Stimuli Generation for Quality-of-Experience Evaluation of Mobile Applications

https://doi.org/10.3991/ijim.v16i06.28691

Lars Wischhof[1(✉)], Jan Andreas Krahl[1], Robert Müllner[2]
[1]Munich University of Applied Sciences HM, Munich, Germany
[2]Telefónica Germany GmbH & Co. OHG, Munich, Germany
wischhof@ieee.org

Abstract—The assessment of Quality-of-Experience (QoE) corresponding to a given set of Quality-of-Service (QoS) parameters is of high importance for any mobile network operator. Methods for subjective QoE assessment such as MUlti Stimulus test with Hidden Reference and Anchor (MUSHRA) or Subjective Assessment Methodology for Video Quality (SAMVIQ) are based on collecting user feedback for one or several stimuli, i.e. differently processed versions of a source. Generating these stimuli corresponding to a given set of QoS parameters for a QoE assessment of a mobile application is a time-consuming, non-trivial task. Therefore, in this paper we propose a novel publicly available open-source framework for stimuli generation for QoE assessment of mobile applications. It is based on the combination of network emulation and automatic user-interface control of a real mobile device, whose behavior is recorded and post-processed to generate suitable stimuli. The article presents the basic concept of the framework, describes its open-source implementation, and concludes with an initial evaluation of the framework based on typical types of mobile applications.

Keywords—quality of experience, QoE, mobile device, mobile applications

1 Introduction

For any mobile network operator, it is of paramount importance that the customer experience meets or outperforms the quality expectations of its services. In general, this degree of user delight of the perceived quality of a service, termed Quality-of-Experience (QoE) [1], depends on the technical performance of the mobile network, referred to as Quality-of-Service (QoS). Typical QoS parameters are the achieved downlink data rate, communication delay, packet loss rate, etc. These parameters can be measured in mobile networks e.g. by performance counters or drive-tests, whereas QoE is a subjective measure with a non-linear relation to QoS. By providing more resources or adapting the assignment of the available resources to individual users and their applications, for example by using a modified scheduling [2], a mobile network operator can control the QoS and the resulting QoE observed by its customers.

While usually a higher QoS will lead to a higher QoE, the mapping of QoS to QoE depends on the specific mobile application and is not trivial. A typical shape for the QoS to QoE mapping is illustrated in Figure 1: In the first zone, the QoS is high enough to achieve the best possible QoE—therefore, increasing the QoS level does not lead to a higher user experience. In the second zone, the QoE is limited by the QoS and therefore decreases with increasing QoS degradation. Small improvements in QoS can lead to a relatively large improvement in QoE—therefore, this zone is of particular relevance when trying to increase QoE with limited resources. In the third zone, the QoE has decreased so much that the service becomes unusable. If the system cannot provide a better QoS so that one of the other zones can be reached, no resources should be assigned at all.

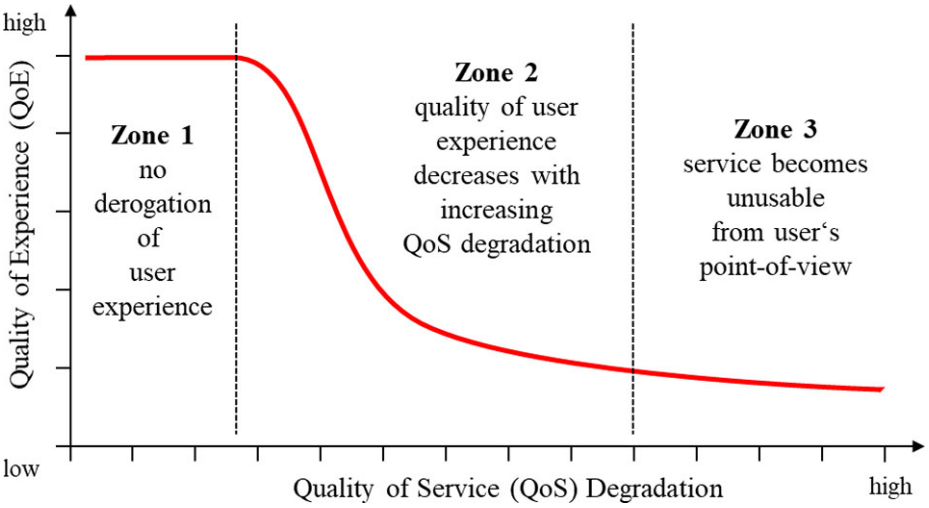

Fig. 1. Illustration of the typical zones of a QoS to QoE mapping (based on [3])

While the basic shape of this QoS to QoE mapping is valid for most services and has been confirmed in experiments for several types of mobile applications, such as VoIP [4], finding the correct mapping for a specific system is a challenging research task since the QoE associated with a QoS can depend on several QoS parameters as well as the specific application type. For mobile applications, numerous QoE classification schemes and QoE tools have been developed (a systematic overview is presented, e.g. in [5]).

In general, a QoE assessment is based on subjective tests [6], where users give feedback to different versions of a source, e.g. an encoded video signal in a specific presentation setting. In the following, the term *stimulus* will be used to describe a specific version that is assessed by a subject on a metric such as the Mean Opinion Score (MOS). Since subjective assessment methods rely on human interaction and are cost-intensive, many objective approaches have been proposed which estimate the QoE based on objective, automatically performed measurements. However, usually these need to address subjective metrics in order to validate the respective QoE analysis methods [5] and objective metrics. Thus, an initial subjective assessment based on suitable stimuli is still required.

1.1 Motivation: generation of stimuli for assessment of mobile applications

Due to the importance of the subjective assessment, generating suitable stimuli for different QoS levels is of high relevance. For mobile applications, one approach is to record the behavior of the mobile device for a specific combination of QoS parameters (QoS parameter tuple). This recorded behavior can finally be used as a stimulus for a QoE assessment such as Double Stimulus Continuous Quality Scale (DSCQS) [7], MUlti Stimulus test with Hidden Reference and Anchor (MUSHRA) [8], Subjective Assessment Methodology for Video Quality (SAMVIQ) [9], [10], etc.

A very challenging aspect in stimuli generation for mobile applications is that in order to record a suitable stimulus for a given QoS parameter tuple of interest, the mobile network must be in the corresponding state—which can be difficult or even infeasible for a live network due to the continuously changing traffic load and radio channel conditions. Thus, a controlled environment allowing network emulation with the QoS parameters is required. In addition, the generation of a stimulus will often require specific user interactions with the device. These—and in particular their timing—need to be deterministic and reproducible in order to avoid side effects. Therefore, an automated interaction with the User-Interface (UI) of the mobile device is an important requirement.

Furthermore, smartphone users typically use a wide range of applications with very different characteristics [11] such as e-mail, video streaming or web-browsing. Therefore, it is assumed in the following that for each category of applications, stimuli of different types must be generated.

1.2 Contributions and structure of the paper

The main contributions of this paper are:

1. Outlining a concept for a framework allowing the automated, reproducible generation of stimuli for the subjective assessment of mobile applications.
2. Presenting QoEval as a novel open-source implementation of a framework for stimuli generation.
3. A first evaluation of the concept and its implementation based on typical example use-cases.

