EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

A system testbed for modeling encrypted video-streaming service performance indicators based on TCP/IP metrics

Pablo Oliver-Balsalobre[1*], Matías Toril[1], Salvador Luna-Ramírez[1] and Rafael García Garaluz[2]

Abstract
For cellular operators, estimating the end-user experience from network measurements is a challenging task. For video-streaming service, several analytical models have been proposed to estimate user opinion from buffering metrics. However, there remains the problem of estimating these buffering metrics from the limited set of measurements available on a per-connection basis for encrypted video services. In this paper, a system testbed is presented for automatically constructing a simple, albeit accurate, Quality-of-Experience (QoE) model for encrypted video-streaming services in a wireless network. The testbed consists of a terminal agent, a network-level emulator, and Probe software, which are used to compare end-user and network-level measurements. For illustration purposes, the testbed is used to derive the formulas to compute video performance metrics from TCP/IP metrics for encrypted YouTube traffic in a Wi-Fi network. The resulting formulas, which would be the core of a video-streaming QoE model, are also applicable to cellular networks, as the test campaign fully covers typical mobile network conditions and the formulas are partly validated in a real LTE network.

Keywords: Testbed, S-KPIs, Estimation, Modeling, YouTube

1 Introduction

In the next few years, it is expected that the traffic in mobile broadband networks continues its exponential growth. It is foreseen that, in the period between 2016 and 2022, mobile communications traffic will dominate even more than it does today, experiencing an eightfold raise in half a decade [1]. In parallel, user expectations are increasing due to the availability of more sophisticated terminals and the large variety of services offered. Because of this constant evolution, mobile operators have been forced to change the way of managing their networks, from a network-centric approach focused on network performance to a more modern user-centric approach focused on user experience. Thus, the upcoming fifth generation on mobile technology (5G) should address these challenges, when mobile traffic will be clearly dominated by video services, accounting for 70% of the total traffic demand [1, 2], and Quality-of-Experience (QoE) management will be of the utmost importance [3].

Amongst video services, YouTube is the predominant service provider by far, representing 40–70% of total video traffic (depending on the network considered) [1]. For this reason, a deep understanding of the characteristics of this application has become mandatory to monitor and control the QoE perceived by the great amount of users consuming videos [4–6]. Thus, it has become mandatory to understand the relationship between service performance indicators and end-user experience, known as a QoE model. In [6], the YouTube stack at different layers is characterized and modeled, going from the generated network traffic to the QoE perceived by the users watching YouTube videos. In line with the work presented here, the buffer level at the YouTube application layer is presented as the central parameter of the models introduced there. A more classical approach is to perform surveys with real observers under different network conditions in controlled lab environments. The main considerations to take into account when choosing the surveys' path is deeply described in [7]. The result of these surveys is a

*Correspondence: pob@ic.uma.es
[1]Ingeniería de Comunicaciones, Universidad de Málaga, Campus de Teatinos S/N, 29071 Málaga, Spain
Full list of author information is available at the end of the article

Mean Opinion Score (MOS) for each network condition. As surveys are impractical for large-scale monitoring, several works have proposed different ways of estimating the QoE of video-streaming services.

QoE monitoring methods can be classified into application-layer and network-layer approaches. Application-layer monitoring leads to accurate QoE measurements as it checks the service performance at the sides of the connection, but requires installing specific software at the terminal. Terminal agents used by operators in drive tests fall into this class. Alternatively, network-layer monitoring provides a highly scalable solution for network and service providers, but it has the problem of identifying relevant Transmission Control Protocol (TCP) and Internet Protocol (IP) metrics on a per-connection basis from which user QoE can be estimated [8–10]. In [8], a user-centric approach for evaluating YouTube (and Facebook) performance is presented, relying on the QoE assessed by a group of mobile broadband users and network-layer Quality-of-Service (QoS) measurements collected in a field trial. It is demonstrated that the ratio between video bitrate and downlink bandwidth has strong impact on user experience, which is in line with the findings shown in this work. However, Casas et al. [8] use laptops as terminals. This difference possesses important implications in software employed and traffic behavior, and, thus, conclusions could not be still valid when mobile terminals are involved. Similarly, in [9], a buffer-level estimation algorithm based on TCP flow is proposed to estimate the QoE of a video user in terms of the expected number and duration of stalling events. In [10], the impact of delivery via the Internet on the QoE of YouTube video-streaming is quantified, comparing also different QoE monitoring approaches and evaluating QoE estimation accuracy.

Instead of estimating QoE from TCP/IP metrics, the vast majority of QoE video-streaming models in the literature (e.g., [11–13]) rely on the estimation of intermediate Service Key Performance Indicators (S-KPIs) (a.k.a. key quality indicators). For buffered video-streaming services, such as YouTube, the main S-KPIs are the initial buffering time (i.e., time elapsed from clicking play until the first image appears on the screen) and the number and duration of re-buffering events (i.e., how many times and for how long the video freezed)[14]. In the past, these S-KPIs could be derived from the analysis of higher-layer protocol messages (e.g., HyperText Transfer Protocol, HTTP). However, content providers are becoming increasingly aware of security and confidentiality risks and hence the introduction of encryption in their services. In 2016, Google reported that 97% of YouTube traffic was encrypted via HTTP Secure (HTTPS) connections with Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. Such an encryption process makes it difficult to compute S-KPIs. One of the few options left is analyzing TCP/IP traffic to isolate those fundamental parameters needed to estimate S-KPIs. TCP video-streaming traffic analysis have already been done with different purposes [15, 16]. In recent works [17], TCP/IP metrics have proved to be very valuable to obtain YouTube S-KPIs and, based on them, QoE perceived by the user. Alternatively, terminal agents can be used to obtain accurate S-KPI measurements. By accessing the corresponding Application Programming Interface (API), these agents can analyze decrypted protocol messages. Due to its success, YouTube is included in the most popular terminal agents [18–21]. Nevertheless, terminal agents are not a scalable monitoring solution as installing any background application needs authorization from the customer, which is rarely granted. Thus, terminal agents are only used by operators for validation purposes, as a trusted source to assess S-KPI measurements obtained by other means.

Due to constant increase of encryption, it is becoming more a more important to make a testing environment available, in order to analyze YouTube traffic and try to obtain meaningful information from it. In [22], a testing framework for analyzing encrypted video streams is presented. Its approach is based on a man-in-the-middle proxy for storing the decrypted video bitstream, active probing and traffic shaping. Furthermore, the influence of the man-in-the-middle proxy on KPIs for video streaming quality is also monitored (a constant offset is created). In [23], a laboratory tested similar to the one presented in this work is shown. A system called YouQ was developed to monitor and analyze application-layer KPIs and corresponding traffic traces. Then, machine learning techniques are applied to classify video instances and, based on that classification, estimate QoE associated. Nevertheless, QoE classes proposed in [23] are just experimental (i.e., non-standardized) and it has been left for future work further testing using network traces from real mobile networks.

