electronics

Article

Practical Evaluation of VMAF Perceptual Video Quality for WebRTC Applications

Boni García [1,*], Luis López-Fernández [1], Francisco Gortázar [2] and Micael Gallego [2]

[1] Theory on Signal and Communication and Telematic Systems and Computing, Universidad Rey Juan Carlos, 28943 Fuenlabrada, Spain
[2] Computing Science, Computer Architecture, Programming Languages and Systems and Statistics and Operative Investigation, Universidad Rey Juan Carlos, 28933 Móstoles, Spain
* Correspondence: boni.garcia@urjc.es

Received: 1 July 2019; Accepted: 30 July 2019; Published: 31 July 2019

Abstract: WebRTC is the umbrella term for several emergent technologies aimed to exchange real-time media in the Web. Like other media-related services, the perceived quality of WebRTC communication can be measured using Quality of Experience (QoE) indicators. QoE assessment methods can be classified as subjective (users' evaluation scores) or objective (models computed as a function of different parameters). In this paper, we focus on VMAF (Video Multi-method Assessment Fusion), which is an emergent full-reference objective video quality assessment model developed by Netflix. VMAF is typically used to assess video streaming services. This paper evaluates the use of VMAF in a different type of application: WebRTC. To that aim, we present a practical use case built on the top of well-known open source technologies, such as JUnit, Selenium, Docker, and FFmpeg. In addition to VMAF, we also calculate other objective QoE video metrics such as Visual Information Fidelity in the pixel domain (VIFp), Structural Similarity (SSIM), or Peak Signal-to-Noise Ratio (PSNR) applied to a WebRTC communication in different network conditions in terms of packet loss. Finally, we compare these objective results with a subjective evaluation using a Mean Opinion Score (MOS) scale to the same WebRTC streams. As a result, we found a strong correlation of the subjective video quality perceived in WebRTC video calls with the objective results computed with VMAF and VIFp in comparison with SSIM and PSNR and their variants.

Keywords: QoE; WebRTC; VMAF; video quality

1. Introduction

1.1. State-of-the-Art

Web Real-Time Communications (WebRTC) is a set of technologies aimed to enable high quality real-time communications between web browsers, mobile platforms, and other web-enabled devices [1]. WebRTC is being standardized by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF). W3C has developed the JavaScript APIs which allow media communication between peers, whilst IETF has developed the set of protocols required to setup and manage reliable media channels. WebRTC was born in May 2011. One of the first major milestones of the project took place in 2013, when the WebRTC interoperability between Firefox and Chrome was achieved. The compatibility with Android devices was supported in 2014. After that, Apple and Microsoft announced official WebRTC support for their browsers Safari and Edge respectively in 2017. Nowadays, the market around WebRTC continues growing, and, as forecasted by a recent report, it is expected to reach 7 billion WebRTC-compliant devices by 2020 [2]. All in all, developers and practitioners demand more and more effective mechanisms to assess the quality of WebRTC-based services [3].

Recently, the term "Quality of Experience" (QoE), mainly related to media communication, has been gaining momentum. This term was coined in contrast to Quality of Service (QoS) to express how users' perceptions are addressed [4]. QoE is a quality strategy first proposed by ITU-T as "the overall acceptability of an application or service, as perceived subjectively by the end-user" [5]. This definition implies that QoE includes the end-to-end system effects and also the user expectations and context. However, this definition was criticized because it only included the acceptability of QoE [6]. For that reason, a refined definition of QoE was proposed in the context of the COST Action IC1003 European Network on Quality of Experience in Multimedia Systems (Qualinet) project. The Qualinet White Paper [7] states that "QoE is the degree of delight or annoyance of the user of an application or service". This definition was included as amendment 5 of the ITU-T Recommendation P.10/G.100 [5].

QoE assessment is not trivial since user experience is difficult to quantify. Thus, QoE assessment methods can be classified as subjective or objective [8,9]. Subjective methods directly quantify QoE by soliciting users' evaluation scores. Subjective evaluation provides the most valid way to measure QoE, but it is time consuming and therefore costly. In order to alleviate this problem, objective quality models has been proposed [10]. Objective quality models compute a metric as a function of QoS parameters and external factors. Objective metrics are supposed to correlate well with the subjective results, which serve as the ground truth QoE.

Objective quality measurement methods are classified as follows [11]: (i) Parametric packet-layer models, which predict the QoE from the packet header information without analyzing media signals. (ii) Parametric planning models, which use quality planning parameters for networks and hosts. (iii) Bit-stream-layer models, in which encoded bit-stream and packet-layer data are used to estimate QoE. (iv) Hybrid models, in which bitstream, and/or packet-header information are taken as input signal. (v) Media-layer models, in which QoE is calculated using some reference and the degraded audio and video signals. Among these methods, media-layer models are further divided into: (i) Full reference: the degraded signal is compared with the original signal . (ii) Reduced reference: these methods require the degraded signal and only part of the reference (typically some feature data about the media is sent to the receiver using a control channel). (iii) No reference: these methods assess the quality only using the degraded signal together with some network and/or application-specific factors.

Video Multi-method Assessment Fusion (VMAF) was recently proposed by Netflix as a full reference perceptual video quality assessment model that combines quality-aware features to predict perceptual quality. VMAF combines human vision modeling with machine learning, offering a good prediction of the video QoE [12,13]. The development of VMAF started between NetFlix and the Professor C.C. Jay Kuo from University of Southern California. In June 2016, VMAF was first open sourced to GitHub (https://github.com/Netflix/vmaf). VMAF uses existing image quality metrics to predict video quality, such as Visual Information Fidelity (VIF) [14] or Detail Loss Metric (DLM) [15]. These features are fused using a supervised learning regression model to provide a single output result called VMAF score. This score ranges from 0 to 100 per video frame, 100 being the quality of a video identical to the reference [16]. As presented in the next section, VMAF is attracting a lot of attention both in academic research and in industrial initiatives nowadays.

1.2. Motivation and Contributions

Originally, the VMAF score was computed using Netflix video streams delivered over TCP (i.e., without packet loss nor bit errors) to adjust compression and scaling parameters that ultimately impact QoE. Moreover, VMAF is being used as an optimization criterion for better encoding decisions in different kinds of applications. For example, Orduna et al. prove that VMAF can be used without any specific training or adjustments to obtain the quality of 360-degree virtual reality sequences actually perceived by users [17]. Zadtootaghaj et al. use VMAF to analyze the video quality of gaming streaming services such as Twitch.tv or YouTubeGaming. In this work, aiming to a value of 65 for the VMAF score, the authors calculate the minimum encoding bitrate in order to significantly reduce the required bandwidth of different streaming games [18]. Sakaushi et al. present a video surveillance system in which video is coding using H.264/AVC and VMAF is used to measure how the quality of the video is degraded for different bitrates [19].

In this paper, we hypothesize that VMAF can be also used in a completely different scenario, i.e., to evaluate the video quality of real-time communications through WebRTC. This kind of communication is significantly different to the original aim of VMAF, since, in WebRTC, which is based in UDP transport, delays, reduced and frozen frame rate, and audiovisual desynchronization might happen. VMAF evaluates video quality following a per-frame approach, and, therefore, it cannot be applied directly to WebRTC reference and impairment streams. Nevertheless, we believe it is technically feasible to implement a mechanism to record the WebRTC stream sent and received in browsers, then carry out a frame alignment to these videos, and, finally, each frame can be compared with its corresponding distorted one using VMAF.