The paper is structured as follows: Section 2 briefly summarizes the related work and Section 3 gives an overview of methods for the subjective assessment of QoE, which we consider the primary usage of the stimuli generated by our QoEval framework. In Section 4, the basic idea and the system architecture of QoEval are explained, a publicly available open-source implementation of this concept is described in detail in Section 5 and evaluated in Section 6. Finally, Section 7 concludes the paper with a brief summary and outlook.

2 Related work

The main motivation for the framework presented in this paper is the automated generation of stimuli for QoE assessment of applications on mobile devices. We are currently not aware of other publicly available open-source frameworks with the same focus. However, in the more general areas of QoE analysis, UI automation and network emulation for mobile devices, several related publications and tools making use of similar technologies exist.

QoE analysis on mobile devices: Due to the high relevance of QoE for a cellular network provider, this topic obtains high attention in the research community. Cecchet et al. developed the software infrastructure mBenchLab for measuring QoE of web applications on mobile devices. It uses the well-known Selenium framework to automate the interaction with the Android web browser and to record QoE statistics for each page. Therefore, mBenchLab is limited to QoE analysis of cloud-based web applications. Prometheus, a prototype system developed at AT&T by Aggarwal et al., applies machine learning to derive a mapping of passive network measurement results to a QoE metric. For training, mobile apps are instrumented to monitor the QoE, e.g. to record the number of video stalls during video playback. In addition, QoS characteristics such as data volume, rates and signal strength are recorded. Afterwards, the model learns the relation of QoS measurement data to the QoE metric. A different approach named QoE Doctor [12] is proposed by Chen et al. of T-Mobile—it applies UI automation techniques to replay user behavior and measure user-perceived latency based on UI changes. The UI is controlled via the InstrumentationTestCase API within a re-signed application package (APK), therefore QoE Doctor does not require a specific instrumentation or access to the mobile application source code. In contrast to these three examples, which are considering QoE based on objective, automatically performed measurements, Casas et al. combine subjective lab tests with end-device passive measurements in a comprehensive analysis [13]. In their subjective study, they use real Android devices which are connected to the Internet via WiFi and all downlink traffic is routed through the netem emulator [14]—which is similar to the approach proposed in our framework. By comparing the results obtained in the lab to those in a live network, they find out that lab results are highly applicable. However, for all related projects mentioned, the automatic generation of stimuli for subjective tests is out-of-scope.

UI automation and testing: An aspect where QoEval is similar to automated black-box testing approaches for mobile devices is the requirement to automatically execute a mobile application in a reproducible way. This implies the need to reset the mobile application state, to start the execution of an activity (e.g. by sending an intent), to observe and control the UI (i.e. sending input events such as touch, click, swipe or hardware key). Many publications and tools exist in this field—a systematic review is presented by Kong et al. in [15], while Cruz et al. focus on the aspect of UI automation and the energy footprint in [16]. In QoEval, we use AndroidViewClient [16], [17] because of its open-source license and a simple Python integration—yet, a different UI automation framework could easily be integrated.

Network emulation for mobile devices: For adapting the QoS of the Internet connection of the mobile device, the framework proposed in this paper uses network emulation. This approach is quite common and has been used by several other research projects. Serban et al. developed a hybrid emulation testbed [18] for testing

Android-based applications in military scenarios which models radio transmissions with specific military waveforms. Similitude [19], a framework for evaluating mobile phone apps for intelligent transportation systems, runs emulated Android clients and emulates the mobile network in the network simulator ns-3. In a similar way, our group has also evaluated mobile network emulation for Android devices for pedestrian communication applications [20] using the network simulator OMNeT++. However, for the QoEval framework, we use the much simpler emulation approach via netem [14] within the Linux kernel due to its lower computational complexity. Furthermore, for stimuli generation, the emulation of QoS parameters such as latency and receive data rates without a detailed model of the wireless network is sufficient.

For a more comprehensive overview of QoE on mobile devices, the reader is referred to survey articles dedicated to this topic, e.g. [5].

3 Background: subjective assessment of quality of experience

Different methods for the subjective assessment of QoE have been developed—in this section we briefly present two methods which can be adapted for the assessment of mobile applications.

3.1 MUSHRA tests

The perception of data services can be assessed by a modified version of MUSHRA. This test method [21], [22] has been designed to provide a reliable and repeatable measure of the quality of intermediate-quality signals [21]. Originally intended for the subjective evaluation of audio quality, the test methodology can be adapted to data services. In MUSHRA multiple stimuli, i.e. differently processed versions of a source, are simultaneously presented on a display to be randomly played by pressing a button. This procedure allows the test subject during the assessment to watch one version and to switch fast to another version. The reference version is also presented, and the subjects are instructed to rate the quality of the different stimuli relative to this reference on a scale ranging from 0 to 100, whilst 100 corresponds to the quality of the reference. The test also includes a hidden reference and so-called anchors among the versions to be assessed. Thus, the reference also appears as one of the stimuli to be graded by the subjects [22]. The inclusion of anchors enables a stable use of the subjective rating scale [8] and both, the hidden reference and anchors, provide a reference grid that covers the grading scale [22].

3.2 SAMVIQ tests

SAMVIQ is a test method primarily aiming at the subjective evaluation of video in a multimedia environment and has been introduced first by the European Broadcasting Union (EBU) [23]. Similar to MUSHRA, SAMVIQ is a multiple stimuli assessment methodology on a continuous quality scale. Two references are used in a session. The first one is defined as the high-quality anchor and the second one is randomly included as hidden reference amongst the differently processed stimuli to be assessed.

The subject is allowed to select the viewing order of the sequences and can repeat viewings during the assessment [24]. In contrast to Absolute Category Rating (ACR) tests [25], the presence of the reference in MUSHRA and SAMVIQ guarantees that the subjects know, how the quality of the versions really should be [22].

3.3 Adapting MUSHRA/SAMVIQ for the assessment of mobile applications

Both test methodologies are similar and can even be combined [10] and used for the assessment of mobile applications. The assumptions are that the stimuli are video files showing the loading of a webpage over the course of time for the assessment of the QoE of web-browsing. For the evaluation of the QoE of video streaming, the generated stimuli represent a short sequence of a video clip including the initial waiting time before reproduction start on the display as well as re-buffering phases occurring during the presentation of the video clip. For the assessment of the QoE of retrieval of e-mails, the video files represent the process of clicking on the header of the e-mail, downloading the e-mail with the defined delay from the server and finally showing the contents of the e-mail on the display. For a better evaluation of the end-user's perception, the quality assessment can be performed by people recruited on the street instead of selecting professional raters, which is a further deviation from the strict MUSHRA concept [22].

4 QoEval system architecture

As motivated in Section 1, the focus of the QoEval framework is the automated, reproducible generation of stimuli for the subjective assessment of mobile applications. The basic idea is to execute the mobile application on a real, remotely controlled smartphone whose network connection is modified in a way that the QoS parameters of interest are met, as illustrated in Figure 2. The corresponding user-experience of the mobile application (i.e. usually the video and audio) is captured and afterwards post-processed into a suitable format (e.g. converted to a suitable video and audio container), to be used as stimulus in user-centric studies.

