Poster: Understanding YouTube QoE in Cellular Networks with YoMoApp - a QoE Monitoring Tool for YouTube Mobile

Florian Wamser (1), Michael Seufert (1), Pedro Casas (2)
Ralf Irmer (3), Phuoc Tran-Gia (1), Raimund Schatz (2)
(1) University of Würzburg, (2) FTW Vienna, (3) Vodafone Research and Development
{florian.wamser|surname}@informatik.uni-wuerzburg.de; {surname}@ftw.at; ralf.irmer@vodafone.com

ABSTRACT

The performance of YouTube in cellular networks is crucial to network operators, who try to find a trade-off between cost-efficient handling of the huge traffic amounts and high perceived end-user Quality of Experience (QoE). In this paper we present YoMoApp (YouTube Performance Monitoring Application), an Android application which passively monitors key performance indicators (KPIs) of YouTube adaptive video streaming on end-user smartphones. The monitored KPIs (i.e., player state/events, re-buffering, and video quality levels) can be used to analyze the QoE of mobile YouTube video sessions. YoMoApp is a valuable tool to assess the performance of cellular networks with respect to YouTube traffic, as well as to develop optimizations and QoE models for mobile HTTP adaptive streaming. We try YoMoApp through real subjective QoE lab tests showing that the tool is accurate to capture the experience of end-users watching YouTube on smartphones.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Miscellaneous; C.4 [**Performance of Systems**]: Measurement Techniques

Keywords

QoE; Smartphones; Subjective Lab Tests; Mobile Apps

1. MOTIVATION & METHODOLOGY

YouTube is one of the most popular services in today's Internet. It has more than 1 billion users and every day people watch hundreds of millions of hours of YouTube videos. Half of those YouTube views are done on mobile devices [1]. On the one hand, operators want to handle the huge amount of video traffic as efficiently as possible (high revenue per bit), on the other hand, they want to deliver a high Quality of Experience (QoE) to satisfy their customers. Therefore, it is paramount to cellular operators to understand the performance of their networks wrt YouTube traffic.

Copyright is held by the owner/author(s). This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in:

DOI: http://dx.doi.org/10.1145/2789168.2795176
MobiCom '15, September 7–11, 2015, Paris, France.
ACM. ISBN 978-1-4503-3619-2/15/09

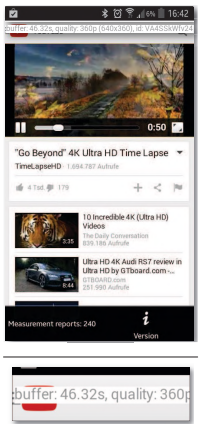
To measure the network performance in terms of QoE, different approaches are proposed in literature. First, operators can conduct subjective studies and ask the users about the perceived service quality. Subjective studies can directly assess the QoE in terms of mean opinion scores (MOS), but their design and execution is complex and costly. Second, operators can perform active measurements with client devices to probe the network. However, these samples can only provide a rough estimation of the network performance, as they only cover the geo-temporal-span of the active measurements. Finally, passive measurements can be conducted either in the network or at the client device. In-network measurements (e.g., YouQMON [2]) have a more global scope and cover more users, but the resulting QoE has to be estimated from traffic characteristics and/or deep-packet inspection. Moreover, the recent trend towards HTTPS (e.g., YouTube, Vimeo) is about to inhibit its applicability. As a consequence, client side passive measurements are becoming a practical means to QoE-based network performance analysis, with the paramount advantage of having a vantage point next to the end-customer (i.e., on its device).

In this paper we present YoMoApp (YouTube Performance Monitoring Application), a passive measurement application for client side monitoring of YouTube video streaming on mobile Android devices. The application uses the YouTube mobile website and the YouTube HTML5 API to exactly replicate the well-known YouTube service, which employs HTTP adaptive streaming (HAS) technology based on resolution adaptation. However, it additionally monitors and stores multiple Key Performance Indicators (KPIs) of the video streaming via the YouTube API (i.e., player state/events, buffer, and video quality level), which allow to analyze the QoE of adaptive video streaming sessions.

YoMoApp is an accurate and highly valuable tool for passive measurement of YouTube performance in cellular networks, which is becoming highly needed and popular among cellular operators. In addition, the tool can be used to develop novel QoE models for HAS in mobile devices, enabling multiple QoE-based monitoring applications for YouTube traffic, such as dynamic traffic engineering [3], troubleshooting, load balancing and caching, and many more.

