, and R. García, "A system testbed for modeling encrypted video-streaming service performance indicators based on TCP/IP metrics," *EURASIP J. Wireless Commun. Netw.*, vol. 2017, p. 213, Dec. 2017.

[38] Telestream. *Wirecast User Manual*. Accessed: Nov. 2018. [Online]. Available: www.telestream.net/application-content/wirecast/help/7-3/win/Wirecast-User-Guide-Windows.pdf

[39] YouTube. *Creator Studio*. Accessed: Nov. 2018. [Online]. Available: https://support.google.com/youtube/answer/6060318

[40] Ascom. *TEMS Pocket Specifications*. Accessed: Nov. 2018. [Online]. Available: http://www.tems.com/products-for-radio-and-core-networks/radio-network-engineering

[41] Linux Foundation. *NetEm*. Accessed: Nov. 2018. [Online]. Available: https://wiki.linuxfoundation.org/netem

[42] A. Jurgelionis, J.-P. Laulajainen, M. Hirvonen, and A. I. Wang, "An empirical study of NetEm network emulation functionalities," in *Proc. 20th IEEE Int. Conf. Comput. Commun. Netw. (ICCCN)*, Aug. 2011, pp. 1–6.

[43] TCPDUMP-Workers. *TCPDUMP*. Accessed: Nov. 2018. [Online]. Available: http://www.tcpdump.org/

[44] I. M. Froseth, S. Leicht, R. J. Reimer, and V. T. Tran, "Obtaining a Video Dataset from YouTube via DASH," 2015.

[45] YouTube. *Live Encoder Settings, Bitrates, and Resolutions*. Accessed: Nov. 2018. [Online]. Available: https://support.google.com/youtube/answer/2853702?hl=en

[46] Ascom. *TEMS Pocket User Manual*. Accessed: Nov. 2018. [Online]. Available: https://goo.gl/wjzxtV

[47] A. Mondal, S. Sengupta, B. R. Reddy, M. J. V. Koundinya, C. Govindarajan, P. De, N. Ganguly, and S. Chakraborty, "Candid with YouTube: Adaptive streaming behavior and implications on data consumption," in *Proc. ACM Proc. 27th Workshop Netw. Oper. Syst. Support Digit. Audio Video*, 2017, pp. 19–24.

[48] A. Golub. *Reverse-Engineering YouTube*. Accessed: Nov. 2018. [Online]. Available: https://tyrrrz.me/Blog/Reverse-engineering-YouTube

[49] YouTube. *Delivering Live Youtube Content via DASH*. Accessed: Nov. 2018. [Online]. Available: https://developers.google.com/youtube/v3/live/guides/encoding-with-dash

[50] R. Pantos and W. May, *HTTP Live Streaming*, document RFC 8216, 2017, pp. 1–60.

[51] W. Navidi, *Statistics for Engineers and Scientists*. New York, NY, USA: McGraw-Hill, 2011.

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

[53] Huawei. *Video Experience-Based Bearer Network Technical White Paper*. Accessed: Mar. 2019. [Online]. Available: https://www.huawei.com/~/media/CORPORATE/PDF/white%20paper/video-experience-based-bearer-network-technical-whitepaper

[54] Z. Xu, X. Zhang, and Z. Guo, "QoE-driven adaptive K-push for HTTP/2 Live Streaming," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.

LUIS ROBERTO JIMÉNEZ received the M.S. degree in electronics and communications engineering from the Santo Domingo Institute of Technology (Intec), Santo Domingo, Dominican Republic, in 2013, and the M.S.E. degree in telematics and telecommunication networks from the University of Malaga, Málaga, Spain, in 2015, where he is currently pursuing the Ph.D. degree in telecommunications engineering. His current research interests include self-optimization networks and performance evaluation of multimedia services over mobile networks based on customer experience. He was a recipient of the Junta de Andalucía Scholarship (2017–2021) over methods planning and optimizing QoE in B4G networks.

MARTA SOLERA received the M.Sc. and Ph.D. degrees in telecommunication engineering from the Polytechnic University of Catalonia (UPC), in 1996 and 2006, respectively. She is currently an Associate Professor with the Department of Communication Engineering, Universidad de Malaga (UMA). Since 1996, she has been a Lecturer in several universities, such as UPC, the Universidad Nacional Autonoma de Mexico (UNAM), and UMA. She has been involved in several public funded national research projects in the field of multimedia and mobile communications. Her research interest includes design and performance evaluation of multimedia services over mobile networks.

MATÍAS TORIL received the M.S. and Ph.D. degrees in telecommunication engineering from the University of Malaga, Spain, in 1995 and 2007, respectively. Since 1997, he has been a Lecturer with the Communications Engineering Department, University of Malaga, where he is currently an Associate Professor. He has authored more than 100 publications in leading conferences and journals, and he holds three patents owned by Nokia Corporation. His current research interests include self-organizing networks, radio resource management, and graph partitioning.