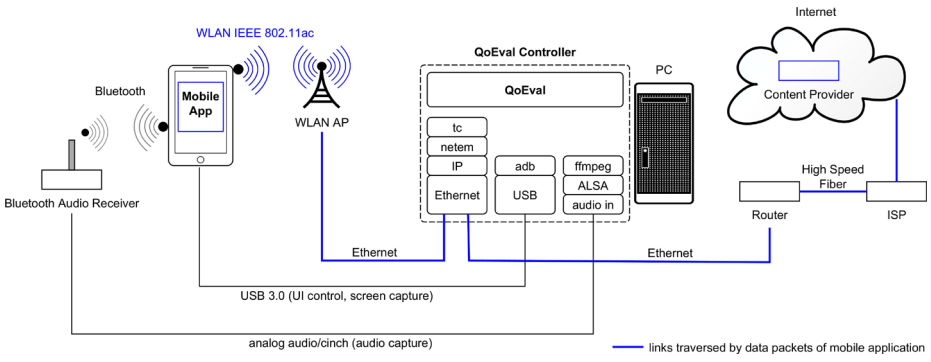

Fig. 2. Overview of QoEval System: data traffic of the mobile device is influenced by the QoEval controller according to the specified QoS parameter tuples. The UI of the mobile device is remotely controlled, all audio and video output of the mobile app are captured and post-processed by QoEval to generate a stimulus

4.1 Stimuli sets

Methods for subjective assessment of QoE usually vary one (or more) QoS parameters to analyze the relation between QoE and QoS. In the following, we will use the term *stimuli set* to describe a set of stimuli generated by varying a specific QoS parameter. For example, to assess the influence of the downlink data rate on the QoE of a specific mobile application, one would generate a stimuli set in which the downlink data rate is varied within the range of interest while all other parameter values (e.g. uplink data rate, uplink/downlink delay, etc.) are kept constant.

4.2 Mobile application types

As mentioned in Section 1.1, a wide range of mobile applications exists. We assume in the following that these applications and their typical uses-cases can be categorized according to their QoS requirements and input parameters. For example, applications of the type "video streaming" (such as YouTube, Vimeo, …) will usually have similar input parameters (URL of video to be played, desired video resolution, …), require relatively high downlink data rates and the corresponding stimuli need to capture video and audio. In contrast, applications of type "local transport and ticketing" use as input parameters the destination and a suitable UI control flow. These applications have relatively low requirements with respect to their uplink and downlink data rate and stimuli will consist of captured video but do not need to include audio.

Within the framework, theses application types enable us to simplify the configuration of the stimulus generation—since based on the application type the required parameters can be read and processes such as capturing and post-processing can be configured so that all relevant output for an application type is captured.

4.3 Input parameter tuples

Within the QoEval framework, three parameter tuples are used as input to generate a stimulus: a QoS, a mobile application, and a post-processing parameter tuple.

QoS Parameter Tuple p_q specifies the QoS parameters of the mobile network as it should be visible to the mobile application. Typical parameters of the tuple are the initial connection initialization delay (e.g. to connect to the mobile network and establish a Packet Data Protocol context), uplink data rate, downlink data rate, uplink latency and downlink latency. As mentioned in Section 4.1, within a stimuli set usually one of these parameters is varied.

In addition to the QoS of the used mobile network, it is well-know that the QoE is also influenced by a variety of other factors. For example, in [26] Ickin et al. report on basis of a user-study about additional factors that significantly impact the QoE such as the user-interface design, application performance, battery efficiency, phone features and user routines. However, these side factors are out-of-scope for the framework presented in this article. Therefore, we assume that they are constant for a specific set of stimuli, as defined in Section 4.1.

Mobile Application Parameter Tuple p_a defines all parameters required to launch the mobile application and execute the use-case to be represented by a stimulus. Usually, p_a will be constant for a stimuli set. Which parameters are required within p_a depends on the application type (Section 4.2).

Post-Processing Parameter Tuple p_p includes all parameters that are required for post-processing the captured audio and video contents to generate the stimuli. For example, during post-processing the captured video is often limited to only the part relevant for the stimulus. In order to detect this relevant section automatically, trigger images and detection thresholds are required, which are part of $\boldsymbol{p}_p$.

4.4 Data and control flow

As shown in Figure 2, the system basically consists of three coupled components:

(a) the mobile device on which the mobile application runs
(b) the QoEval Controller, which runs the QoEval framework, performs network emulation based on QoS parameters and coordinates the generation of the stimuli
(c) a high-speed Internet connection to the content provider required for the respective mobile applications.

While the mobile application runs on the mobile device, its cellular communication (i.e. Global System for Mobile Communications (GSM)/Enhanced Data rates for GSM Evolution (EDGE), Long Term Evolution (LTE)/LTE-Advanced (LTE-A), 5th Generation (5G)) is disabled and all in- and outgoing data traffic of the mobile application (blue line in Figure 2) is received and sent via a dedicated, high-speed Wireless Local Area Network (WLAN) connection to the QoEval framework where the QoS parameters can be controlled. The QoEval Controller then forwards the data traffic to the respective content provider in the Internet.

In order to be able to set the desired QoS parameters when generating a stimulus, the QoS parameters of the Internet connection must exceed those defined in the QoS parameter tuple p_q for this stimulus. As explained in more detail in Section 5, before starting the stimulus generation the framework verifies whether this condition can be fulfilled. Furthermore, for some parameters within p_q, e.g. the uplink and downlink delay, the framework needs to consider the influence of the WLAN connection and the connection to the Internet. Table 1 summarizes the parameters in p_q which QoEval currently takes into account by modifying the data flow.

Table 1. Parameters within p_q and corresponding handling by QoEval

Symbol	Description [unit]	Handling in QoEval
T_{init}	connection initialization time [ms]	During the first T_{init} milliseconds of a stimulus, all data traffic (up-/downlink) is blocked.
R_{UL}	uplink data rate [kbps]	At the QoEval Controller the uplink data rate is limited to R_{UL}.
R_{DL}	downlink data rate [kbps]	At the QoEval Controller the downlink data rate is limited to R_{DL}.
D_{UL}	uplink delay [ms]	Before generation of each stimulus, QoEval measures the uplink delay d_{ULdev} at the device. If it exceeds D_{UL}, the stimulus cannot be generated. Otherwise, QoEval configures the network emulation with a delay of $D_{UL} - d_{ULdev}$ so that the mobile application sees an uplink delay of D_{UL}.
D_{DL}	downlink delay [ms]	Analogously to the procedure for D_{UL}, network emulation configures a delay of $D_{DL} - d_{DLdev}$.

Besides the connections required for handling the data traffic of the mobile application (blue line in Figure 2), two more connections are required: Via an Universal Serial Bus (USB) 3.0 connection, QoEval controls the UI of the mobile device. This is necessary for the preparation before recording a stimulus (e.g. to unlock the device, reset the application state, etc.) as well as during the recording, e.g. to select buttons, enter text or get information what is currently shown on the device if a use-case depends on the availability of a specific information on the device. Furthermore, the same connection is also used for controlling the video capturing on the mobile device and later copying the captured video for post-processing to the QoEval Controller. Since capturing audio directly on the mobile device often is not feasible and many novel mobile devices do not have a headphone jack, we use a Bluetooth audio receiver (A2DP profile) to receive the audio, convert it to an analog audio output and capture it separately at the QoEval controller. For mobile devices with a headphone jack or with the capability to capture audio directly on the device, this connection is not necessary.