In this paper, a system testbed is presented for automatically constructing a simple, albeit accurate, QoE model for encrypted video-streaming services in a wireless network. The testbed allows to (a) emulate video user interaction through a smartphone with a terminal agent, (b) change link conditions as in the radio or core network by a network-level emulator, (c) process all the traffic at different layers by network probe software [24] to obtain significant TCP/IP metrics, and, more importantly, (d) compare end-user and network-level measurements by correlating the output of the Probe and the terminal agent. For illustration purposes, the testbed is used to derive the formulas to compute video performance metrics from TCP/IP metrics for encrypted YouTube traffic in a Wi-Fi network. The resulting formulas, which would be the core of a video-streaming QoE model, are also applicable to cellular networks, as the test campaign fully covers typical mobile network conditions. The models, based on aggregated values, are partly validated with a terminal agent in a real LTE network. It is expected that by using not averaged values, or averaged over shorter periods, results would be better in the majority of cases. Live mobile networks, however, usually make available only one throughput value per video reproduction. Actually, current probes [24, 25] usually aggregate information obtaining only one value per session. Moreover, it should be emphasized that models developed in this work are not designed for a video server, where an end to end connection is available, but for network optimization based in QoE terms, where aggregated information in the most commonly available information. Operators demand working with not too complex processes and/or information. Furthermore, models using aggregated live information are able to be included into mobile network optimization tools, where the trade-off between accuracy and complexity is quite active, and, thus, models should remain as simple as possible without compromising accuracy.

The main contributions of this work are as follows: (a) a system testbed to perform automatic QoS and QoE measurements that can be used for any kind of mobile communications service and radio access technology and (b) three very accurate network-layer models to estimate YouTube S-KPIs precisely from TCP/IP metrics, meeting nowadays network operator demands. The rest of the paper is organized as follows. Section 2 describes the system testbed developed to automate the construction of the S-KPI estimation models. Then, Section 3 presents the video-streaming service performance model proposed in this work, which is derived with the testbed. Section 4 presents the results of the model in a real LTE network. Finally, Section 5 outlines the conclusions drawn from this work.

2 System testbed

In this section, a system testbed to automate the construction of QoE models in wireless networks is presented. Automation includes generating video sessions, shaping traffic to emulate different network conditions and measuring network and service performance. For clarity, the main components of the testbed are first described. Then, it is explained how traffic is generated, controlled, and captured in the testbed.

2.1 Testbed elements

In Fig. 1, a diagram representing the structure of the testbed is shown. The testbed comprises a terminal agent, a radio access network (e.g., Wi-Fi access point), an IP-level network emulator, and a network probe [24]. All these elements are described below.

2.1.1 Terminal agent

The mobile terminal is a *Samsung Galaxy S5 G-900F* smartphone with Android 4.4.2 Operating System (OS). The terminal is rooted to have full access to system resources. Then, a terminal agent is installed to mimic user interaction and collect end-user application-level measurements (i.e., S-KPIs) for YouTube. With this software, both video playing and measurement collection are executed in the background, reducing user interaction to the minimum. In this work, the terminal agent is V3D [18]. V3D solution combines a mobile application and a powerful server interacting together over-the-air. The mobile app monitors the user activity on the device, such as web browsing, using an application, watching a video, or making a call, and collects all relevant data related to the network performance and customer experience. After the session has finished, the data is transferred to the server for further computing, aggregation, and real-time display. With regard to YouTube, V3D app has an embedded video player that is connected to public YouTube API. Due to this, all the important events (e.g., the playback end of a video) are not exactly detected but inherent in the app since all the playback information is accessed through the API. Thus, the output of the terminal agent is a report with video S-KPI statistics per session. These service performance measurements are then used to validate the formulas to estimate YouTube S-KPIs from TCP/IP metrics.

2.1.2 Wireless network

The testbed includes a Wi-Fi Access Point (AP). A Wi-Fi AP acts as a *tunnel* between the terminal and the traffic control element. For this purpose, a standard Wi-Fi router is used. The Wi-Fi router model is *HG556a*, with *Wireless 802.11 b,g,n* and *Ethernet 802.3u* as interfaces, operating at 2.4 GHz [26]. The wireless router is wired to the computer performing the traffic control through an Ethernet cable. This connection allows traffic from the terminal to go through the network emulator.

2.1.3 Network emulator

The network emulator is the last element before the Internet. It consists of a computer with Linux OS including *NetEm* [27] software. *NetEm* is a network emulator that controls delays, losses, duplication, and reordering of IP packets [28]. It is included in the Linux kernel since version 2.6, as part of the *iproute2* package. It comprises a small kernel module for a queueing discipline and a command-line utility (*tc*) for configuration purposes. The latter can be automated using shell scripts for convenience.

In *NetEm*, packet delay and its variation (jitter) are described by the mean value and standard deviation, and, optionally, a correlation coefficient. By default, a uniform

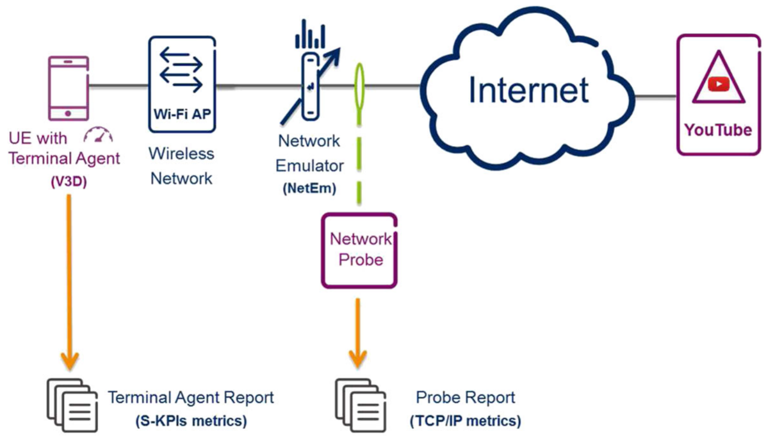
UE with Terminal Agent (V3D)
Wi-Fi AP
Wireless Network
Network Emulator (NetEm)
Internet
YouTube
Network Probe
Terminal Agent Report (S-KPIs metrics)
Probe Report (TCP/IP metrics)

Fig. 1 System testbed

jitter distribution is used, which can be substituted by other functions, such as Pareto, Pareto-normal, normal, or custom distributions created from experimental or simulation data [29].

The use of *NetEm* in order to create laboratory testbeds has been already done [30–32]. In this work, the network emulator is treated as a black box, used to see the impact that different conditions may have on network-layer indicators. Thus, the goal behind *NetEm* use is not to perfectly emulate each network feature, but generating network performance conditions as similar as those encountered in live networks.