All in all, this paper presents several contributions which are completely missing in the current literature for the best of our knowledge. First, we evaluate the quality of a WebRTC video call using VMAF in different network scenarios. To that aim, we carry out a WebRTC one to one controlled (i.e., using a test sequence) video call. This communication is done first without packet loss, and, then, the same communication is repeated simulating different packet loss rates in the network. Second, the results obtained with VMAF are compared with other existing well-known objective video QoE metrics, namely VIFp, SSIM, MS-SSIM, PSNR, PSNR-HVS, and PSNR-HVS-M. Last but not least, the obtained WebRTC stream recordings are used to carry out a subjective evaluation using a heterogeneous group of people which are asked to rate the perceived video quality using a Mean Opinion Score (MOS) rate. In order to carry out the experimentation, a complete test infrastructure has been created and released as open source artifacts. As described in the next section, this infrastructure is based on well-known technologies such as JUnit 5 as the basis testing platform, Selenium to drive automatically the browsers which implements the WebRTC video call, and Docker to isolate the required browsers as containers, instrumenting the underlying network and simulating packet loss.

This paper has been done in the context of the ElasTest (https://elastest.io/) project, which is an H2020 European Union funded initiative. ElasTest has born with the big ambition of providing a comprehensive solution to ease the process of development, operation, and maintenance of end-to-end tests for different kind of applications, including web and mobile among others [20]. WebRTC is a relevant demonstrator technology in ElasTest, and thus, this work is a part of the research efforts in this line.

2. Materials and Methods

In order to assess the video quality of a WebRTC communication using VMAF, we need to setup the proper mechanisms to provide automated and repeatable assets to carry out the experimentation. To that aim, first we have created and open sourced a toolbox built on top of well-known technologies, such as JUnit 5, Selenium, and Docker. Second, we have designed a custom experiment to assess an existing WebRTC video call service by means of an automated test which allows for recording the WebRTC stream sent by a browser and the one received (i.e., degraded) by another browser. These video files are used to feed VMAF, and also additional objective full reference video quality methods. Finally, the same videos are used to carry out a subjective analysis of the perceived video quality.

2.1. Tooling Infrastructure

VMAF is a media-layer objective full-reference video quality metric. For that reason, in order to calculate the VMAF score, we need to use the original and the degraded signals as input. Therefore, to use VMAF in a WebRTC video call, the first step is to provide the capability to record the WebRTC media sent by a browser (original signal) as well as that received by one or several peers (degraded signal). This capability is not trivial, and so, in order to do it in an automated and repeatable manner, we have created, implemented, and released a specific toolbox, based on existing open source frameworks and utilities, as described next.

First of all, we use Selenium (https://www.seleniumhq.org/), which is the de facto standard framework for end-to-end web testing nowadays. It allows for driving programmatically web browsers, such as Chrome, Firefox, etc. Selenium is a very convenient tool for our purposes, due to the fact that, in order to assess WebRTC automatically, we need to manage programmatically browsers implementing the WebRTC stack. This feature is provided by Selenium WebDriver project.

Selenium is traditionally used in conjunction with a unit testing framework, which allows for running Selenium scripts exercising a System Under Test (SUT), a web application in this case. All in all, Selenium is used to evaluate a given feature and JUnit gives a verdict (pass or fail) about the automated web execution. In our case, we use the well-known testing framework JUnit (https://junit.org/junit5/). JUnit is the most popular test frameworks for Java, and it is considered one of the most influential frameworks in software engineering. Version 5 of JUnit was released on September 2017, and it has been designed as a modular framework from the scratch [21]. In its core, we find the *JUnit platform*, which is a generic executor of test cases within the Java Virtual Machine (JVM). Out of the box, two types of test engines are supported in the platform, namely: i) *Jupiter*, which is the brand-new programming and extension model of the JUnit 5 framework; and ii) *Vintage*, which allows running legacy JUnit tests (version 3 or 4) in the platform.

Another relevant aspect in our testing infrastructure is Docker (https://www.docker.com/), which is an open source software technology to pack and run any application as a lightweight and portable container. It provides a command-line program, a background daemon, and a set of remote services that simplifies the life cycle of containers. Docker is a very convenient asset for the purpose of this paper, since it can be used to isolate web browsers using containers. Moreover, the use of Docker allows the instrumentation of the web browsers, which is key in this piece of research. In particular, as introduced before, we are interested in evaluating the video quality of a WebRTC call using different network conditions in terms of packet loss. As explained next, this is implemented by changing the network setup within one of the browser Docker containers.

In order to use the presented technologies so far (Selenium, JUnit 5, and Docker), we have created a tool called Selenium-Jupiter (https://bonigarcia.github.io/selenium-jupiter/), which is an extension for JUnit 5 aimed to provide seamless integration with Selenium and Docker through JUnit 5 tests. Selenium-Jupiter provides a rich variety of features for web testing. At first glance, it allows for using local or remote browsers from JUnit 5 tests. Moreover, it allows for using browsers and Android devices in Docker containers out of the box. Selenium-Jupiter can be used to carry out different types of test of web applications, including functional, compatibility (i.e., using different browsers types and versions) or performance (i.e., using a big number of browsers) tests. Selenium-Jupiter is based on the following JUnit 5 extension model capabilities:

- Dependency injection. This capability enables different object types to be injected in JUnit 5 as methods or constructor parameters in test classes. Specifically, Selenium-Jupiter allows for creating subtypes of the WebDriver interface (e.g., `ChromeDriver`, `FirefoxDriver`, and so on) when defined as test parameters.
- Test lifecycle. The Jupiter extension model allows for executing custom code before and after actual tests. Selenium-Jupiter use this feature to properly create WebDriver objects before the test, resolving WebDriver binaries (e.g., chromedriver, geckodriver) in the case of local browsers, or pulling and executing containers in the case of Docker. After the test is executed, Selenium-Jupiter uses the before test callbacks to properly dispose browsers, containers, and so on.
- Test templates. These templates can be seen as a special kind of parameterized tests, in which the test is executed several times according to the data provided by some extension. Selenium-Jupiter uses this feature to define a group of browsers which will be used to execute a given test logic.

The Docker containers used in Selenium-Jupiter have been created and maintained by the ElasTest project. These images are open and available on Docker Hub (https://hub.docker.com/u/elastestbrowsers/) and are based on the containers provided by the Selenoid (https://aerokube.

com/selenoid/latest/) project (a scalable Selenium Hub implementation in Go language). A relevant feature of the ElasTest Docker browsers for WebRTC is the ability to record video and audio inside the container.

The last ingredient we need in our infrastructure is FFmpeg (https://ffmpeg.org/). FFmpeg is an open source project consisting of a suite of libraries for media handling. The FFmpeg program has been designed for command-line-based processing of video and audio files, including a wide variety of features, including format transcoding, editing (such as trimming or concatenation), or video scaling, among others.

2.2. Design of the Experiment

Our experiment is divided in two parts. On the one hand, an automated test built with Selenium-Jupiter is executed against an existing WebRTC service. As a result, several objective indicators (VMAF score and others) are calculated. After that, and using the WebRTC video recordings obtained in the test, a subjective analysis is also carried out.