4.5 Generating stimuli: processing phases

For generating a stimulus, the QoEval framework steps through three major processing phases:

1. **Preparation** of the mobile device and emulation environment:
   (a) **Measure the delay bias at the mobile device:** The QoEval controller establishes the connection to the mobile device and measures the Round-Trip Time (RTT) for the connection from the mobile device via the QoEval controller to the server of the content provider (see Figure 2) when no network emulation is active. For simplicity, it is assumed that the delay is symmetrical, i.e. the uplink delay d_{ULdev} as introduced in Table 1 as well as the downlink delay d_{DLdev} can be approximated by 0.5 RTT.
   (b) **Configure network emulation:** At the QoEval controller, a network emulation is set-up so that the mobile device sees a network connection with the parameters specified within p_q (Section 4.4). A second RTT measurement is performed to check the emulation setup.
   (c) **Prepare use-case:** Depending on the use-case type of the stimulus, additional preparation needs to be performed, e.g. for a web-browsing use-case, the browser-cache on the mobile device must be cleared. Furthermore, for all use-case types, in which the date or time is visible on the mobile device UI, the device time is set to a specific value that is identical for all stimuli within a set to avoid deductions based on these values. Furthermore, the audio volume is set to a constant value.
2. **Execution** of use-case and capturing of raw stimulus material:
   (a) **Capture video and audio** on the mobile device: During the complete execution phase, the mobile device screen is captured to a video file. For all stimuli where audio is also relevant, the audio output is also captured—usually to a separate audio file since most mobile devices do not support audio capturing directly at the mobile device.

(b) **Emulate** network according to the QoS parameter tuple: p_q The QoEval controller activates the network emulation. It sets up the required network traffic filters, so that all in- and outgoing traffic of the mobile device is routed through the emulation module.

(c) **Control UI** of the mobile device: The QoEval framework can interact with the mobile device UI according to a use-case and stimulus specific script. It reads all UI elements visible on the screen and can send arbitrary touch and keypad events. For example, the UI script can wait for a specific button to appear on the mobile device UI screen and send a touch event with a configurable delay to select the button when it appears. In this way, the complete user interaction required for a stimulus can be performed in a reproducible way.

(d) **Record data traffic statistics:** While the use-case is executed, network statistics are collected to enable a later analysis of the transmitted and received data.

3. **Post-processing** of raw material to create the final stimulus:

(a) **Join audio and video:** The QoEval controller copies the captured video file from the mobile device to its local storage and joins it with the captured audio.

(b) **Detect start and end of stimulus:** In general, capturing the output on the mobile device will start a short time prior to the beginning of the stimulus to guarantee that all required output is captured. For example, for a web-browsing use-case, capturing could start when the browser is launched on the mobile device but the stimulus should begin at the time when the web address is entered in the browser. Analogously, capturing will stop some time after the use-case is completed. In order to detect (and later cut) the parts relevant for the stimulus, so-called trigger images can be specified—these are used by the framework to detect the start and end of the stimulus.

(c) **Detect end of initial buffering phase (optional):** For some use-cases, the initial buffering time within the captured video can include images which could be distracting. For example, in case of YouTube video streaming, images of other videos might be visible during the initial buffering phase until the selected video starts. In order to avoid distracting a test person, the framework can optionally try to detect the end of the initial buffering phase—enabling a replacement of this initial part by a standardized buffering animation in step (e).

(d) **Cut joined video:** Based on the start and end times detected in (b), the video is now cut so that it only includes parts that are relevant for the stimulus.

(e) **Substitute distracting parts of the video (optional):** If required by the use-case, parts of the video such as the initial buffering phase can be replaced by a standard animation, e.g. the initial buffering phase detected in (c).

(f) **Recode video:** Video is usually captured with a simple encoding so that the computational performance impact on the mobile device is low. However, for subjective QoE assessment, a different video encoding is often required. Therefore, in this last post-processing step, the video container is recoded. Of course, the recoding should use parameters, which do not have a noticeable impact on the video (and audio) quality itself.

For reproducibility, these phases need to be consecutively applied for all stimuli within a stimuli set, without any mandatory manual user-interaction.

4.6 Option: time-variant QoS parameters $p_q(t)$

Until now, the QoS parameter tuple p_q as introduced in Section 4.4 was assumed to be constant during the time of a stimulus. However, in general the QoS parameters during a connection in a mobile network are time-variant—which can have a significant impact on its QoE. In order to enable the framework to generate stimuli with time-variant QoS parameters $p_q(t)$, the network emulation within the QoEval controller (see Section 4.5, Execution phase) is extended: While capturing the mobile device output, QoEval can apply $p_q(t)$—thus the time-variant influence of these parameters becomes visible in the recorded stimulus.

4.7 Option: artificially generated buffering phases

For some use-case types, such as video streaming on a mobile device, a low QoS will lead to re-buffering phases on the mobile device, which is perceived by the user by an interruption of the ongoing video reproduction. In some cases, the influence of the duration and number of these re-buffering phases on the QoE assessment is of particular interest. One possible approach to achieve a particular duration or length is to find corresponding QoS parameters for the network emulation within the QoEval controller (see Section 4.5, Execution phase). However, directly inferring these QoS parameters can be challenging, since usually the internal buffering and rate adaptation strategy of a mobile streaming app such as YouTube is unknown. Therefore, QoEval optionally adds artificial buffering animations during the post-processing phase (see Section 4.5).

5 Prototype implementation

The QoEval architecture (Figure 2) and concept as described in Section 4 were prototypically implemented based on open-source components and we make this implementation and its source code publicly available [27]. This section briefly describes the QoEval Controller implementation, the hardware used within the experimental testbed and the mobile application types supported by the prototype.

5.1 QoEval controller

The QoEval Controller coordinates the process of generating a stimuli set (see Section 4.5), implements the required network emulation functionalities as specified within the input parameter tuples (Section 4.3) and interacts with the mobile application on the real mobile device (Section 4.4). It also includes functionalities to record network parameters such as uplink and downlink data rate, so that for each generated stimulus the data rate for the mobile application on the device can be analyzed.

The main implementation of the QoEval Controller itself, which is not time-critical, is a Python 3.9 package referred to as *qoeval_pkg* in the following. Input parameters, as introduced in Section 4.3, are specified in a Comma-Separated-Value (CSV) parameter file. The QoEval Controller supports a command-line and a simple graphical user-interface. These are used to configure all relevant parameters and select a stimuli set or an individual stimulus to be generated.

Emulating mobile network conditions corresponding to the QoS parameters p_q (see Section 4.4, Table 1) is implemented based on Linux network traffic control: all in- and outgoing data traffic of the application on the mobile device is routed through the *netem* kernel module [14], [28], which allows to control data rate, delay, loss, corruption, duplication and reordering. The *qoeval_pkg* controls *netem* by executing Linux kernel traffic control (tc) calls in a dedicated process. Optionally, data packet statistics can be recorded for later analysis. This is implemented based on the well-known Python packages *pyshark* for monitoring data traffic and *pandas*, *matplotlib* for analyzing and plotting the results.