The network emulator is installed on a personal computer with an Intel® 4-core ™ i3 CPU 540 at 3.07 GHz with 4 GB of RAM and Ubuntu 14.04 LTS 64-bit OS. This computer has two network cards for connecting the AP to the Internet (one is wired to the Wi-Fi AP via an Ethernet cable and the other one is connected to a local area network with access to the Internet).

2.1.4 Network probe

In the previous computer, *tcpdump* is used to capture messages exchanged at different levels of the protocol stack. *Tcpdump* is an open source command-line tool for sniffing network traffic [33]. Its output is a capture file in *libcap* format file (*.pcap* extension), supported by most packet analyzing tools. Then, a Probe software provided by a mobile vendor is used to derive relevant TCP/IP metrics, from which S-KPIs are estimated.

2.2 Testbed processes

The main processes executed for collecting measurements with the testbed are explained in the following paragraphs.

2.2.1 Traffic generation

Video-streaming traffic is generated by emulating user interactions with a terminal agent installed in the smartphone. In this work, the terminal agent is V3D [18]. An associated web tool allows the creation of automatic tests, reducing human intervention to a minimum. Automation starts by defining the URL of YouTube videos selected to obtain S-KPI measurements together with the time gap between videos, i.e., the minimum amount of time between the end of one video and the beginning of the next one. The time gap chosen must be long enough to avoid that measurements of two consecutive videos affect each other. It should be pointed out that time gap is only included for model construction purposes, making easier the process of extracting statistics and building the models. Once the models are created, they can be applied to any single video or playlist without any time gap between videos. In addition, the time scheduled during which the terminal agent is taking background measurements must be also set up. Thus, the time period selected must be aligned to the other testbed processes explained hereafter.

2.3 Traffic control

NetEm is used to model different network conditions. The term “network” includes access (i.e., radio, transmission and core) and transport (i.e., Internet) segments. The command *tc* is used to configure *NetEm* parameters, with the following syntax:

$$\begin{array}{l} \textit{tc qdisc ... dev DEVICE add/change netem OPTIONS} \\ \textit{OPTIONS} := [\textit{LIMIT}]\ [\textit{DELAY}]\ [\textit{LOSS}]\ [\textit{RATE}] \\ \textit{LIMIT} := \textit{limit packets} \\ \textit{DELAY} := \textit{delay TIME}\ [\textit{JITTER}\ [\textit{CORRELATION}]] \\ \textit{LOSS} := \textit{loss PERCENT} \\ \textit{RATE} := \textit{rate RATE} \end{array} , \tag{1}$$

where *qdisc* is the queue associated to interface *DEVICE* through which packets are sent. Regarding *OPTIONS* parameters, *LIMIT* limits the effect of other selected options to the indicated number of next packets, *DELAY* adds the chosen average delay in milliseconds to the packets outgoing to chosen network interface, *JITTER* is used to quantify delay variation (uniformly distributed by default) also in milliseconds, *CORRELATION* is a percentage controlling how much the current delay value depends on the previous one, *LOSS* is the packet loss probability (as a percentage) and *RATE* limits the throughput to the specified value in kilobytes per second (i.e., throttling) by a token bucket filter [28]. In this work, only delay, packet loss, and throttling features in the downlink (i.e., server-to-client link) are modified by the network emulator for simplicity.

2.4 Traffic capturing

Traffic capturing is a key part of the whole process, since TCP/IP metrics, required for the construction of S-KPI models, are obtained from .pcap files. In this work, *tcpdump* is used for this purpose. This tool works by capturing and displaying a description of packets on a network interface that match certain criteria. Criteria comprise boolean search operators, host names, IP addresses, network names, and protocols. In the testbed, traffic is captured both at the terminal and the connection to the Internet only for validation purposes. .pcap files captured at the terminal are uploaded to the computer where *NetEm* is installed and automatically deleted from the terminal to avoid draining its storage capacity.

3 Video-streaming service performance model

In this section, an analytical model to estimate S-KPIs for an encrypted video-streaming service from TCP/IP metrics collected on a per-connection basis is proposed. For clarity, the relevant S-KPIs for video-streaming service are first defined. Then, the methodology used to identify the main TCP/IP metrics affecting video performance is described. Finally, the formulas relating S-KPIs and relevant metrics are presented for YouTube traffic.

3.1 S-KPI definition

Video-streaming experience is mainly affected by two issues: delay before the video starts and image freezes due to buffer underruns at the client (known as re-buffering or stalling events). As in [14], three S-KPIs are selected here to cover these issues: *Initial Buffering Time*, *Re-buffering Ratio*, and *Re-buffering Frequency*.

- *Initial Buffering Time*
  From the user point of view, it is defined as the time elapsed from the moment the user clicks the play button until the first video image appears on the screen (in seconds).
- *Re-buffering Ratio*
  It is defined as the total re-buffering time (i.e., the total time the video is frozen during its reproduction) divided by the time since the video starts until the video ends, computed as

$$\text{Re-buffering Ratio} = \frac{\text{Re-buffering Time [s]}}{\text{Re-buffering Time [s]} + \text{Video Duration [s]}} \quad [\%]\,. \tag{2}$$

- *Re-buffering Frequency*
  It is the number of re-buffering events per minute of played video (in events/min), computed as

$$\text{Re-buffering Frequency} = \frac{\#\ \text{Re-buffering Events}}{\text{Reproduction Time [min]}} \quad [1/\text{min}]\,. \tag{3}$$

3.2 Experimental methodology

A test campaign is carried out to derive a simple and robust model for YouTube S-KPIs in wireless networks, and, more specifically, to identify which metrics show a higher impact on video performance. It should be noted that test campaigns are configured and launched from a web portal to the specific terminal where the terminal agent app is installed. In fact, both video playing and measurement collection are executed in the background, reducing user interaction to the minimum. Thus, all measurements are collected with a static terminal agent downloading video sequences through a Wi-Fi radio link.

The selected sequences were the four most visited videos of YouTube history until 2015 [34–37]. All videos are tested with the same fixed resolution (640×360 pixels), as the terminal agent software does not support Dynamic Adaptive Streaming over HTTP (DASH) [38, 39]. However, since resolution is fixed at an intermediate value (not very low or very high), a wide range of S-KPI performance results can be found when network conditions are changed. Other video features as duration and bitrate have been also summarized for each video tested in Table 1.

To check the impact of network conditions on S-KPIs, *NetEm* parameters are configured in the following ranges,

Table 1 YouTube videos tested

Name	Resolution	Duration [s]	VBR [kbps]
PSY - GANGNAM STYLE M/V [34]	640 × 360	253	757.3
Taylor Swift - Blank Space [35]		273	417.5
Justin Bieber - Baby ft. Ludacris [36]		225	653.2
Katy Perry - Dark Horse (Official) ft. Juicy J. [37]		226	723.3

according to values experienced by users in a live LTE downlink:

- Packet loss ratio [%]: 0, 0.75, 1.5, and 3.
- Packet delay [ms]: 0, 50, 100, 200, and 400.
- Maximum throughput limit [kbps]: 250, 500, 1000, 2000, and 4000.