2.2.1. Objective Evaluation

As depicted in Figure 1, the methodology for the objective evaluation can be seen as a process made by three separate stages. The first step consists on generating the input video. This action is aimed to generate a proper test video file to evaluate the QoE of a WebRTC communication using VMAF and other QoE video metrics. We use the tool FFmpeg to generate this input video. Several considerations need to be taken into account to create this video. Regarding the content, we have selected a video used in a similar study [22], which is a close up of a woman being interviewed (credits for the video file used: "Internet personality Christa Casebeer, aka Linuxchic on Alternageek.com, Take One 03" by Christian Einfeldt, https://archive.org/details/e-dv548_lwe08_christa_casebeer_004.ogg). This original file has a resolution of 540 × 360 pixels and it is encoded with H.264. This video is converted using FFmpeg to YUV4MPEG2 format (since it is the format required to feed a fake WebRTC stream, as described next). Moreover, the original resolution has been changed to 640 × 480 pixels. The reason to change the resolution has to do with some of the QoE metrics used at the end of the process. Specifically, in VIFp, the height and width of the video have to be multiple of 8, and, in MS-SSIM, the height and width of the video have to be multiple of 16 (required by the tool VQMT (https://mmspg.epfl.ch/downloads/vqmt/, described next). Finally, the original audio track has been removed, since we are only interested in assessing the video quality.

We use Chrome browsers in Docker containers to support the experiment. The reason to use Chrome and not a different browser such as Firefox is because Chrome provides a rich variety of utilities for automated testing. With respect to WebRTC, we use the following configuration capabilities:

- `--use-file-for-fake-video-capture=/path/to/video.y4m`: This option allows for feeding a video test sequence in Y4M format for WebRTC streaming instead of using live camera input. We use this option in the browser which acts as *presenter*, using the video generated with FFmpeg (called `test.y4m`).
- `--use-fake-ui-for-media-stream`: This option avoids the need to grant camera and microphone permissions before accessing to the WebRTC feed.
- `--use-fake-device-for-media-stream`: This option allows for feeding a synthetic video pattern (a green video with a spinner which completes a spin and beeps each second) for WebRTC streaming instead of using live camera input. We use this option in the Chrome browser acting as a *viewer*.

As introduced before, our test infrastructure needs to record the WebRTC stream sent by the presenter and the one received by the viewer. These two videos are latter used to feed VMAF and other QoE metrics. However, in order to calculate the VMAF score properly, recordings (presenter and viewer) need to start and finish at the same time. This synchronization can be tricky to achieve, since in WebRTC, the playback of the stream in the presenter starts a few moments before the actual transmission. Before a WebRTC peer can start a media session and the media can start flowing, a group of signaling activities has to take place: the SDP negotiation and the gathering of the ICE candidates. Due to this signaling, media in the viewer starts a few moments after the media started in the presenter [23]. In order to alleviate this problem, we present a simple but effective mechanism. Before sending the actual media of 60 s, we append at the beginning of the stream a padding video around 10 s. This media is going to be discarded in a post-processing stage since we are not interested in when exactly the padding starts in the viewer side. Additionally, the same padding content is appended at the end of the actual media, in order to avoid the same problem when stopping the WebRTC call. The scheme of this video sequence is depicted in Figure 2. The padding sequence is generated with FFmpeg using the `testsrc` source, which generates a test video pattern showing a color pattern, a scrolling gradient and a timestamp.

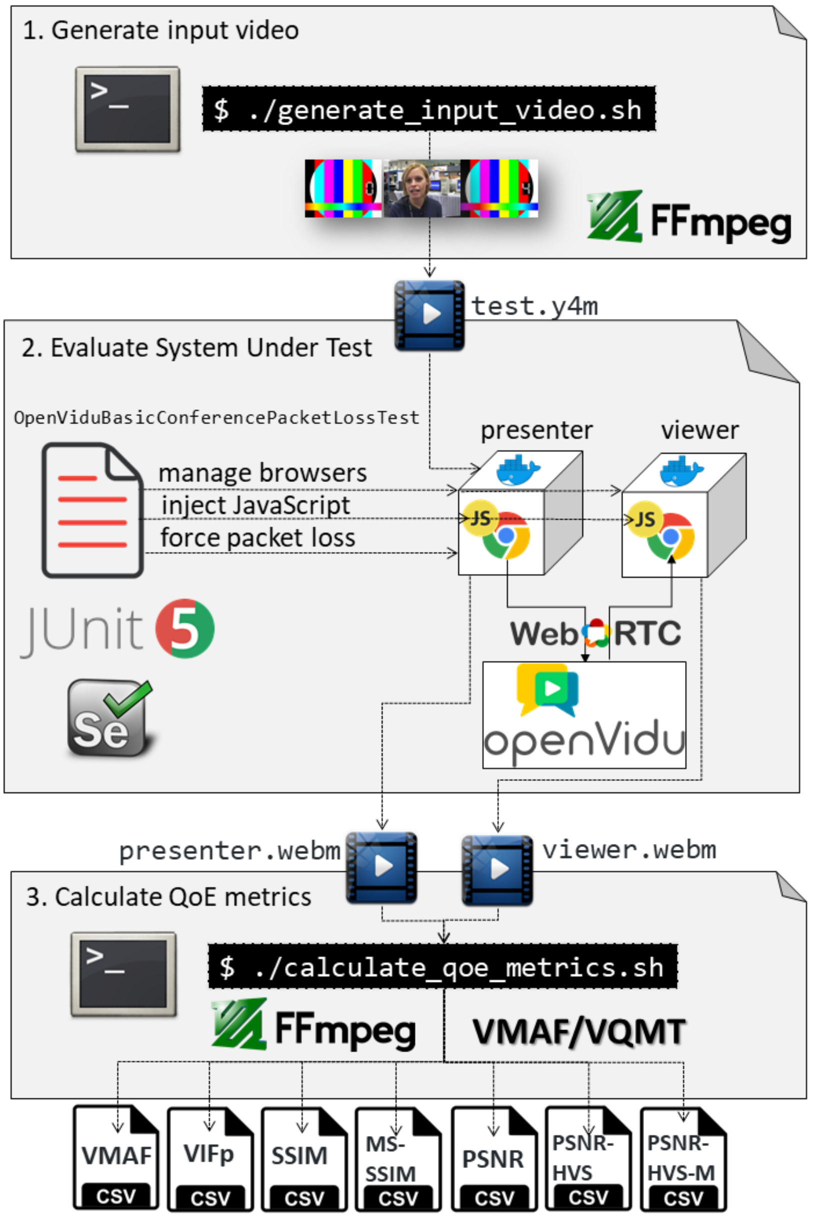

Figure 1. Objective assessment three-tier methodology.