For controlling the mobile device and application (Section 4.5), the prototype is currently limited to the Android platform: Within the preparation phase, QoEval has to be able to reset the device to a specific state (e.g. reset locally stored data of the mobile application, set device volume, etc.) to achieve reproducibility. Furthermore, QoEval measures the RTT on the mobile device since it needs to be considered when configuring the network emulation. During the execution phase, QoEval controls the UI of the mobile application. All these tasks are platform-specific, since various calls to APIs on the mobile device are required—the prototype performs these tasks using the Android Debugging Bridge (ADB) and AndroidViewClient [16], [17]. For supporting additional mobile platforms such as iOS, this interface towards the mobile device needs to be modified. For recording the screen of the mobile device during the execution phase, QoEval uses Genymobile Srccpy [29] that can capture the content directly on the device. After the execution phase, the captured video is copied via USB-connection to the QoEval Controller, where it is combined with the separately captured audio part and post-processed.

The post-processing steps outlined in Section 4.5 are implemented in QoEval in a separate module. Analyzing and recoding the stimulus video is based on the popular ffmpeg framework [30], the optional artificial buffering phases (see Section 4.7) are inserted using the bufferer Python package [31].

5.2 Supported mobile application types

As argued in Section 4.2, categorizing the mobile applications simplifies generating the stimuli since for applications of the same type similar parameters are required. For the QoEval prototype, we support four application types listed in Table 2. These were selected based on the following assumptions: Mobile video streaming (VS) is a popular application that requires a relatively high downlink data rate. Therefore, we expect to see a significant impact of the QoS parameters on the perceived video quality,

the duration of the initial buffering phase as well as interruptions due to re-buffering. In order to be able to evaluate the impact of the number and duration of re-buffering phases on the QoE, QoEval can insert artificial buffering phases during post-processing. Since in this case, additional input parameters such as the number and duration of the re-buffering phases are required, it is considered to be a different use-case type (VSB). For both video streaming types, QoEval can optionally set a specific video resolution supported by the YouTube app, or let the YouTube player automatically adapt the resolution. In the latter case, QoEval can visualize the used video resolution at the end of the execution phase.

Web Browsing (WB) and the more generic App Launch (AL) types allow to generate stimuli sets for a wide range of other popular mobile applications. User-interaction for these applications is specified as a list of user-interaction elements, each consisting of a condition (e.g. a text that needs to be visible on the mobile device) and an action (e.g. selecting a button, entering a text or performing a swipe-action on the device). To model a specific time which a user would need to read the content on the screen, an additional delay can be added. The framework can easily be extended with further mobile application types, such as online radio, messaging, gaming, etc.

Table 2. Mobile application types supported by current QoEval implementation

ID	Name	Description
VS	Video Streaming	playback of video within the YouTube app
VSB	Video Streaming + Generated Buffering	VS extended with artificially generated re-buffering phases added during post-processing
WB	Web Browsing	opening a webpage within the standard browser on the mobile device, optionally interacting with the page (e.g. enter search term)
AL	App Launch	launch a mobile application, retrieve online information, interact with the application

5.3 Implementation of time-variant QoS parameters

As introduced in Section 4.6, QoEval can optionally consider time-variant QoS parameters $p_q(t)$. The prototype reads QoS parameters from a CSV file with tuples p_q, each extended with a duration for which the parameter set should be active. These are passed to the controller, which will repeatedly iterate over these tuples during stimuli generation and use each for the corresponding specified time frame to control the QoS emulation. Figure 3 shows a simple example for a time-variant downlink data rate. The shape of this step function is derived from measurements in a live network using a test mobile. It represents the period immediately before a re-buffering phase occurred caused by a combination of temporarily high traffic load and poor local radio channel conditions at the cell border with high interference and several adjacent cells within a 4 dB measurement window.

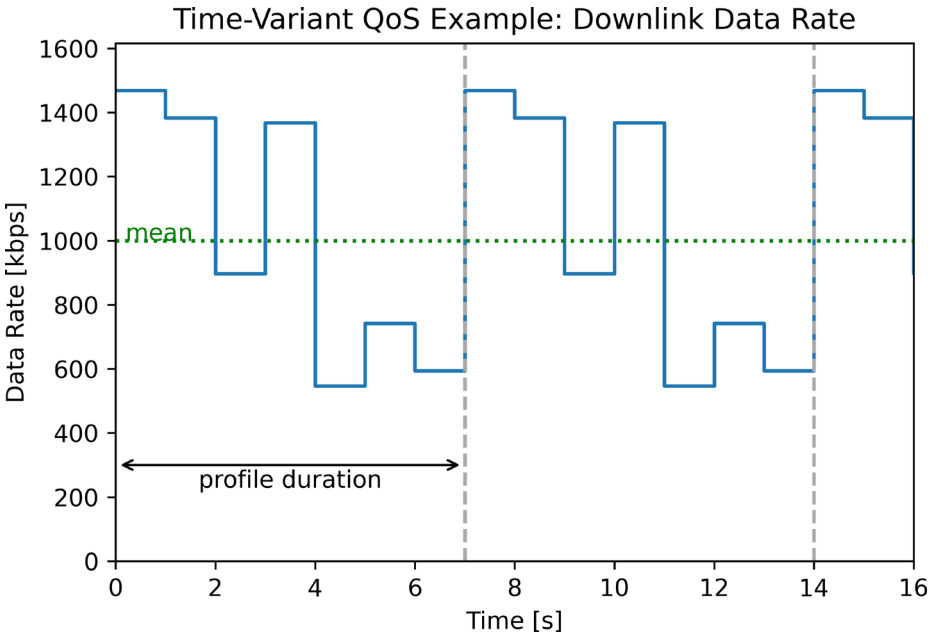

Fig. 3. Example for modelling time-variant QoS parameters in QoEval: time-variant downlink data rate profile is repeatedly applied in the emulation. In this case, the profile of length 7 s is based on a measurement in a live network, where the available downlink data rate was averaged in one second intervals and later scaled to achieve the desired mean rate (here: 1000 kbps)

For applying the time-variant QoS parameters during network emulation, the controller (see 5.1) starts a separate thread which updates the *netem* kernel module [28] [14] parameters by using the Linux traffic control command line interface. Since each update of the QoS parameters takes approx. 20 ms within our experimental testbed (see 5.4), this approach is only feasible if the duration during which each QoS parameter tuple should be active exceeds this update time significantly—which is a limitation of the current prototype implementation.

5.4 Experimental testbed

For an evaluation of the proposed approach, it was implemented within an experimental testbed with the components listed in Table 3. All software components of the QoEval prototype as described above were installed and a first measurement on the mobile device measuring the up- and downlink data rate as well as the latency without any network emulation was done in order to confirm that the mobile device can access hosts in the Internet with a high data rate (>50 Mbps) and low latency (<10 ms). The results of the evaluation in this experimental testbed are summarized in the next section.

Table 3. Hardware configuration of experimental testbed

Component	Hardware Description
QoEval Controller	Intel Core i7-8700K CPU @ 3.70GHz, 16 GB RAM, Linux 5.11.0-40-lowlatency kernel, Ubuntu 21.04, NVIDIA GeForce GTX 1060 6GB
Mobile Device	Google Pixel 5, Android Version 11, Build RQ3A.210705, Baseband-Version g7250-00132
WLAN Access Point	TP-Link EAP225 AC1350
Audio Receiver	Elegiant Wireless Audio Receiver, Bluetooth 5.0

6 Evaluation

Main motivation for this initial experimental evaluation of QoEval is to confirm the feasibility of the proposed approach. Therefore, typical examples for stimuli in each of the supported mobile application types (Section 5.2) are selected, generated via QoEval and evaluated with respect to the typical QoE-relevant aspects such as number of re-bufferings or loading times. Due to the focus on the QoEval framework, a user-centric study with the generated stimuli is out-of-scope of this article.