All the combinations of these values are considered in the sensitivity analysis. Each combination is applied for 40 min, during which between six and nine videos are downloaded, depending on network conditions. As a result, more than 700 video measurement samples are generated, each correspond to a video sequence and a noise configuration.

The previous *NetEm* parameter sweep is used to detect TCP/IP metrics with a higher impact on the above described video S-KPIs. For this purpose, .pcap files obtained with *tcpdump* are processed off-line by the Probe software. A set of candidate predictors must be selected in order to find which one(s) of them are relevant for the video S-KPI model. Two main requirements for the selection of candidates can be stated as follows: (a) feasibility, i.e., any candidate predictor must be able to be measured by the Probe and testbed platform, and (b) significance, i.e., the set of predictors must reflect any (or the majority of) possible effects in the network and should have a significant impact on the selected video S-KPI. Three TCP/IP metrics are selected as candidate predictors, namely *average downlink throughput* (THRU), *overall downlink packet loss ratio* (PLR), and *average downlink round trip time* (RTT), all of them measured at the IP level. These three predictors provide meaningful information about the available bandwidth [40] and, furthermore, are closely related to network indicators. Although TCP/IP protocols possess other parameters providing additional information, the three predictors selected have been already validated as a reliable source of information [41, 42].

Once S-KPI models are built, a second campaign is launched in order to assess the robustness of the models in a new set of network situations. In this second campaign, there is no throughput limitation and only packet loss and packet delay were modified using *NetEm*. Moreover, the packet loss and delay ranges are very wide, covering all performance cases in a real mobile network. These ranges are:

- Packet loss [%]: 0–19 (1% step).
- Packet delay [ms]: 0–900 (100 ms step).

As in the previous campaign, every possible combination of packet loss and packet delay is considered (e.g., 2% and 400 ms).

3.3 Identification of relevant variables

A preliminary sensitivity analysis is carried out to come up with a model including only significant predictors. *NetEm* parameters are swept and THRU, PLR, and RTT metrics are collected from the Probe. Figure 2a shows how THRU varies with the PLR, both measured by the Probe, when the packet delay parameter in *NetEm* is fixed at 100 ms. Likewise, Fig. 2b shows the THRU and RTT, both measured by the Probe, when the packet loss ratio parameter in *NetEm* is fixed to 1.5%. In both subfigures, it is observed that, when either PLR or RTT move around low values, the limiting factor is the bandwidth throttling feature of *NetEm* that sets an upper bound to THRU. That maximum value is not always reached since it is possible that the throughput requested by a video user is below that bound. In contrast, as PLR and RTT increase, these tend to limit the experienced user throughput. Trends in Fig. 2a, b also support the idea that the three TCP/IP metrics (i.e., THRU, PLR and RTT) are highly correlated, and, therefore, using all of them would increase model complexity without increasing estimation accuracy. To confirm this statement, a thorough multivariate regression analysis was performed and several estimation models combining these three metrics were tested, being the most accurate, a linear model including, from the three candidates, only THRU metrics. From these results, it can be concluded that THRU provides enough information about TCP/IP performance from those selected from the Probe, as any change in the other two TCP/IP metrics (PLR and RTT) is reflected immediately in THRU measurements reported by the Probe.

At this point, it could be stated that a S-KPI estimation model can be constructed from the representative THRU network indicator. Note, however, that THRU has been selected from parameters only reflecting network performance and none parameter including any video feature has been studied. It is expected that video requirements (bandwidth, typically) also influences on S-KPIs additionally to network conditions (i.e., different videos would result in different S-KPI values for the same THRU value). Thus, a second predictor variable is required to be included in a complete S-KPI model, describing the bandwidth requirements of the specific video downloaded by the user. As previously stated, all YouTube videos are tested with the same fixed resolution (640×360). Nonetheless, each of the four videos has its own bandwidth requirements, depending on the coding scheme and video content. After checking all the information reported by the Probe and the terminal agent, the *average video bitrate* (VBR) (reported by the terminal agent) arises as the best choice since it is a key indicator directly related to video resolution and, therefore, quality experienced by the video consumer. In the four selected videos, VBR ranges