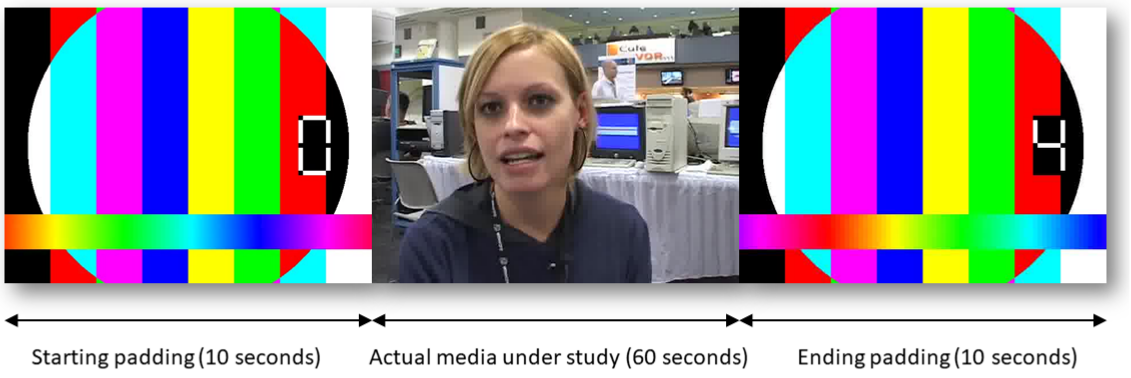
Starting padding (10 seconds)
Actual media under study (60 seconds)
Ending padding (10 seconds)

Figure 2. Test video sequence (`test.y4m`).

As a result of the first stage, we create the file `test.y4m`, which is used in the second step, i.e., the actual evaluation of the System Under Test (SUT). To carry out this evaluation, an automated Selenium test has been implemented in JUnit 5 using Selenium-Jupiter. The test is going to use two Chrome browsers in Docker containers. The first browser (presenter) sends the test video sequence by WebRTC to the second browser (viewer), which in turn uses the synthetic video pattern (green video with a spinner) provided by Chrome for test purposes. Figure 3 shows an screenshot of the presenter browser taken during the execution of the test.

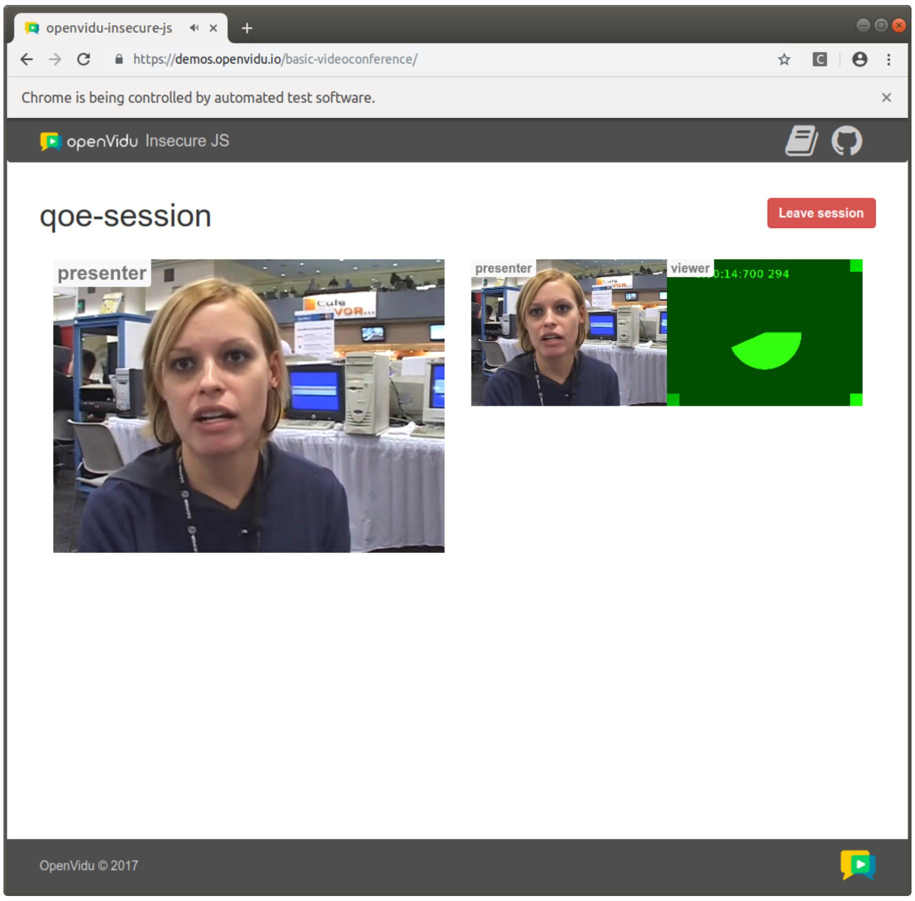

Figure 3. Screenshot of the OpenVidu basic conference demo being executed by the Selenium test.

Our System Under Test (SUT) is, of course, a WebRTC application. Nowadays, there are two widely accepted strategies for implementing group communications on WebRTC. On the one hand, Multipoint Controller Unit (MCU) topologies in which multiple input media streams are decoded, aggregated into a single output stream and re-encoded. On the other hand, Selective Forwarding Units (SFU) clone and forward the received encoded media stream onto many outgoing media streams [24]. Table 1 summarizes several relevant WebRTC media servers for group communications. Our objective is not evaluating the specific quality of a given WebRTC server, but to evaluate how the VMAF metric behaves in WebRTC in different scenarios of packet loss. For the shake of simplicity, we use the OpenVidu basic conference online demo (https://demos.openvidu.io/basic-videoconference/) as SUT in the experiment.

Table 1. Group communications WebRTC media serves.

Name	Description	URL
Medooze	A multiparty videoconferencing service based on MCU	http://www.medooze.com/
Licode	General purpose videoconferencing system	http://lynckia.com/licode/
Jitsi	SFU videoconferencing system	https://jitsi.org/
Janus	General purpose WebRTC server developed by Meetecho	https://janus.conf.meetecho.com/
OpenVidu	SFU videoconferencing system based on Kurento	https://openvidu.io/

Our JUnit 5 test is called `OpenViduBasicConferencePacketLossTest` and it is publicly available on GitHub (https://github.com/elastest/elastest-webrtc-qoe-meter/). This test uses Selenium-Jupiter to ease the use of browsers in Docker containers, as explained in Section 2.1. In order to record the presenter and viewer WebRTC stream, the test injects a JavaScript file in the browsers (presenter and viewer). Among other functions, this script uses the open source *RecordRTC* library (https://github.com/muaz-khan/RecordRTC), which allows for storing a WebRTC stream to a *webm* video file in a simple manner.

Moreover, the test is able to simulate a given rate of packet loss in the Docker container of the presenter. To that aim, an integer parameter is passed to the test execution. In our experiment, this value can be 0 (no packet loss), 10, 20, 30, 40, or 50 (and it represents the packet loss percentage to be induced in the presenter outbound network). When this figure is different from 0, the Linux command `tc` is invoked in the container. This tool allows for manipulating traffic control settings and it can be invoked using the shell. When the test finishes, a couple of video files, namely `presenter.webm` and `viewer.webm`, are generated using JavaScript and extracted from the browsers using Selenium. These two are latter used as input for the third and final stage of the experiment. In this stage, the resulting videos are processed and several QoE metrics are calculated.

Before the actual calculation of these QoE objective metrics, several actions need to be done. Since VMAF is the main QoE under study, we need to meet some requirements to compute it properly. VMAF produces one score per video frame. In other words, the input videos to calculate the VMAF score (`presenter.webm` and `viewer.webm` in our case) should be aligned. All in all, we use FFmpeg for remuxing the frame rate of the videos to a fixed value (24 FPS in our case, which is the frame rate used when generating the original test video). Then, we need to remove the starting and ending padding videos sent by WebRTC, using FFmpeg to cut the video. To do that, the Linux command `convert` is used to check when the color sequence displayed in padding video bars (white, cyan, purple, blue, yellow, green, red, and black) is present in each frame. This way, the script is able to determine automatically the exact frames in which the actual content starts and finishes.