6.1 Video streaming (VS)

VS is a mobile application type where we expect to see a strong influence of QoS variations. This evaluation uses the native YouTube app (Version 15.18.39) on the mobile device to generate a stimuli set. In each stimulus, the same video[1] was streamed to the mobile device while varying the QoS parameters, as listed in Table 4. Commonly used metrics for QoE assessment models for video streaming applications are initial buffering, re-buffering, and stalling [6]. Additionally, we also evaluate the used codec/video resolution since the YouTube app in its default settings automatically adapts to the currently available downlink data rate.

Scenario 1: default settings (auto codec). In addition to inspecting the QoE relevant metrics, we first validate the network emulation of QoEval by monitoring the QoS parameters while recording a stimulus. Figure 4 shows an example, in which the achieved downlink data rate during VS is plotted for two different QoS parameter tuples. While the data rate varies depending on the demand of the VS application and availability of streaming data, the maximum is limited by QoEval to R_{DL} of the corresponding QoS parameter tuple in Table 4. Similar measurements were done to confirm the emulation for all other QoS parameters supported by QoEval—we omit the corresponding result plots here due to space limitations.

The QoE-relevant metrics we evaluated are based on the recorded and post-processed stimuli sets. Table 5 lists the time until the video stream playback starts

[1] A classical concert of the Wiener Philharmoniker performing "Beethoven: Symphony no. 5 in C Minor, op. 67" conducted by Christian Thielemann available in full HD at https://www.youtube.com/watch?v=yTL8j-JU_ow, excerpt from appr. 00:00:57 to 00:01:14 used for all VS stimuli sets presented in this paper.

(time to start), the number of re-bufferings, the duration of re-buffering and the auto-selected codec that is used at the end of the stimulus, i.e. when the YouTube app has adapted to the current QoS. IDs in the table are the same as used in Table 4. It can be observed that with decreasing QoS the time to start increases significantly. No re-buffering during video playback occurs as long as R_{DL} is equal or higher than 5000 kbps. Also, with decreasing QoS, the VS app selects a codec of lower resolution. We also assume that switching to the codec of lower resolution is the reason for observing no further re-buffering events in this stimulus after occurrence of the first re-buffering event that lead to the codec adaptation.

Table 4. QoS parameter tuples p_q for VS evaluation

ID	T_{init} [ms]	R_{UL} [kbps]	R_{DL} [kbps]	D_{UL} [ms]	D_{DL} [ms]
1	720	80	50000	16	16
2	720	63	20000	16	16
3	720	55	10000	16	16
4	720	50	5000	16	16
5	3300	45	2000	160	160
6	6450	42	1000	220	220
7	7500	41	750	235	235
8	9200	41	500	260	260

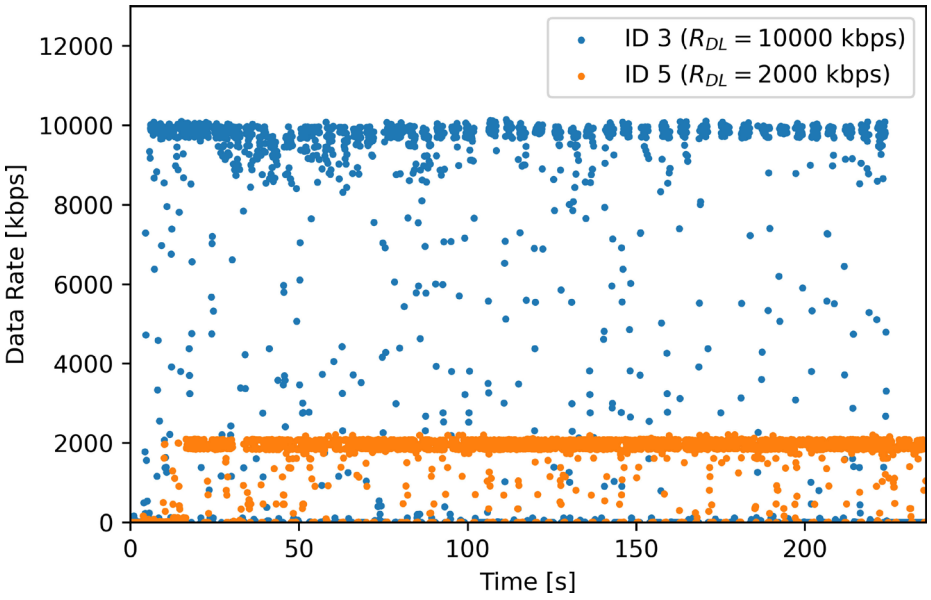

Fig. 4. Effect of QoEval settings for QoS parameters: As the measurement illustrates, the downlink data rate R_{DL} available at the mobile device is limited according to the QoS setting of the respective ID

Table 5. QoE relevant metrics for VS—Scenario 1

ID	Time to Start [ms]	Number of Re-bufferings	Duration of Re-buff. [ms]	Auto-selected Codec
1	5266	0	–	1080p
2	7402	0	–	1080p
3	8419	0	–	1080p
4	11870	0	–	720p
5	29813	1	4781	480p
6	54816	1	8817	360p
7	68322	1	7242	240p
8	108549	1	12281	240p

Scenario 2: fixed codec (1080p). In contrast to the previous scenario, we now configure the YouTube app (via QoEval automated UI interaction) to use the 1080p codec and disable the adaptive codec selection. Since in this case the app cannot automatically adapt the resolution when the QoS is low, multiple re-buffering phases within one stimulus occur as shown in Table 6. Up to the QoS tuple ID 4 (R_{DL} = 5000 kbps), the initial buffering in combination with the available downlink data rate allow the mobile application to play back the video stream without any re-bufferings. When the available downlink data rate decreases to 2000 kbps (QoS tuple ID 5), three re-buffering phases with long durations occur. A decrease of R_{DL} to 1000 kbps (ID 6) leads to a considerably long time to start delay (we assume that the YouTube app increases its buffer size due to the low data rate)—which would not be acceptable for most mobile users. This stimulus also reveals an unacceptable high number of re-buffering phases of long durations. The complete stimuli set covers the range of all three zones introduced in Figure 1.

Table 6. QoE relevant metrics for VS—Scenario 2 (codec fixed at 1080p)

ID	Time to Start [ms]	Number of Re-bufferings	Duration of Re-bufferings [ms]
1	5658	0	–
2	7207	0	–
3	8036	0	–
4	8874	0	–
5	25016	3	29342; 43264; 19142
6	63000	4	26348; 22766; 25483; 23649

6.2 Web browsing (WB)

To evaluate the support of QoEval for mobile applications of type WB, we model a Google search for the city of Perpignan with the following UI interaction: The user opens the standard browser (Chrome) on the smartphone, enters the URL of the Google website, enters the search term, starts the search and views the results for 2.5 s. Thus, the test subject is put into the situation of executing an inter-active task. As QoE

relevant metric, we use the total time until the interaction is complete, i.e. the time between starting the mobile browser until the search results have been completely visible for 2.5 s. Since side-effects of any content within the browser cache need to be avoided, QoEval clears the browser cache completely during the preparation phase of stimulus generation. The results in Figure 5 illustrate that until the downlink data rate decreases to 500 kbps (ID 8, Table 7), decreasing QoS does not lead to an increased UI time for this example. This part of the described stimuli set corresponds to zone 1 in Figure 1, for which saturation is reached, i.e. an increase of the QoS does not lead to an increase of QoE.