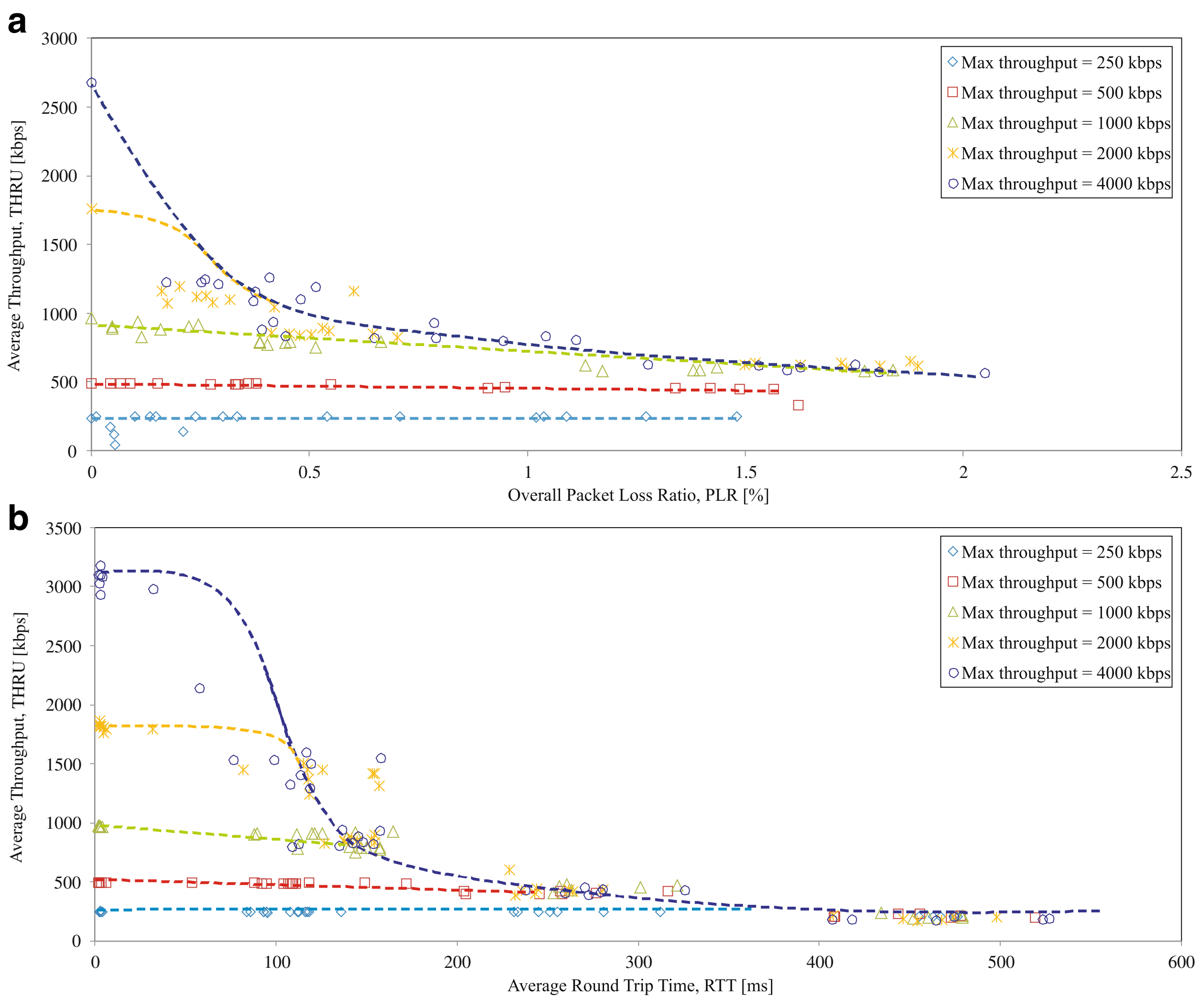

Fig. 2 Sensitivity analysis. **a** *NetEm* packet delay 100 ms. **b** *NetEm* packet loss ratio 1.5%

from ∼ 400 to ∼ 800 kbps. Note that, in spite of the name, VBR is the sum of video and audio data rate.

3.4 S-KPI estimation models

In this section, three formulas are derived to estimate the most relevant S-KPIs for YouTube from THRU measurements obtained by the Probe and VBR video bandwidth. Once predictors are known, a model structure is selected. Taking into account the trade-off between complexity and accuracy, always inherent in network optimization tools for which models proposed in this work are designed, a linear regression model structure with a single predictor variable is chosen. Not shown in the text, it has been confirmed that more complex non-linear models (e.g., quadratic, piecewise, ...), does not supply additional accuracy. Thus, a composite predictor for the lineal model is defined as the ratio of the average IP-level throughput (THRU) and average video bitrate (VBR), VBR/THRU (i.e., demanded vs experienced bandwidth). This predictor was already presented in [8], where it is demonstrated that QoE highly depends on downlink-encoding bottlenecks. The resulting linear models only have two constants (i.e., intercept and slope) and are therefore easy to calibrate. In spite of their simplicity, it is shown that the models provide accurate estimates of S-KPIs. Likewise, the proposed models, constructed with 700 measurement samples, show good results when applied to other sets of samples (i.e., second campaign previously described). In fact, models provide very accurate results for the three video S-KPI ($R^2 > 0.65$ in every case). The robustness of the models is also assessed since performance is already good when no calibration is involved and there is not great change when calibration (i.e., constants of the linear models are recalculated) appears.

It should be pointed out that VBR is known in the testbed as it is provided by the terminal agent. However, this is not the case in the real network due to YouTube traffic encryption, so VBR would have to be estimated by other means, such as (a) most common VBR values from non-encrypted videos, (b) VBR estimation from probe live measurements calculated as video size divided by video duration, or (c) VBR estimation from probe measurements based on YouTube traffic ON-OFF behavior [4].

3.4.1 Initial Buffering Time

From a regression analysis, it is deduced that the *Initial Buffering Time* of a connection *c* (i.e., the time the user must wait until the video starts) can be estimated by the formula

$$\text{Initial }\widehat{\text{Buffering}}\text{ Time}\,[c] = 5.91 \cdot \frac{VBR\,[c]}{THRU\,[c]} + 1.43 \qquad [\text{s}]\,. \tag{4}$$

Figure 3 shows a scatter plot to check the correlation between the initial buffering time measured by the terminal agent and that estimated with TCP/IP metrics from the Probe using (4). Each point corresponds to one video reproduction. It is observed that estimation accuracy is reasonably good, with all the samples around the unit line (representing a perfect estimation). Those samples more distant from the unit line can be explained by the fact that all the estimation models are based on average values. In this case, samples above the unit line (i.e., estimate larger than measurement) are related to videos where the download throughput at the beginning is evidently higher than the average throughput through the whole download, causing that the initial buffering time is less than expected (i.e., overestimated). On the other hand, samples below the unit line correspond to the opposite situation, where download throughput at the beginning is much lower than on average. Nonetheless, in general, results are promising, since the regression equation is reasonably close to the identity (i.e., $y = 0.8309x + 1.6967$), the determination coefficient is large (i.e., $R^2 = 0.81$), and the 80th percentile error is just 3.15 s. Note that a great range of values is covered in the figure (from 0 to 50 s), as a result of the different packet loss, delay, and bandwidth throttling parameters in *NetEm*.

3.4.2 Re-buffering Ratio

A regression analysis shows that the *Re-buffering Ratio* of a connection *c*, defined as the percentage of time the video is frozen, can be estimated as

$$\text{Re-}\widehat{\text{buffering}}\text{ Ratio}\,[c] = \max\left(0, -91.5 \cdot \frac{THRU\,[c]}{VBR\,[c]} + 96.67\right) \qquad [\%]\,. \tag{5}$$

Figure 4 compares the re-buffering ratio measured by the terminal agent and that estimated with throughput measurements from the Probe using (5). In the figure, it is observed that estimation in this case is very accurate for most of the samples. The determination coefficient is close to 1 ($R^2 = 0.98$), and the 80th percentile error is very low (2.15%) compared to the whole percentage range.

For completeness, a detailed analysis is carried out to explain those outliers where the terminal agent measures a non-negligible re-buffering ratio (up to 40%), but the model estimates that no re-buffering should occur. In all these cases, the average throughput is much greater than the video bitrate and hence the absence of re-buffering events estimated by the model. However, at some point of time, an isolated packet loss may eventually cause a temporary decrease in THRU, or some larger video frames (e.g., intra-frame coded frames) may cause a sudden increase in the instantaneous VBR, draining out the buffer and staying without anything to reproduce for a

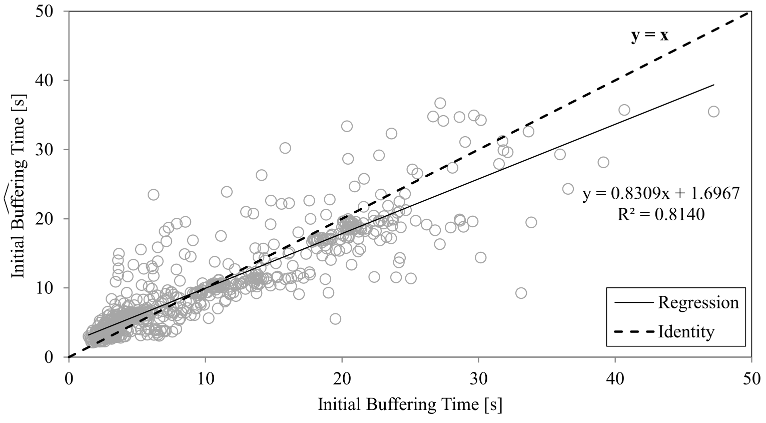

Fig. 3 Comparison of initial buffering time measured by the terminal agent and estimated by the model