The final part of this stage is the actual execution of the QoE objective algorithms to calculate the resulting video quality indicators. The first of these metrics is VMAF. The Netflix VMAF implementation is open source and it is available on GitHub (https://github.com/Netflix/vmaf). In addition to VMAF, additional existing QoE video metrics are also calculated. Specifically, we use the Video Quality Measurement Tool (VQMT, https://github.com/Rolinh/VQMT) open source suite, which provides implementation for the following objective video quality metrics:

- VIFp: Visual Information Fidelity in the pixel domain is derived from the mutual information between the input and the output of the Human Visual System (HVS) channel when no distortion channel is present [25].
- SSIM: Structural Similarity measures the difference of structure between the original and the distorted image in terms of luminance, contrast and structure [26].
- MS-SSIM: Multi-Scale SSIM, which is an advanced form of SSIM in which the evaluation is conducted through a process of sub-sampling, reminiscent of multiscale processing [27].
- PSNR: Peak Signal-to-Noise Ratio is the proportion between the maximum signal and the corruption noise [28].
- PSNR-HVS: It is an extension of PSNR incorporating properties of the HVS such as Contrast Sensitivity Function (CSF) [29].
- PSNR-HVS-M: It is an improvement of PSNR-HVS by taking into account visual masking [30].

All in all, when the script `calculate_qoe_metrics.sh` is executed, 7 Comma Separated Values (CSV) files are generated, one per QoE metric (VMAF, VIFp, SSIM, MS-SSIM, PSNR, PSNR-HVS, and PSNR-HVS-M). These files can be processed easily using a spreadsheet processor to analyze the outcome.

2.2.2. Subjective Evaluation

In order to compare the objective video quality results obtained in the objective assessment, a subjective evaluation is also carried out. Using the recordings of the viewer browsers in normal conditions and also with forced packet loss, a survey has been developed. The participants of the survey are asked to watch 6 different video sequences from the beginning to the end (1 min each video) and then score video quality perceived. We use a five point Mean Opinion Score (MOS) scale rated as follows: 1 = bad, 2 = poor, 3 = fair, 4 = good, and 5 = excellent. The survey was made public using the Twitter accounts of the authors in order to find a heterogeneous range of participants.

3. Results

Once both parts (objective and subjective evaluation) of the experiment has been finished, we are able to analyze the obtained data. Regarding the objective experimentation, we collect seven different CSV files. The raw data are available in the GitHub repository in which the rest of the experiment assets (JUnit test, scrips, etc) has been made public (https://github.com/elastest/elastest-webrtc-qoe-meter/). Each entry of each CSV data contains the result for a given QoE metrics (VMAF, VIFp, SSIM, MS-SSIM, PSNR, PSNR-HVS, and, PSNR-HVS-M) per frame. Taking into account that the test video sequence had 60 s and the frame rate is 24, we have around 1440 samples per QoE metric. As a first attempt to get an idea of the results, Figure 4 shows the evolution of each metric during the experiment time. In order to interpret this figure, it is important to know the range of each metric:

- VMAF scores range from 0 (lowest quality) to 100 (distorted and reference video are equal).
- VIFp score is bounded below by 0 (indicates that all information about the reference has been lost in the distortion channel) and 1 (the distorted and the reference video are the same)
- SSIM (and MS-SSIM) index is a decimal value between 0 (no structural similarity) and 1 (two identical sets of data).
- PSNR (and PSNR-HVS and PSNR-HVS-M) normal range lies between 20 dB (lower quality) and 60 dB (good quality) [31].

In the top left of Figure 4, we see the evolution of the different QoE metrics in time when no packet loss is carried out. We can check that the values remain quite stable, following a similar trend for each of the different metrics. The rest of the charts shows the evolution of the same QoE metrics but with different figures of packet loss (0%, 10%, 20%, 30%, 40%, and 50%). As expected, the resulting

QoE values evolve negatively as long as the packet loss increases. This is particularly relevant in the case of 40% and 50% packet loss.

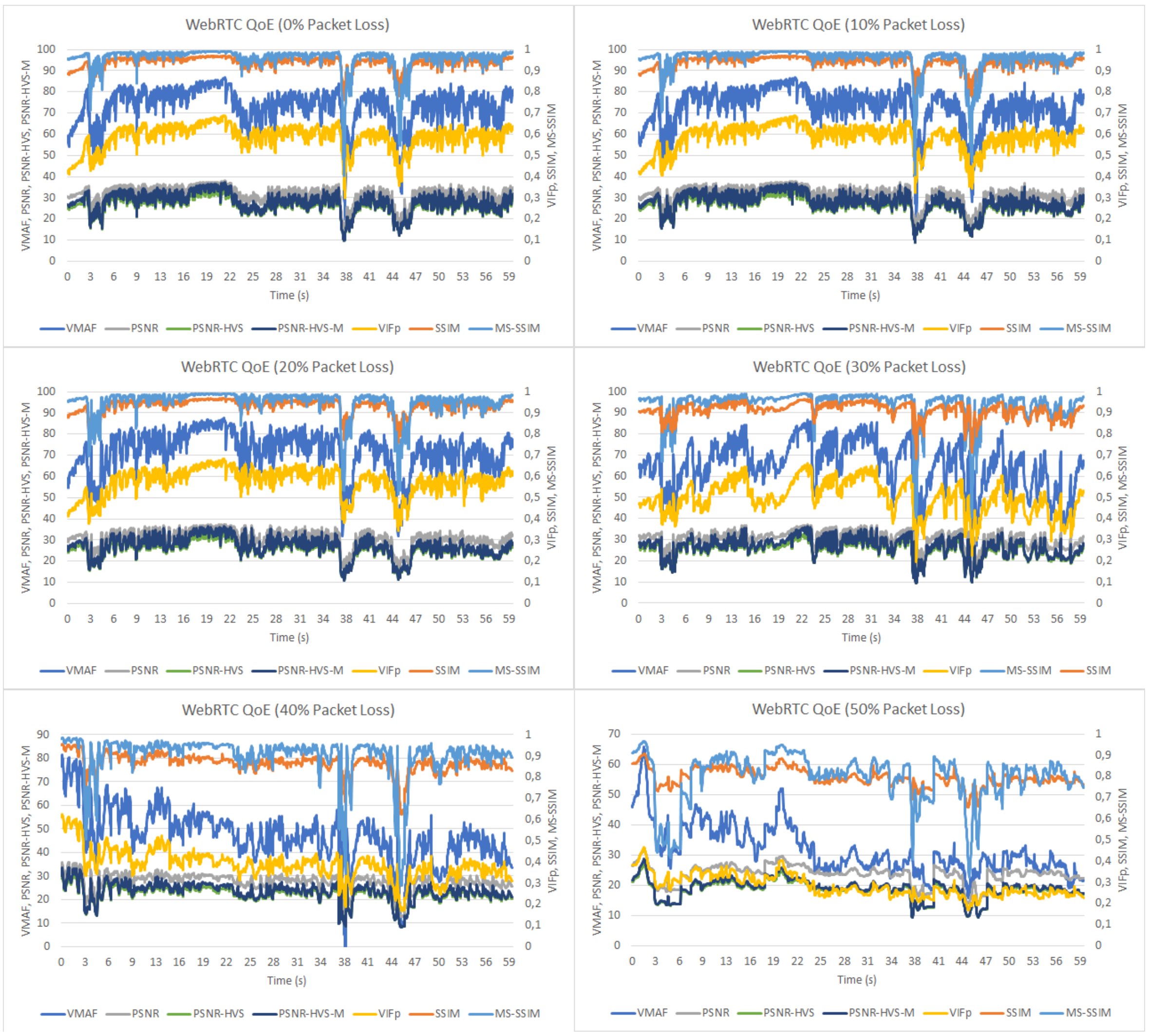

Figure 4. QoE vs. packet loss.