2. THE YOMOAPP TOOL

The goal is to monitor application layer KPIs of YouTube that have a high correlation with the actual QoE of mobile app users. According to [4], the main influence parameters of the YouTube QoE are *stallings* and *video quality*. To obtain these parameters, we monitor the buffer filling levels and the resolution of the YouTube videos.



Name	Description
buffered	List of time ranges of the media content that have been buffered
height/ width	Height and width of the video's display area in CSS pixels
played	Object indicating all the ranges of the video that have been played
currentTime	Current video playtime
youtubeId	Object indicating YouTube identifier of the video content
totalVideo-Frames	Total number of frames that would have been displayed if no frames are dropped
dropped-Video-Frames	Total number of frames dropped predecode or dropped because the frame missed its display deadline
corrupted-Video-Frames	Total number of corrupted frames that have been detected
Timestamp	Timestamp of the data query
Name	Pre-defined device name
Session timestamp	Session timestamp to identify the YouTube session.

Figure 1: Screenshot of the app and selected parameters from the HTML5 ⟨*video*⟩ object [6], Media Source Extensions [7], and device, which can be investigated by the app.

YoMo works as follows. The original YouTube app is fully replicated in functionality and design, see Fig. 1. To this end, existing libraries from YouTube are used that are available for YouTube developers. An Android web view browser element is embedded for the YouTube video playback, such that HTML5 video playback is possible, including adaptive streaming according to the MPEG Dynamic Adaptive Streaming over HTTP (DASH) approach [5] of YouTube. Additional functions are added, which ultimately perform the monitoring of the application parameters in the newly created app. The monitoring is done at runtime via JavaScript, which queries the embedded HTML5 ⟨*video*⟩ object. In Fig. 1, the utilized parameters are listed. Note that the obtained parameters can be displayed in YoMoApp for validation (see Fig. 1), but are usually hidden. The measurement methodology in the app follows 4 steps:

Step #1: *HTML Detection* - the YouTube web page is detected and the HTML video player element is identified. YouTube consists of many web pages and elements. Thus, the name and id of the relevant video elements must be determined. The detection is done via the injection of JavaScript code to the running Android WebView browser element of the app. The JavaScript code analyzes the HTML Document Object Model (DOM) tree for the ⟨*video*⟩ element.

Step #2: *Request the Data* - parameters such as current playtime or current available video content are retrieved. This is realized by injecting JavaScript code that requests the application parameters from the detected video player element in Step #1 every second.

Step #3: *Calculation* - from the retrieved parameters, the buffer filling level can be calculated. The parameter *buffered* is subtracted from *currentTime*, which results in the current buffer level. Likewise, the video resolution is obtained based on *height* and *width* of the video player.

Step #4: *Data Transfer* - finally, data is transferred to an external database located in the Internet. Data is compressed and stored as structured objects, and can be transmitted at different time instances: a) when closing the app, b) manually by the user, c) at predetermined intervals, etc. Data is also locally cached, since the connection from the smartphone to the Internet server is often not reliable.

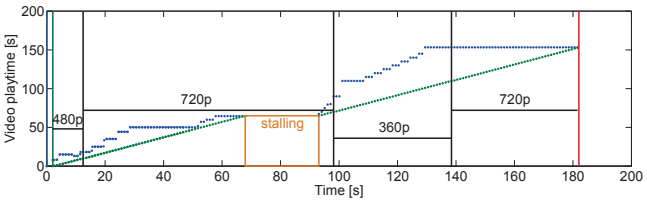


Figure 2: Illustration of monitored parameters for an exemplary video streaming: current video playtime (green) and buffered video playtime (blue). Events are displayed as vertical lines: page load (blue), playback start (green), quality switch (black), playback end (red). Stalling is depicted as orange box. The horizontal black lines indicate the currently played out video quality/resolution.

Fig. 2 shows the data of an exemplary run in their processed form. Postprocessing of the data is recommended because the usage of JavaScript can sometimes introduce inconsistencies and obvious errors, e.g., missed player events, non-equidistant data queries, missing/incorrect values. However, after removing unusable runs and the recovery of missing events (e.g., stalling events can be estimated from buffer filling level), YoMoApp proved to perform accurate measurements on a sufficiently small time scale (∼1 s).