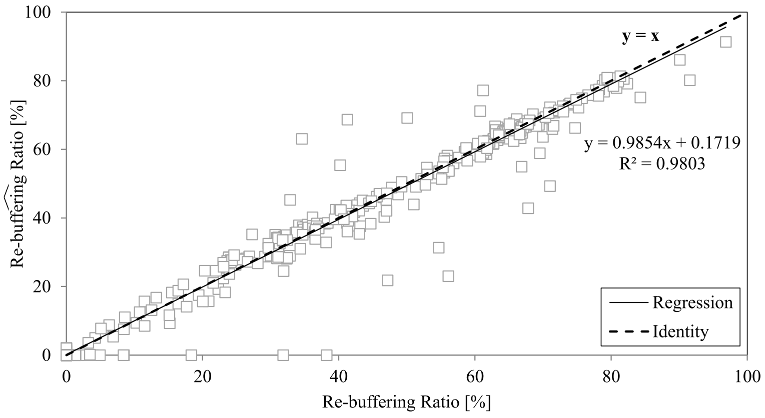

Fig. 4 Comparison of re-buffering ratio measured by the terminal agent and estimated by the model

significant period of time. Note that the small-time granularity required in measurements to reflect these situations is lost when average values for the whole connection are considered, although simplicity in model construction is gained, which is highly appreciated by network operators. Models' integration in network optimization tools and the use of live data is also easier with long-time (e.g., hourly) averages.

3.4.3 Re-buffering Frequency

A regression analysis shows that the *Re-buffering Frequency* of a connection *c*, defined as the number of video freezes per minute, can be estimated as

$$\widehat{\text{Re-buffering Frequency}}\,[c] = \max\left(0, -7.75 \cdot \frac{THRU\,[c]}{VBR\,[c]} + 8.37\right) \quad [1/\text{min}]\,. \tag{6}$$

Figure 5 compares the re-buffering frequency measured with the terminal agent and that estimated from throughput metrics in the Probe with (6). Once again, estimates are very accurate, with the vast majority of samples around the identity line. Specifically, $R^2 = 0.95$ and the 80th percentile error is 0.365 [1/min].

As in previous S-KPIs, there are some samples located on the horizontal axis, i.e., videos for which the model predicts zero freezes, but the terminal agent measures several freezes. The explanation is exactly the same as in the previous case, directly related to sudden VBR peaks or a temporary decrease in THRU. Moreover, there are a few samples well above the unit line. A closer analysis shows that, in all these cases, the average throughput is lower than the video bitrate, and because of that, the estimated number of re-buffering events per minute is large. The mismatch between estimates and measurements from the terminal agent can be due to the buffer strategy, which is specific of each player. In those situations where the buffer content is fluctuating around the level needed for the player to start video reproduction again, the player may decide to delay the reproduction until the buffer is re-filled again. In these situations, the TCP/IP model might estimate more than one re-buffering event when actually only one *longer* re-buffering event has happened. Since these cases are represented by a few outliers, the robustness and accuracy of the model proposed are not compromised.

The application of the previous three S-KPI models (i.e., (4)-(6)) in a second test campaign, described in Section 3.4 is assessed. Models show good results in this new set of network conditions. More specifically, initial buffering time reaches a $R^2 = 0.635$ (0.743 if model constants are recalibrated), re-buffering ratio model shows a very good behavior with $R^2 = 0.98$ (and no need of constant calibrations), and, finally, re-buffering frequency shows $R^2 = 0.653$ (0.808 if constants are recalibrated). These values demonstrate model robustness, since S-KPI models can be easily adapted, or they are even still valid, for two very different network scenarios.

4 Model assessment in a real network

In this section, the above described service performance models are tested in a live LTE network. The trial setup is explained first and results are presented later.

4.1 Trial setup

In the trial, a smartphone with a terminal agent is located in a fixed position within a cell of the LTE system. Due to operator constraints, a different terminal agent is used (Nemo [21]). As in the V3D case, Nemo is an application provisioned and installed over-the-air on commercial smartphones, and it calculates KPIs for professional customer experience monitoring of wireless network performance and services from the end-user point of view. With Nemo, operators can easily and discreetly collect QoE data for mobile customer experience management directly from their customers while they are using their

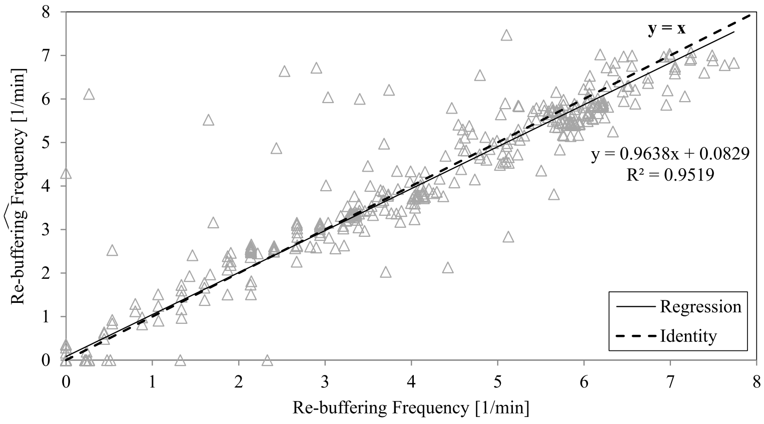

Fig. 5 Comparison of re-buffering frequency measured by the terminal agent and estimated by the model

smartphones. The server side of the solution aggregates KPIs transferred via HTTP from one or a million smartphones simultaneously. Note that the definition of S-KPIs may differ between terminal agents, as not all of them consider the same components (e.g., the decoding delay, the presentation delay, ...). Likewise, the live network may add new elements and processes that might have an impact on the model. Thus, even if the methodology to construct the model is still the same (i.e., regression analysis), the model has to be calibrated, resulting in different constants in the formulas. Unlike lab tests, no network emulator is needed in the field trial, since data transmission is carried out over the real network. As the terminal is static, throughput fluctuations are only due to changes in interference and cell load conditions (mainly in the downlink). In this campaign, the same video [43] is tested continuously during three consecutive days (from 11.30 AM of day 1 until 4 PM of day 3). Thus, 365 video measurement samples (one per video reproduction) are available for comparison. Due to a low network load and a good terminal position, link performance was extremely good throughout the tests, and no re-buffering events were observed in any of the YouTube video playbacks. Thus, the analysis is hereafter restricted to the initial buffering time. Nonetheless, it is worth noting that the models proposed for re-buffering S-KPIs (i.e., re-buffering ratio and re-buffering frequency) were capable of estimating the proper value (i.e., 0) in every tested case.