In order to visualize the evolution of QoE video quality in this experiment in a single chart, Figure 5 shows the average of each metric (VMAF, VIPp, SSIM, SSIM, PSNR, PSNR-HVS, and PSNR-HVS-M) for each packet loss rate (0%, 10%, 20%, 30%, 40%, and 50%). This figure shows clearly the rate of change for each metric. Interestingly, all of the QoE metrics show a good (or acceptable in the case of PSNR) level of video quality in the case of no packet loss together with the case of 10% and 20% packet loss: VMAF score around 72, VIFp score is around 0.58, SSIM/MS-SSIM index is around 0.94, and PSNR/PSNR-HVS/PSNR-HVS-M around 30 dB. Table 2 contains these values.

When the packet loss rate is higher than 20%, each quality metric shows a linear decrease of video quality. Another interesting finding is that both VMAF and VIFp show a lower tolerance to packet loss than the other QoE metrics under study (i.e., SSIM and PSNR in their different flavors). In other words, the loss of video quality when packet loss is higher is much more relevant in VMAF/VIFp than in SSIM/PSNR in our experiment.

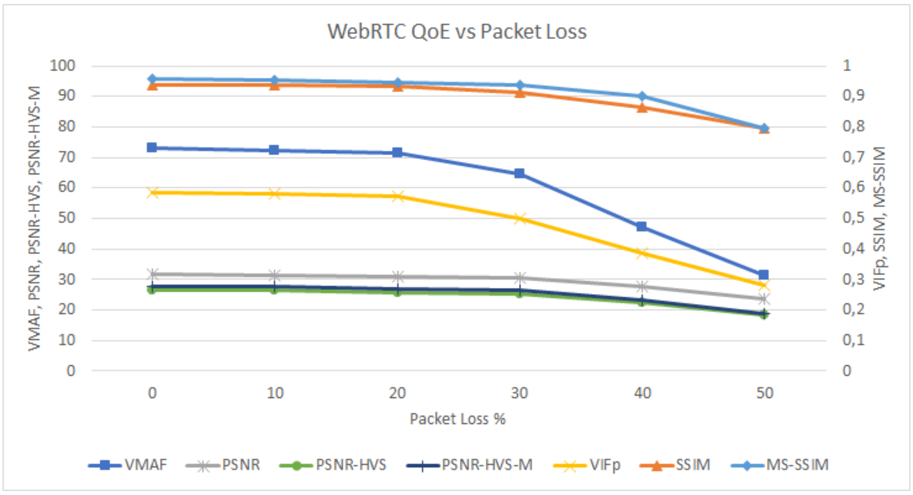

Figure 5. Mean QoE vs. packet loss (chart).

At this point, we can compare the objective with the subjective results, i.e., the MOS scores obtained in the survey. The questionnaire was answered by a total of 45 participants. Out of the total, five answers was considered as anomalies (e.g., rate all videos with the same MOS score) and it was discarded. All in all, we had 40 valid answers. Within these answers, there was a rate of 90% of men and 10% of women from 14 different countries (Australia, Canada, France, Germany, India, Ireland, Israel, Italy, Romania, Russia, Spain, Turkey, United Kingdom, and United States). The age of the participants ranges from 23 to 68 years (average 35.5 years, standard deviation 10.99 years). Figure 6 shows the mean MOS for each video sequence and the last column of Table 2 shows the specific values. As expected, the video quality perceived by the participants is better when no packet loss is present (the mean MOS score was 3.43 in in this case, i.e., between fair and good quality), and this value decreases as long as packet loss increases until a final value of 1 in the case of 50% packet loss (i.e., bad quality perceived).

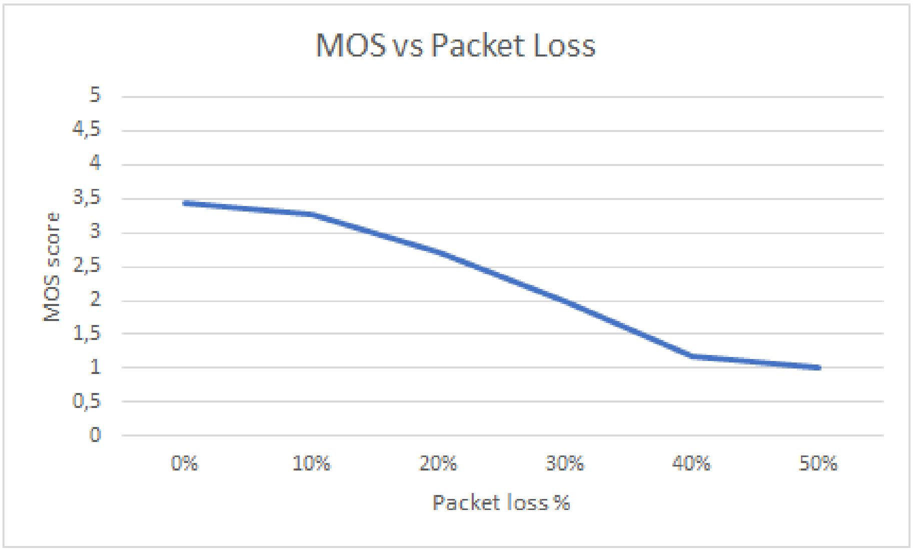

Figure 6. MOS vs. packet loss.

Table 2. Mean QoE (objective and subjective) vs. packet loss (values).

Packet Loss	VMAF	VIFp	SSIM	MS-SSIM	PSNR	PSNR-HVS	PSNR-HVS-M	MOS
0%	72.93	0.58	0.94	0.96	31.61	26.57	27.8	3.43
10%	72.44	0.58	0.94	0.95	31.48	26.45	27.67	3.28
20%	71.38	0.57	0.93	0.95	30.88	25.83	26.94	2.7
30%	64.74	0.5	0.91	0.94	30.38	25.35	26.6	1.98
40%	47.1	0.39	0.87	0.9	27.6	22.42	23.37	1.18
50%	31.52	0.28	0.79	0.79	23.64	18.43	18.94	1

In order to check if the subjective results from the survey match the objective results obtained automatically, we need to measure the extent to which these results tend to change together. In other words, we need to calculate the correlation coefficient between MOS and the rest of objective metrics, i.e., VMAF, VIFp, SSIM, MS-SSIM, PSNR, PSNR-HVS, and PSNR-HVS-M. To that aim, we calculate the Pearson correlation, which evaluates the strength and direction of linear relationship between pairs of continuous variables [32]. A coefficient of Pearson correlation (r) ranges from 0 (no correlation) to ± 1 (perfect correlation). A positive correlation means there is a direct relationship between the variables (i.e., change together) while a negative correlation means an inverse relationship (i.e., change opposite).