3. YOMOAPP IN ACTION

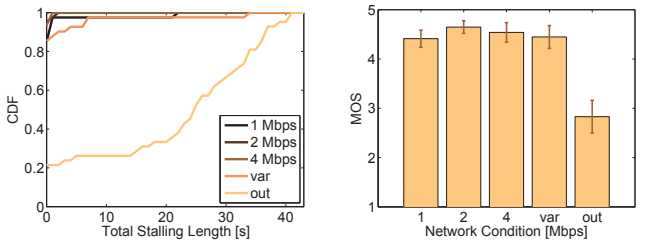
To demonstrate the applicability of YoMoApp to analyze the QoE of YouTube in mobile devices, we conducted a subjective study in which participants watched YouTube videos in smartphones instrumented with the YoMoApp tool and rated the resulting watching experience. To induce different QoE levels, the downlink traffic from YouTube servers to the mobile devices was throttled through an intermediate instrumented router imposing different bandwidth profiles.

3.1 Subjective Study Overview

The subjective study was performed in a dedicated lab for subjective analysis, compliant with the recommendations provided by the QoE subjective studies standards [8]. 52 people participated in the study (29 female, 23 male), the average age was 32 years old, with 40 participants being less than 30 years old.

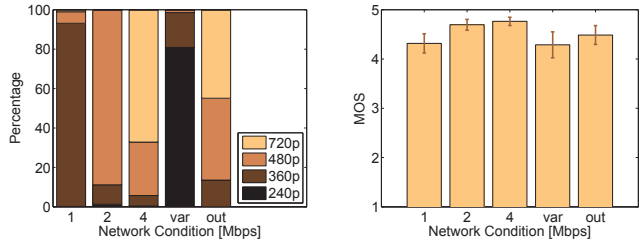
Android smartphone devices (Samsung Galaxy S4, OS Android 4.4 KitKat) were used in the tests. Devices were connected to the Internet through independent WiFi access points. The downlink traffic was routed through a modified version of the well-known NetEm network emulator to impose different access network bandwidth profiles to the video streaming. Three different bandwidth profiles were used: (i) constant downlink bandwidth: 1 Mbps, 2 Mbps, and 4 Mbps; (ii) fluctuating downlink with variable bandwidth ("var"): downlink bandwidth is periodically increased from 1 Mbps to 3 Mbps for periods of 5 seconds, 3 times per minute. The resulting average downlink bandwidth is 1.5 Mbps; (iii) downlink bandwidth outages ("out"): downlink bandwidth drops from 4 Mbps to 0 Mbps for periods of 10 seconds, twice per minute. In this case, the resulting average downlink bandwidth is 2.7 Mbps.

The specific task imposed to participants was to watch two minute long YouTube adaptive streaming videos using YoMoApp for the five different downlink bandwidth condi-



(a) Total stalling time. (b) Impacts of stallings on QoE.

Figure 3: Monitoring of stallings and their impact on QoE.



(a) Distribution of quality levels. (b) Video image QoE.

Figure 4: Monitoring of video quality switches and the resulting image QoE.

tions, and rate the resulting watching experience afterwards (i.e., overall experience, impact of initial playback delay and stalling, video image quality, etc.) on continuous ACR MOS scales [8]. The tested videos are available as 4K ultra-HD videos (i.e., 2160p), but the maximum video quality observed in the tests was HD (i.e., 720p) due to the devices' display capabilities (i.e., screen size and resolution).

3.2 Analysis of Stallings

Stalling, i.e., the interruption of playback due to a playout buffer under-run, is considered the worst quality degradation of video streaming [4]. Stalling occurs when the available network bandwidth is lower than the video bit rate. The playout buffer is filled slower by the download than it is emptied by the playback, which will eventually lead to stalling. During a stalling event, the playback is paused until the buffer is filled with a sufficient amount of video data to continue the playback. Authors in [4] found an exponential relationship between stalling parameters and MOS and that users tolerated at most one stalling event of up to three seconds length. Thus, it is important to monitor the stalling during the video playback.