4.2 Trial results

The experimental model for the initial buffering time resulting from the regression analysis in the live network is

$$\text{Initial } \widehat{\text{Buffering}} \text{ Time}\,[c] = 5.91 \cdot \frac{VBR\,[c]}{THRU\,[c]} \quad [\text{s}]\,. \quad (7)$$

When comparing it with the model obtained in the lab, shown in (4), it is observed that the slope is exactly the same (i.e., 5.91), and the only difference is the intercept point, which is now 0 (instead of 1.43), due to the fact that Nemo does not follow the same measurement strategy as V3D, i.e., video S-KPI definition is not the same for those two terminal agents. That minor change is a clear indication of the robustness of the model. Figure 6 compares the initial buffering time measured by the terminal agent and that estimated with the model. Results are promising, with all the samples very close to the identity line. Likewise, the regression equation is reasonably close to the identity, the determination coefficient is large, and the 80th percentile error is 0.255 s. All these results confirm the great accuracy and robustness of the proposed model.

5 Conclusions

In this paper, a methodology for deriving models to estimate key service performance indicators from TCP/IP metrics for video-streaming in a wireless network has been presented. In the proposed testbed, a terminal agent (V3D) is used to automate user interaction with the video service provider (YouTube). In parallel, a network emulator (*NetEm*) is used to modify network conditions in terms of packet delay, packet loss ratio and maximum throughput in the downlink. At the same time, all the traffic generated by the terminal agent is captured with a network traffic analyzer (tcpdump) and processed by Probe software to obtain TCP/IP metrics. Then, the models relating TCP/IP metrics with service performance indicators are obtained by simple regression analysis. The models proposed have all the features desired by network operators: simplicity, robustness, usability, and accuracy. The variable defined as VBR/THRU (i.e., demanded vs experienced bandwidth), was already presented in [8] were its dependency with QoE was proved and has been endorse in this work. Results in the lab with a Wi-Fi network have

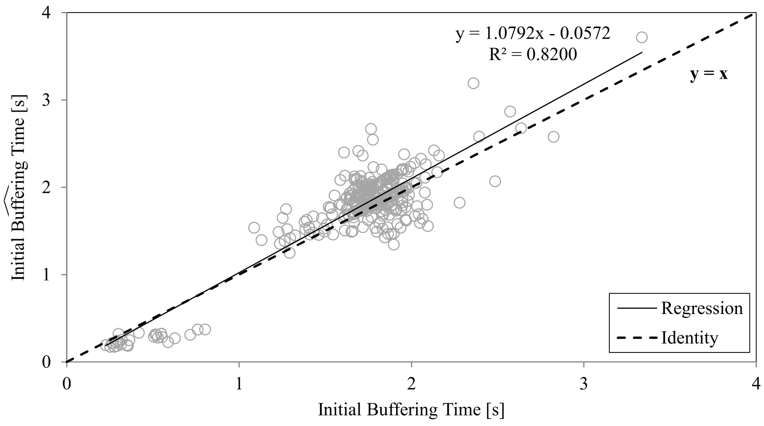

Fig. 6 Comparison of initial buffering time measured and estimated by the model in a real network

shown that service performance estimates obtained with the models are very close to measurements obtained with the terminal agent, reaching a determination coefficient of 0.98 in some cases. The methodology has been partly validated in a real network, where it has been checked that the models obtained in the lab are still valid for a live network with minor changes. The proposed methodology could be applied to any radio access technology and service, provided that models are calibrated with measurements from the live network. Based on the field trial results, it is also expected that the formulas presented here could be applied to other network operators with few modifications. It is left for future work further testing models in real networks to assess the S-KPI models under any circumstances, including the derivation of the formulas for adaptive video-streaming services (e.g., DASH), already supported by many content providers. Furthermore, next tests are intended to use different videos and terminals in order to widen the validity range of the models proposed.

Acknowledgements
This work has been funded by the Spanish Ministry of Economy and Competitiveness (TEC2015-69982-R) and Ericsson.

Authors' contributions
POB and MT participated in all the stages of this work, including manuscript drafting. Additionally, RGT collaborated on models' design and its application over the testbed, and SLR collaborated on the analysis of results and redesign of experiments, as well as manuscript drafting. All authors have read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]Ingeniería de Comunicaciones, Universidad de Málaga, Campus de Teatinos S/N, 29071 Málaga, Spain. [2]Ericsson, Severo Ochoa 51, 29590 Málaga, Spain.

Received: 13 June 2017 Accepted: 26 November 2017
Published online: 15 December 2017

References

1. Ericsson, Mobility report. Technical Report (2017). https://www.ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-june-2017.pdf
2. NGMN Alliance, NGMN 5G white paper. White paper (2015)
3. E Liotou, H Elshaer, R Schatz, R Irmer, M Dohler, N Passas, L Merakos, in *7th International Workshop on Quality of Multimedia Experience (IEEE QoMEX)*. Shaping QoE in the 5G ecosystem, (Costa Navarino, 2015), pp. 1–6. doi:10.1109/QoMEX.2015.7148089, https://www.researchgate.net/publication/279037429_Shaping_QoE_in_the_5G_ecosystem
4. P Ameigeiras, JJ Ramos-Munoz, J Navarro-Ortiz, JM Lopez-Soler, Analysis and modelling of YouTube traffic. Trans. Emerging Telecommun. Technol. **23**(4), 360–377 (2012). http://onlinelibrary.wiley.com/doi/10.1002/ett.2546/full
5. JJ Ramos-Muñoz, J Prados-Garzon, P Ameigeiras, J Navarro-Ortiz, JM López-Soler, Characteristics of mobile YouTube traffic. IEEE Wireless Commun. **21**(1), 18–25 (2014)
6. F Wamser, P Casas, M Seufert, C Moldovan, P Tran-Gia, T Hossfeld, Modeling the Youtube stack: From packets to quality of experience. Comput. Netw. **109**, 211–224 (2016)
7. International Telecommunication Union, Subjective video quality assessment methods for multimedia applications. ITU-T Recommendation P.910 (2008). https://www.itu.int/rec/T-REC-P.910-200804-I/en
8. P Casas, A Sackl, S Egger, R Schatz, in *IEEE Globecom Workshops (GC Wkshps)*. Youtube & Facebook Quality of Experience in Mobile Broadband Networks (IEEE, Anaheim, 2012), pp. 1269–1274. http://ieeexplore.ieee.org/document/6477764/
9. M Eckert, TM Knoll, F Schlegel, in *IEEE 7th International Conference on Signal Processing and Communication Systems (ICSPCS)*. Advanced MOS calculation for network based QoE Estimation of TCP streamed video services (IEEE, Carrara, 2013), pp. 1–9. http://ieeexplore.ieee.org/document/6723923/
10. T Hoßfeld, R Schatz, E Biersack, L Plissonneau, *Internet Video Delivery in YouTube: From Traffic Measurements to Quality of Experience*. (Springer, Berlin, Heidelberg, 2013), pp. 264–301
11. G Gómez, J Lorca, R García, Q Pérez, Towards a QoE-Driven Resource Control in LTE and LTE-A Networks. J. Comput. Netw. Commun. **2013**, 15 (2013). doi:10.1155/2013/505910
12. F Lozano, G Gómez, M-C Aguayo-Torres, C Cárdenas, A Plaza, A Garrido, J Baños-Polglase, J Poncela, Network Performance Testing System Integrating Models for Automatic Qoe Evaluation of Popular Services: Youtube and Facebook. Wireless Pers. Commun. **81**(4), 1377–1397 (2015)
13. P Oliver-Balsalobre, M Toril, S Luna-Ramírez, JM Ruiz Avilés, Self-tuning of scheduling parameters for balancing the quality of experience among