We use IBM SPSS Statistics to calculate the Pearson correlation between the MOS and the objective results. We also calculate the statistical significance (p) of this dataset. This value can be used in data analysis to reject a null hypothesis when p is lower than a given threshold (α). In our case, the null hypothesis (H_0) is that subjective and objective results are not correlated. A common value for this threshold in the literature is $\alpha = 0.05$. This value is used by researchers to claim statistical significance when $p \leq 0.05$. Nowadays, the concept of statistical significance is subject of controversy [33], and, therefore, a more stringent level such as $p \leq 0.01$ might be used to be particularly confident when claiming a given scientific finding [34].

The resulting figures for the Pearson correlation and the statistical significance in our research are summarized in Table 3. In the light of the results, we find there is a strong correlation between the subjective results (MOS) and all the objective results (VMAF, VIFp, SSIM, MS-SSIM, PSNR, PSNR-HVS, and PSNR-HVS-M) since r is always greater than 0.8. Nevertheless, and using a significance level of $p \leq 0.01$, we can only reject the null hypothesis H_0 in the case of VIFp ($r = 0.938$, $p = 0.006$) and VMAF ($r = 0.915$, $p = 0.010$). In other words, there is a very strong correlation between the subjective results (MOS score), i.e., the video quality perceived by a group of people, and the objective results obtained automatically with VIFp and VMAF using the same video of a WebRTC stream in different scenarios of packet loss, but not in the case of using SSIM and PSNR (and their variants).

This result seems reasonable for a number of reasons. First, VMAF is an aggregated QoE metric based, among others, in Visual Information Fidelity (VIF). Therefore, it makes sense that the trend of both metrics is similar for the same data input. Second, as reported in [35], the VIF index score of distorted images (equivalent to videos with packet loss in our experiment) showed a correlation index of 0.96 with respect to human perception of image quality, suggesting a strong correlation between the subjective and objective results, in the same way as the results of this paper. Finally, a previous study has shown that VMAF has a better performance that other quality metrics such as SSIM, PSNR-HVS and VQM-VFD when compared to subjective ratings [13], which again is aligned with the result of this piece of research.

Table 3. Pearson correlation (r) and statistical significance (p) of the subjective (MOS) compared to the objective results (VMAF, VIFp, SSIM, MS-SSIM, PSNR, PSNR-HVS, and PSNR-HVS-M).

	VMAF	VIFp	SSIM	MS-SSIM	PSNR	PSNR-HVS	PSNR-HVS-M
MOS	$r = 0.915$ $p = 0.010$	$r = 0.938$ $p = 0.006$	$r = 0.885$ $p = 0.019$	$r = 0.809$ $p = 0.051$	$r = 0.878$ $p = 0.022$	$r = 0.879$ $p = 0.021$	$r = 0.871$ $p = 0.024$

4. Conclusions

This paper presents a practical use case aimed to demonstrate that VMAF can be used as a mechanism to evaluate the video quality of WebRTC communications. VMAF is a perceptual full reference video quality assessment model developed by Netflix. VMAF score is typically applied to video streams delivered over TCP in order to find out optimization criterion for better encoding decisions. WebRTC is a quite different use case for VMAF, since in WebRTC different impairments such as delays, reduced and frozen frame rate, and audiovisual desynchronization might happen.

In order to evaluate VMAF in WebRTC, we have created a test infrastructure based on JUnit 5, Selenium, and Docker, which allows for driving web browsers in Docker container in a seamless manner. Containers allow for isolating the browsers when generating packet loss during a WebRTC communication. Then, an existing online WebRTC video call service based on the OpenVidu framework has been used to carry out a video call in different network scenarios. The video stream sent by a browser (presenter) and received by another browser (viewer) is recorded using JavaScript and Selenium. The frame rates of these recordings are then aligned using FFmpeg, and, as a result, these videos are used to calculate the resulting VMAF score per frame. In order to compare the VMAF results, additional existing objective metrics (VIFp, SSIM, MS-SSIM, PSNR, PSNR-HVS, and PSNR-HVS-M) are also calculated using the same video sequences (original and impairment) transmitted by WebRTC using different packet loss rates (0%, 10%, 20%, 30%, 40%, and 50%). Finally, a subjective evaluation based on a MOS score is also carried out using the same set of video recordings. In the light of the results, we conclude that VMAF and VIFp are objective metrics with a better correlation with subjective score compared to SSIM and PSNR (and their variants).

By releasing the source code and the data collected during the experiments, we enable other researchers to repeat or extend our methodology at a low implementation cost. Our test infrastructure allows for continuing this research line analyzing the impact of further QoE scenarios for WebRTC applications. For instance, our work can be extended by evaluating audiovisual desynchronization or audio quality, for instance using QoE assessment methods such as PESQ (Perceptual Evaluation of Speech Quality) or POLQA (Perceptual Objective Listening Quality Assessment) to name a few. Moreover, other WebRTC use cases can be exercised, for example using additional input video sequences, or different topologies such as the broadcasting scenario (one to many) or video chat (many to many), and using heterogeneous QoS scenarios (e.g., simulating network congestion or different traffic constraints).

Author Contributions: Conceptualization, B.G.; methodology, B.G. and M.G.; software, B.G.; validation, B.G. and F.G.; investigation, B.G.; resources, F.G. and M.G.; data curation, B.G.; writing–original draft preparation, B.G.; writing–review and editing, B.G. and F.G.; visualization, B.G. and F.G.; supervision, F.G. and M.G.; project administration, F.G. and M.G.; funding acquisition, L.L.-F.

Funding: This work has been supported by the European Commission under project ElasTest (H2020-ICT-10-2016, GA-731535); by the Regional Government of Madrid (CM) under project EDGEDATA-CM (P2018/TCS-4499) cofunded by FSE and FEDER; by the Spanish Government under projects LERNIM (RTC-2016-4674-7) and BugBirth (RTI2018-101963-B-I00) cofunded by the Ministry of Economy and Competitiveness, FEDER and AEI.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Loreto, S.; Romano, S.P. *Real-Time Communication with WebRTC: Peer-to-Peer in the Browser*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2014.
2. Sale, S.; Rebbeck, T. Operators Need to Engage with WebRTC and the Opportunities it Presents. *Anal. Mason* 2014. Available online: http://www.analysysmason.com/ (accessed on 1 July 2019).
3. Gouaillard, A.; Roux, L. Real-time communication testing evolution with WebRTC 1.0. In Proceedings of the Principles, Systems and Applications of IP Telecommunications (IPTComm), Chicago, IL, USA, 25–28 September 2017; pp. 1–8.