Table 7. QoS parameter tuples p_q for WB and AL evaluation

ID	T_{init} [ms]	R_{UL} [kbps]	R_{DL} [kbps]	D_{UL} [ms]	D_{DL} [ms]
1	40	350	50000	18	18
2	40	350	20000	18	18
3	40	350	10000	18	18
4	85	300	5000	18	18
5	160	220	2000	18	18
6	180	186	1000	18	18
7	184	180	750	18	18
8	190	172	500	18	18
9	200	155	200	18	18
10	205	80	100	18	18

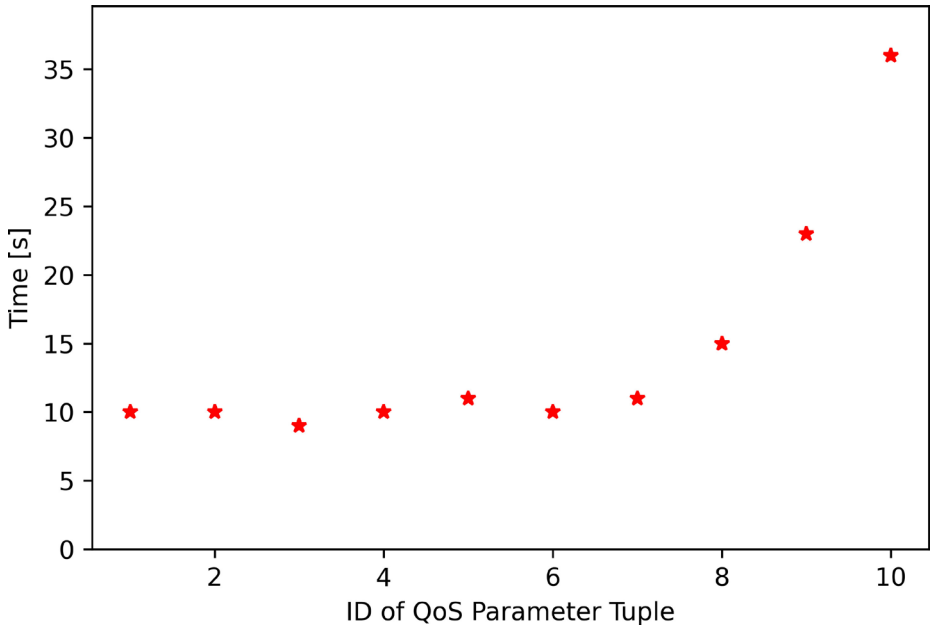

Fig. 5. WB search use-case: Up to QoS parameter tuple with ID 8 (i.e. a downlink data rate of 500 kbps), a decreasing QoS does not lead to an increase in total time

6.3 Mobile ticketing app (AL)

For this mobile application type, we evaluate the use-case of searching for trip information in a typical mobile ticketing app[2]: The user opens the app, enters start and stop locations for the trip (Munich to Hamburg), waits until search results are displayed, searches for later trips, scrolls down and views details on a specific trip in the app. QoS parameters are the same as for WB, see Table 7. As shown in Figure 6, this specific mobile app is even less sensitive to low QoS. However, we assume that this result is only valid for this specific mobile application, since we suppose that it is optimized for low QoS, since this occurs frequently on typical train routes.

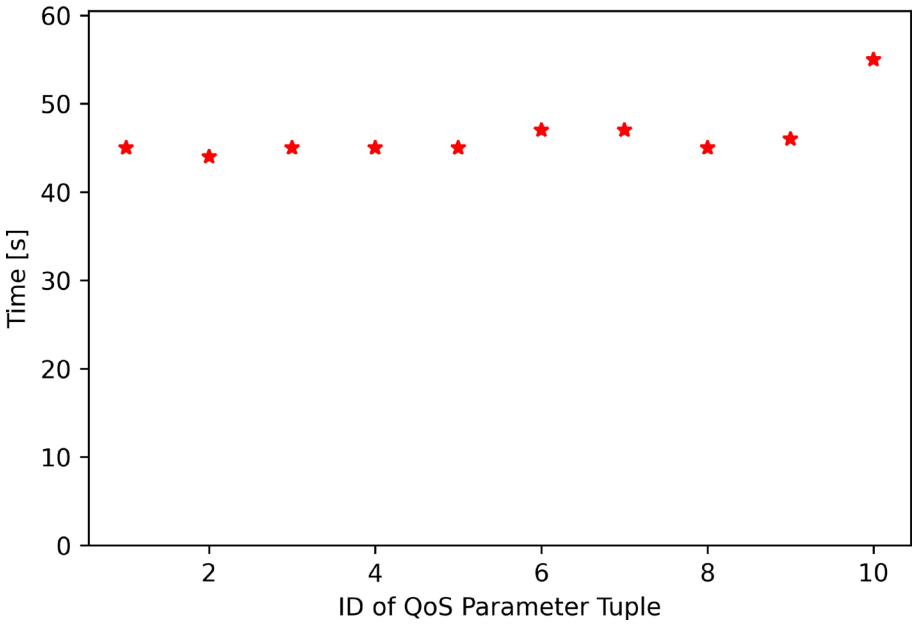

Fig. 6. AL mobile ticketing use-case: Up to QoS parameter tuple with ID 9 (i.e. a downlink data rate of 200 kbps, see Table 7), a decreasing QoS does not lead to an increase in total time

7 Conclusion

In this paper, we presented a concept for enabling the automated, reproducible generation of stimuli for the subjective assessment of mobile applications. The proposed QoEval framework was prototypically implemented and is publicly available as open source, hopefully inspiring further research in this area. An initial evaluation demonstrated the feasibility of the proposed approach by generating stimuli for three typical

[2] Used mobile ticketing app: DB Navigator Version 21.06.p04.00 of Deutsche Bahn AG (most popular mobile ticketing app for train tickets in Germany)

mobile application types: video streaming, web-browsing, and a native mobile ticketing application. The impact of the QoS parameters on QoE metrics such as the number of re-buffering events, codec or total time for a user-interaction was evaluated—illustrating that the QoS requirements are application-specific and can differ by several orders of magnitude.

The next step is to use the generated stimuli for QoE assessment based on subjective tests, where users give feedback in a metric such as the Mean Opinion Score (MOS), which will be part of our future work.

8 References

[1] M. Fiedler, K. Kilkki, and P. Reichl, "From Quality of Service to Quality of Experience," Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2009.

[2] N. Elshennawy, "Modified Proportional Fair Scheduling Algorithm for Heterogeneous LTE-A Networks," *International Journal of Interactive Mobile Technologies (iJIM)*, Bd. 14, p. 22–34, 2020. https://doi.org/10.3991/ijim.v14i10.14389

[3] M. Alreshoodi and J. Woods, "Survey on QoE\QoS Correlation Models For Multimedia Services," *CoRR*, Bd. abs/1306.0221, 2013.