services in LTE. EURASIP J. Wireless Commun. Netw. **2016:7**(1), 12 (2016). doi:10.1186/s13638-015-0508-x
14. RK Mok, EW Chan, RK Chang, in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*. Measuring the Quality of Experience of HTTP Video Streaming (IEEE, Dublin, 2011), pp. 485–492. https://www.researchgate.net/publication/221293512_Measuring_the_Quality_of_Experience_of_HTTP_Video_Streaming
15. Y Douga, M Bourenane, A Mellouk, Y Hadjadj-Aoul, TCP based-user control for adaptive video streaming. Multimedia Tools Appl. **75**(18), 11347–11366 (2016)
16. T Porter, X-H Peng, An objective approach to measuring video playback quality in lossy networks using TCP. IEEE Commun. Lett. **15**(1), 76–78 (2011)
17. W Pan, G Cheng, H Wu, Y Tang, in *IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. Towards QoE assessment of encrypted YouTube adaptive video streaming in mobile networks (IEEE, Beijing, 2016), pp. 1–6. http://ieeexplore.ieee.org/document/7590437/
18. V3D. http://www.v3d.fr/. Accessed on October 2017
19. NPT. https://play.google.com/store/apps/details?id=com.ericsson.mbbmeasurement&hl=en./ Accessed on October 2017
20. XCAL. http://www.accuver.com/home/sub.php?localNum=1&pageNum=0&subNum=3&subNum2=0. Accessed on October 2017
21. Nemo. http://www.keysight.com/en/pd-2767489/nemo-customer-experience-monitor?nid=-32104.1200414&cc=ES&lc=eng. Accessed on October 2017
22. S Göring, A Raake, B Feiten, in *IEEE Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. A Framework for QoE Analysis of Encrypted Video Streams (IEEE, Erfurt, 2017), pp. 1–3. http://ieeexplore.ieee.org/document/7965640/
23. I Orsolic, D Pevec, M Suznjevic, L Skorin-Kapov, in *IEEE Globecom Workshops (GC Wkshps)*. Youtube QoE Estimation Based on the Analysis of Encrypted Network Traffic Using Machine Learning (IEEE, Washington, 2016), pp. 1–6. http://ieeexplore.ieee.org/document/7849088/
24. Ericsson Expert Analytics. https://www.ericsson.com/ourportfolio/it-and-cloud-products/expert-analytics. Accessed on October 2017
25. Osix Monitoring Polystar. https://www.polystar.com/products/osix-monitoring/. Accessed on October 2017
26. Router HG556a - User Guide. https://www.vodafone.co.nz/cms/documents/broadband-complete-manual.pdf. Accessed on October 2017
27. NetEm. https://wiki.linuxfoundation.org/networking/netem. Accessed on October 2017
28. S Hemminger, *et al*, in *6th Australia's National Linux Conference*. Network emulation with NetEm (Linux Conference, Australia, 2005), pp. 18–23. https://www.rationali.st/blog/files/20151126-jittertrap/netem-shemminger.pdf
29. A Jurgelionis, J-P Laulajainen, M Hirvonen, AI Wang, in *IEEE Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*. An Empirical Study of NetEm Network Emulation Functionalities (IEEE, Maui, 2011), pp. 1–6. http://ieeexplore.ieee.org/document/6005933/
30. R Gass, J Scott, C Diot, in *IEEE 7th Workshop on Mobile Computing Systems and Applications(WMCSA)*. Measurements of in-motion 802.11 networking (IEEE, Orcas Island, 2005), pp. 69–74. http://ieeexplore.ieee.org/document/1691716/
31. E Ibarrola, I Taboada, R Ortega, *et al*, in *IEEE Fifth International Conference on Autonomic and Autonomous Systems (ICAS)*. Web qoe evaluation in multi-agent networks: Validation of itu-t g. 1030 (IEEE, Valencia, 2009), pp. 289–294. http://ieeexplore.ieee.org/document/4976618/
32. DH Bui, K Lee, S Oh, I Shin, H Shin, H Woo, D Ban, in *IEEE 34th Real-Time Systems Symposium (RTSS)*. Greenbag: Energy-efficient bandwidth aggregation for real-time streaming in heterogeneous mobile wireless networks (IEEE, Vancouver, 2013), pp. 57–67. http://ieeexplore.ieee.org/document/6728861/
33. Tcpdump. http://www.tcpdump.org/. Accessed on October 2017
34. PSY - GANGNAM STYLE M/V. https://www.youtube.com/watch?v=9bZkp7q19f0. Accessed on October 2017
35. Taylor Swift - Blank Space. https://www.youtube.com/watch?v=e-ORhEE9VVg. Accessed on October 2017
36. Justin Bieber - Baby ft. Ludacris. https://www.youtube.com/watch?v=kffacxfA7G4. Accessed on October 2017
37. Katy Perry - Dark Horse (Official) ft. Juicy J. https://www.youtube.com/watch?v=0KSOMA3QBU0. Accessed on October 2017
38. M Michalos, S Kessanidis, S Nalmpantis, Dynamic Adaptive Streaming over HTTP. J. Eng. Sci. Technol. Rev. **5**(2), 30–34 (2012)
39. X Yin, A Jindal, V Sekar, B Sinopoli, A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. ACM SIGCOMM Comput. Commun. Rev. **45**(4), 325–338 (2015)
40. J Padhye, V Firoiu, DF Towsley, JF Kurose, Modeling tcp reno performance: a simple model and its empirical validation. IEEE/ACM Trans. Netw. (ToN). **8**(2), 133–145 (2000)
41. B Wang, J Kurose, P Shenoy, D Towsley, A model for tcp-based video streaming. University of Massachusetts Technical Report (2003). http://lass.cs.umass.edu/papers/pdf/TR03-TCP.pdf
42. SMH Shah, A Rehmanur, AN Khan, MA Shah, in *IEEE International Conference on Emerging Technologies (ICET)*. Tcp throughput estimation: A new neural networks model (IEEE, Islamabad, 2007), pp. 94–98. http://ieeexplore.ieee.org/document/4516323/
43. Ericsson Networked Society Film. https://www.youtube.com/watch?v=rZvDqumaLIY. Accessed on October 2017