4. Hoßfeld, T.; Schatz, R.; Varela, M.; Timmerer, C. Challenges of QoE management for cloud applications. *IEEE Commun. Mag.* **2012**, *50*, 28–36. [CrossRef]
5. ITU-T. Recommendation P.10/G.100. Vocabulary for Performance and Quality of Service, 2006. Available online: https://standards.globalspec.com/std/10264138/G.100 (accessed on 1 July 2019).
6. Timmerer, C.; Ebrahimi, T.; Pereira, F. Toward a new assessment of quality. *Computer* **2015**, *48*, 108–110. [CrossRef]
7. Brunnström, K.; Beker, S.A.; De Moor, K.; Dooms, A.; Egger, S.; Garcia, M.N.; Hossfeld, T.; Jumisko-Pyykkö, S.; Keimel, C.; Larabi, M.C.; et al. Qualinet White Paper on Definitions of Quality of Experience. In Proceedings of the Output from the Fifth Qualinet Meeting, Novi Sad, Serbia, 12 March 2013.
8. Frnda, J.; Nedoma, J.; Vanus, J.; Martinek, R. A Hybrid QoS-QoE Estimation System for IPTV Service. *Electronics* **2019**, *8*, 585. [CrossRef]
9. Rehman, I.U.; Nasralla, M.M.; Philip, N.Y. Multilayer perceptron neural network-based QoS-aware, content-aware and device-aware QoE prediction model: A proposed prediction model for medical ultrasound streaming over small cell networks. *Electronics* **2019**, *8*, 194. [CrossRef]
10. Chen, Y.; Wu, K.; Zhang, Q. From QoS to QoE: A tutorial on video quality assessment. *IEEE Commun. Surv. Tutor.* **2014**, *17*, 1126–1165. [CrossRef]
11. Chikkerur, S.; Sundaram, V.; Reisslein, M.; Karam, L.J. Objective video quality assessment methods: A classification, review, and performance comparison. *IEEE Trans. Broadcast.* **2011**, *57*, 165–182. [CrossRef]
12. Liu, T.J.; Lin, Y.C.; Lin, W.; Kuo, C.C.J. Visual quality assessment: recent developments, coding applications and future trends. *APSIPA Trans. Signal Inf. Process.* **2013**, *2*, 1–20. [CrossRef]
13. Li, Z.; Aaron, A.; Katsavounidis, I.; Moorthy, A.; Manohara, M. Toward a practical perceptual video quality metric. *The Netflix Tech Blog*. Available online: https://medium.com/netflix-techblog/toward-a-practical-perceptual-video-quality-metric-653f208b9652 (accessed on 1 July 2019).
14. Sheikh, H.R.; Bovik, A.C. Image information and visual quality. In Proceedings of the 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing, Montreal Que, CA, USA, 17–21 May 2004; pp. 1520–6149.
15. Li, S.; Zhang, F.; Ma, L.; Ngan, K.N. Image quality assessment by separately evaluating detail losses and additive impairments. *IEEE Trans. Multimedia* **2011**, *13*, 935–949. [CrossRef]
16. Lin, J.Y.; Liu, T.J.; Wu, E.C.H.; Kuo, C.C.J. A fusion-based video quality assessment (FVQA) index. In Proceedings of the Signal and Information Processing Association Annual Summit and Conference (APSIPA), Siem Reap, Cambodia, 9–12 December 2014; pp. 1–5.
17. Orduna, M.; Díaz, C.; Muñoz, L.; Pérez, P.; Benito, I.; García, N. Video Multimethod Assessment Fusion (VMAF) on 360VR Contents. *arXiv* **2019**, arXiv:1901.06279.
18. Zadtootaghaj, S.; Schmidt, S.; Barman, N.; Möller, S.; Martini, M.G. A Classification of Video Games based on Game Characteristics linked to Video Coding Complexity. In Proceedings of the 16th Annual Workshop on Network and Systems Support for Games (NetGames), Amsterdam, The Netherlands, 12–15 June 2018; pp. 1–6.
19. Sakaushi, A.; Kanai, K.; Katto, J.; Tsuda, T. Image quality evaluations of image enhancement under various encoding rates for video surveillance system. In Proceedings of the IEEE 6th Global Conference on Consumer Electronics (GCCE), Nagoya, Japan, 24–27 October 2017; pp. 1–2.
20. Bertolino, A.; Calabró, A.; De Angelis, G.; Gallego, M.; García, B.; Gortázar, F. When the testing gets tough, the tough get ElasTest. In Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, Gothenburg, Sweden, 27 May–3 June 2018; pp. 17–20.
21. García, B. *Mastering Software Testing with JUnit 5: Comprehensive Guide to Develop High Quality Java Applications*; Packt Publishing Ltd: Birmingham, UK, 2017.
22. André, E.; Le Breton, N.; Lemesle, A.; Roux, L.; Gouaillard, A. Comparative Study of WebRTC Open Source SFUs for Video Conferencing. In Proceedings of the Principles, Systems and Applications of IP Telecommunications (IPTComm), Chicago, IL, USA, 16–18 October 2018; pp. 1–8.
23. García, B.; Gallego, M.; Gortázar, F.; Bertolino, A. Understanding and estimating quality of experience in WebRTC applications. *Computing* **2018**, 1–23. [CrossRef]
24. Garcia, B.; Lopez-Fernandez, L.; Gallego, M.; Gortazar, F. Kurento: the Swiss army knife of WebRTC media servers. *IEEE Commun. Stand. Mag.* **2017**, *1*, 44–51. [CrossRef]

25. Thakur, N.; Devi, S. A new method for color image quality assessment. *Int. J. Comput. Appl.* **2011**, *15*, 10–17. [CrossRef]
26. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef] [PubMed]
27. Wang, Z.; Simoncelli, E.P.; Bovik, A.C. Multiscale structural similarity for image quality assessment. In Proceedings of the Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, Pacific Grove, CA, USA, 9–12 November 2003.
28. Huynh-Thu, Q.; Ghanbari, M. Scope of validity of PSNR in image/video quality assessment. *Electron. Lett.* **2008**, *44*, 800–801. [CrossRef]
29. Egiazarian, K.; Astola, J.; Ponomarenko, N.; Lukin, V.; Battisti, F.; Carli, M. New full-reference quality metrics based on HVS. In Proceedings of the Second International Workshop on Video Processing and Quality Metrics, Scottsdale, AZ, USA, 22–24 January 2006.
30. Ponomarenko, N.; Silvestri, F.; Egiazarian, K.; Carli, M.; Astola, J.; Lukin, V. On between-coefficient contrast masking of DCT basis functions. In Proceedings of the Third International Workshop on Video Processing and Quality Metrics, Scottsdale, AZ, USA, 25–26 January 2007.
31. Alvarez, A.; Cabrero, S.; Pañeda, X.G.; Garcia, R.; Melendi, D.; Orea, R. A flexible QoE framework for video streaming services. In Proceedings of the 2011 IEEE GLOBECOM Workshops (GC Wkshps), Houston, TX, USA, 5–9 December 2011; pp. 1226–1230.
32. Yeager, K. *SPSS Tutorials: Pearson Correlation*; University Libraries: Geneva, Switzerland, 2019.
33. Amrhein, V.; Greenland, S.; McShane, B. Scientists Rise up Against Statistical Significance. 2019. Available online: https://www.nature.com/articles/d41586-019-00857-9 (accessed on 1 July 2019).
34. Wasserstein, R.L.; Lazar, N.A. The ASA's statement on p-values: context, process, and purpose. *Am. Stat.* **2016**, *70*, 129–133. [CrossRef]
35. Simoncelli, E.P.; Freeman, W.T. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In Proceedings of the International Conference on Image Processing, Washington, DC, USA, 23–26 October 1995; pp. 444–447.