[4] Z. Hu, H. Yan, T. Yan, H. Geng, and G. Liu, "Evaluating QoE in VoIP networks with QoS mapping and machine learning algorithms," *Neurocomputing*, Bd. 386, p. 63–83, 2020. https://doi.org/10.1016/j.neucom.2019.12.072

[5] A. C. Garcia and S. Casas, "Quality of Experience in Mobile Applications: A Systematic Mapping of Metrics and Tools," *International Journal of Interactive Mobile Technologies (iJIM)*, Bd. 14, p. 126–139, 2020. https://doi.org/10.3991/ijim.v14i08.12819

[6] ITU-T, *Recommendation ITU-T G.1011: Reference guide to quality of experience assessment methodologies*, 2016.

[7] K. Bouraqia, E. Sabir, M. Sadik, and L. Ladid, "Quality of Experience for Streaming Services: Measurements, Challenges and Insights," *IEEE Access*, Bd. 8, p. 13341–13361, 2020. https://doi.org/10.1109/ACCESS.2020.2965099

[8] ITU-R, *Recommendation ITU-R BS.1534-3: Method for the subjective assessment of intermediate quality level of audio systems*, 2015.

[9] ITU-R, *Recommendation ITU-R BT.500-14: Methodologies for the subjective assessment of the quality of television images*, 2019.

[10] H. Thoma, "A system for Subjective Evaluation of Audio, Video and Audiovisual Quality using MUSHRA and SAMVIQ Methods," in *Proc. of Fourth International Workshop on Quality of Multimedia Experience*, 2012. https://doi.org/10.1109/QoMEX.2012.6263877

[11] M. Jamalova and M. Constantinovits, "Attitudes Toward Smartphone Characteristics: What Do Users Pay For?," *International Journal of Interactive Mobile Technologies (iJIM)*, Bd. 14, p. 15–33, 2020. https://doi.org/10.3991/ijim.v14i09.13089

[12] Q. A. Chen, H. Luo, S. Rosen, Z. M. Mao, K. Iyer, J. Hui, K. Sontineni, and K. Lau, "QoE Doctor," in *Proc. of 2014 Conference on Internet Measurement*, 2014. https://doi.org/10.1145/2663716.2663726

[13] P. Casas, M. Seufert, F. Wamser, B. Gardlo, A. Sackl, and R. Schatz, "Next to You: Monitoring Quality of Experience in Cellular Networks From the End-Devices," *IEEE Transactions on Network and Service Management*, Bd. 13, p. 181–196, June 2016. https://doi.org/10.1109/TNSM.2016.2537645

[14] J. D. Beshay, A. Francini, and R. Prakash, "On the Fidelity of Single-Machine Network Emulation in Linux," in *Proc. of 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 2015. https://doi.org/10.1109/MASCOTS.2015.18

[15] P. Kong, L. Li, J. Gao, K. Liu, T. F. Bissyande, and J. Klein, "Automated Testing of Android Apps: A Systematic Literature Review," *IEEE Transactions on Reliability*, Bd. 68, p. 45–66, March 2019. https://doi.org/10.1109/TR.2018.2865733

[16] L. Cruz and R. Abreu, "On the Energy Footprint of Mobile Testing Frameworks," *IEEE Transactions on Software Engineering*, Bd. 47, p. 2260–2271, October 2021. https://doi.org/10.1109/TSE.2019.2946163

[17] D. T. Milano, "AndroidViewClient," 2021. [Online]. Available: https://github.com/dtmilano/AndroidViewClient

[18] C. Serban, A. Poylisher, A. Sapello, Y. Gottlieb, C. J. Chiang, and R. Chadha, "Testing android devices for tactical networks: A hybrid emulation testbed approach," in *Proc. of MILCOM 2015 - 2015 IEEE Military Communications Conference*, 2015. https://doi.org/10.1109/MILCOM.2015.7357490

[19] S. N. Hetu, V. S. Hamishagi, and L.-S. Peh, "Similitude: Interfacing a Traffic Simulator and Network Simulator with Emulated Android Clients," in *Proc. of 80th Vehicular Technology Conference (VTC2014-Fall)*, 2014. https://doi.org/10.1109/VTCFall.2014.6966178

[20] M. Rupp, S. Schuhbäck, and L. Wischhof, "Coupling Microscopic Mobility and Mobile Network Emulation for Pedestrian Communication Applications," *CoRR*, Bd. abs/2109.12018, 2021.

[21] A. Mason, "The MUSHRA Audio Subjective Test Method, BBC Research & Development, White Paper WHP 038," 2002.

[22] M. Säily, G. Sébire, and E. Riddington, Gsm/Edge: Evolution and Performance, WILEY, 2010. https://doi.org/10.1002/9780470669624

[23] T. Oelbaum, "Design and Verification of Video Quality Metrics," München, 2008.

[24] S. Péchard, R. Pépion, and P. Le Callet, "Suitable Methodology in Subjective Video Quality Assessment: A Resolution Dependent Paradigm," in *Proc. of International Workshop on Image Media Quality and its Applications, IMQA2008*, Kyoto, 2008.

[25] ITU-T, *Recommendation ITU-T P.800: Methods for Subjective Determination of Transmission Quality*, 1996.

[26] S. Ickin, K. Wac, M. Fiedler, L. Janowski, J.-H. Hong, and A. K. Dey, "Factors Influencing Quality of Experience of Commonly Used Mobile Applications," *IEEE Communications Magazine*, Bd. 50, p. 48–56, April 2012. https://doi.org/10.1109/MCOM.2012.6178833

[27] L. Wischhof and J. A. Krahl, "QoEval Prototype Implementation - Source Code," 2021. [Online]. Available: https://github.com/research-com/QoEval

[28] S. Hemminger, "Network Emulation with NetEm," in *Proc. of Australia's 6th National Linux Conference*, 2005.

[29] Genymobile, "Scrcpy," November 2021. [Online]. Available: https://github.com/Genymobile/scrcpy

[30] FFmpeg Developers, "FFmpeg: A complete, cross-platform solution to record, convert and stream audio and video," 2021. [Online]. Available: https://ffmpeg.org/

[31] W. Robitza, "Bufferer," 2021. [Online]. Available: https://github.com/slhck/bufferer

9 Authors

Lars Wischhof is a professor at the department of computer science and mathematics at Munich University of Applied Sciences HM, Munich. He received his Dipl.-Ing. degree in computer engineering from Technische Universität Berlin, and a Dr.-Ing. from the Institute of Communications at Technische Universität Hamburg-Harburg, Germany. His research interests are in applied mobile communication, in particular vehicular and pedestrian communication. He is member of IEEE, ACM and VDE. (email: wischhof@ieee.org)

Jan Andreas Krahl is currently working towards his Bachelor of Science degree at Munich University of Applied Sciences HM. He is student research assistant at the department of computer science and mathematics (email: krahl.jan@hm.edu).

Robert Müllner graduated from University of Munich, Germany, with a diploma degree in physics, worked for Siemens AG and Nokia Siemens Networks on simulation models and product development for mobile networks. His research interests are Quality of Service and Quality of Experience, radio resource management and performance efficiency. Since 2010, he is with Telefónica working on radio design and quality strategies with focus on Quality of Experience (email: robert.muellner@telefonica.com).

Article submitted 2021-12-07. Final acceptance 2022-01-22. Final version published as submitted by the authors.