Fig. 3a shows the distribution of the total stalling time for each tested condition. Almost no stalling occurs for the constant bandwidth conditions. For 4 Mbps, 93.48% of the streaming sessions have no stalling, the remaining sessions have less than 1 s of stalling. A maximum of 2 s of stalling was measured for the 2 Mbps condition, with 95.12% of the sessions showing no stalling at all. For the 1 Mbps condition, 85.00% of the streaming sessions do not include stalling. However, one outlier with 22 s stalling was observed. For the variable condition ("var"), stalling occurs in 14.63% of the conditions ranging up to a total stalling time of 34 s. 78.57% of the outage condition ("out") streams contained stalling. The average total stalling time in this condition is 25 s, and the maximum total stalling time is 41 s.

In Fig. 3b, the MOS and 95% confidence intervals of the QoE ratings are presented. Participants were asked to which extend they perceived the interruptions caused by stalling as disturbing. The MOS values for the disturbance of stalling range from 1 (very disturbing) to 5 (not disturbing at all). For 1 Mbps, 2 Mbps, 4 Mbps and variable condition stalling is not considered as disturbing, having MOS scores of at least 4.4. This corresponds to the measured total stalling time, which indicated very short stalling events, if any at all. Only for the outage condition, stalling is perceived as disturbing, having a MOS of 2.83. Again, this corresponds to the frequently long total stalling times measured by YoMoApp.

3.3 Analysis of Quality Switches

HTTP adaptive streaming trades off stalling for video quality. It adapts the downloaded video quality (i.e., video bit rate) to the currently available network conditions, thereby avoiding stalling to the greatest possible extend. In YouTube, this is implemented by adaptively changing the resolution of the streamed video. However, quality switches, i.e., the change of the video bit rate, occur during streaming and can be perceived by the users. In the following, the monitored quality of the video streaming will be investigated.

Fig. 4a shows the percentage of time on each quality level per condition, i.e., the percentage of time which each video resolution was played out during the streaming. In the 1 Mbps condition, all available resolutions were used with the following overall shares: 240p (0.23%), 360p (92.89%), 480p (5.85%), 720p (1.02%). The 2 Mbps condition shows a larger percentage of 480p quality (88.84%), and in the 4 Mbps condition a significant share of 720p quality (67.15%) can be played out. Additionally, the outage condition has similar quality shares to the 4 Mbps conditions, which is not surprising considering that it is a 4 Mbps on/off pattern. The variable condition contains a large percentage of the lowest resolution (240p, 80.86%), which indicates that the YouTube adaptation is very conservative when the network conditions fluctuate considerably.

Fig. 4b shows how the image quality of each condition was rated by the participants (MOS and 95% confidence intervals). It can be observed that the image quality is rated good for all conditions, having a MOS of at least 4.17. This means that resolution adaptation does not have a big impact on the subjectively perceived image quality, which is most probably linked to the small display size of smartphones. Nevertheless, the increasing levels of quality played out for 1 Mbps, 2 Mbps, and 4 Mbps correspond to the increasing image quality ratings by the participants.

4. REFERENCES

[1] [Online]. https://www.youtube.com/yt/press/statistics.html
[2] P. Casas, M. Seufert, and R. Schatz, "YOUQMON: A System for On-line Monitoring of YouTube QoE in Operational 3G Networks," *ACM SIGMETRICS PER*, vol. 41, 2013.
[3] F. Wamser, D. Hock, M. Seufert, B. Staehle, R. Pries, and P. Tran-Gia, "Using Buffered Playtime for QoE-Oriented Resource Management of YouTube Video Streaming," *IEEE Trans. ETT*, vol. 24, 2013.
[4] T. Hoßfeld, R. Schatz, M. Seufert, M. Hirth, T. Zinner, and P. Tran-Gia, "Quantification of YouTube QoE via Crowdsourcing," in *MQoE*, 2011.
[5] ISO/IEC, "23009-1:2012 Information Technology – Dynamic Adaptive Streaming over HTTP (DASH) – Part 1: Media Presentation Description and Segment Formats," 2012.
[6] [Online]. Available: http://developer.mozilla.org/en-US/docs/Web/HTML/Element/video
[7] [Online]. Available: http://dvcs.w3.org/hg/html-media/raw-file/tip/media-source/media-source.html
[8] ITU-T, "Recommendation P.910: Subjective Video Quality Assessment Methods for Multimedia Applications," 2008.