Dynamic Optimization of the Quality of Experience during Mobile Video Watching

Toon De Pessemier, Luc Martens, Wout Joseph
iMinds - Ghent University, Dept. of Information Technology
G. Crommenlaan 8 box 201, B-9050 Ghent, Belgium
Tel: +32 9 33 14908, Fax:+32 9 33 14899
Email: {toon.depessemier, luc.martens, wout.joseph}@intec.ugent.be

***Abstract*—Mobile video consumption through streaming is becoming increasingly popular. The video parameters for an optimal quality are often automatically determined based on device and network conditions. Current mobile video services typically decide on these parameters before starting the video streaming and stick to these parameters during video playback. However in a mobile environment, conditions may change significantly during video playback. Therefore, this paper proposes a dynamic optimization of the quality taking into account real-time data regarding network, device, and user movement during video playback. The optimization method is able to change the video quality level during playback if changing conditions require this. Through a user test, the dynamic optimization is compared with a traditional, static, quality optimization method. The results showed that our optimization can improve the perceived playback and video quality, especially under varying network conditions.**

Keywords*—Subjective evaluation techniques, Mobile TV, Quality of experience.

I. Introduction

Mobile video traffic exceeded 50 percent of total mobile data traffic for the first time in 2012. Moreover, world's mobile data traffic coming from video will increase 13-fold between 2014 and 2019, accounting for 72 percent of total mobile data traffic by 2019 [1]. As mobile network connection speeds increase, the average bit rate of video accessed through mobile devices will increase and the popularity of high-definition video will continue to grow.

The Mobile Data Traffic Forecast of Cisco states that the proportion of streamed video content, as compared to side-loaded content, is also expected to increase with average mobile network connection speed. This is emphasized by the popularity of streaming video services such as Netflix[1] and Hulu[2]. This biggest advantage of streaming is that it enables playing the video before the entire file has been transmitted.

Streaming video is a media format with strict network requirements for different purposes (Video-on-demand streaming, video conferencing, online interactive gaming, or mobile TV) [2]. Offering a good experience to users remains challenging, and given the dependence on several influencing factors, this especially holds true in the context of mobile video applications.

[1] www.netflix.com
[2] http://www.hulu.com/

Traditionally, network operators and service providers used to pay close attention to the Quality of Service (QoS), where the emphasis was on delivering a fluent service. QoS is defined by the ITU-T as "the collective effect of service performance" [3]. The QoS is generally assessed by objectively-measured video quality metrics. These quality metrics play a crucial role in meeting the promised QoS and in improving the obtained video quality at the receiver side [4].

Although useful, these objective quality metrics only address the perceived quality of a video session partly, since these metrics do not correlate perfectly with the human perception and not all visual distortions are always taken into account. Moreover, audible distortions and playback failures, which have an influence on the user experience, are often ignored in these objective quality metrics.

After years of research on the QoS, researchers have shifted their focus towards the end user and the quality of interaction between user and service also known as the Quality of Experience (QoE). Although QoE and QoS are closely related, subtle but important differences shift the emphasis from 'quantitatively measuring the service performance' (QoS) [3] to 'how a service is really perceived by the user' (QoE) [5]. In contrast to QoE, QoS measurements neglect the context of the user (e.g., watching video on a moving train), the coupled quality expectations (which may be lower on the train), and the influence of the context on how the service is perceived [6] (e.g., users may pay less attention to the video quality on the train).

To get insights into the QoE and assess how a video is really perceived by a viewer, subjective quality measurements are often performed. These subjective quality assessments are usually conducted by asking human subjects to rate the perceived visual quality of the displayed media according to a provided quality scale [7]. Also for subjective quality assessments, different metrics exist. In the domain of communication, MOS testing (Mean Opinion Score) is predominantly used as a subjective measure of voice quality [8]. To obtain a MOS score, i.e. a numerical indication of the perceived quality, a number of test subjects (typically 15-30) [9] are asked to evaluate quality parameters by means of standardized scales using labels ranging from 'Excellent' to 'Bad' as defined by the ITU-T [3]. The average rating over all viewers for a given clip is also known as the Mean Opinion Score (MOS). Also for video content, subjective experiments using the MOS score are a reference methodology for obtaining quality ratings [9].

The aim of this study is to improve the QoE by using an agent that anticipates changing conditions (such as the environmental context and network) and dynamically adapts the parameters of a streaming video during playback. The remainder of this paper is organized as follows. Section II describes the current approach used by YouTube and many other similar service. Our dynamic approach is explained in detail in Section III. Section IV provides some important implementation details to get the dynamic optimization working on mobile devices. Section V describes the setup of our user test, insights regarding the users' interactions with the video system, and results of the qualitative user feedback. Section VI draws conclusions and points to interesting future work.

II. YOUTUBE'S STATIC OPTIMIZATION OF THE QUALITY

The currently existing mechanisms to adapt the video parameters to the technical parameters of the device, context, and network are based on a static optimization of the video quality. In case of streaming video from YouTube, the video quality (defined by video parameters such as the resolution) is determined before the video starts based on only 2 parameters: the network and the device resolution. During video playback, the video parameters will not be reconsidered if network conditions change, which refers to the static characteristic of this optimization. Since version 5.7, users also have the possibility to specify the desired video quality; but also in this way, changing conditions during playback are not automatically taken into account.

As a result, the static optimization might be suboptimal in the case of video watching with rapidly changing network conditions. E.g., watching video in a moving train might induce changing network conditions due to variable speeds, environmental factors, such as tunnels, and varying distances to cell towers. Degraded network conditions may cause empty video buffers, thereby causing video choppiness. Interruptions in the video playback due to rebuffering lead to user frustration and should be avoided if alternative video configurations exists [10].

III. DYNAMIC WATERFALL OPTIMIZATION OF THE QUALITY

To address the problems of a static optimization, we propose a dynamic waterfall mechanism to determine the video parameters for an optimal quality, taking into account the device characteristics, the user context, and the real-time network conditions, which are continuously monitored as a background process of the device. The determination of the optimal video parameters is addressed as a waterfall process in which the different parameter configurations flow (as visualized in Figure 1); unfeasible configurations are rejected, and an optimal configuration is selected. This waterfall process is a continuous monitoring and optimization process, running during video playback, thereby enabling to optimize video parameters in real-time taking into account changing conditions. The monitoring process follows the observer-design pattern: the objects keeping track of the context and network parameters are observed by the waterfall optimization, and every change of the parameters will be reported in order to reconsider the chosen video parameters.

The waterfall optimization consists of four subsequent phases. In the first phase, the *network type* is considered, making a distinction between WiFi networks, broadcasting channels, and mobile data networks. For mobile data networks, an additional distinction is made when data roaming is active, which may induce additional costs for data traffic. If this is the case, the video parameters are adjusted to reduce data traffic. Although the network type provides no hard guarantee on the available bandwidth, it gives some insights into the possible video parameter configurations.

In case of a mobile data network, the *movement* of the user (as measured by the gyroscope and GPS of the device) is considered in the second phase. If the user is moving at a high speed (e.g., traveling by car, train, or bus), the risk of network problems and a varying network quality increases, thereby inducing empty video buffers and the coupled playback interruptions. To anticipate these varying network conditions, the video parameters are chosen in such a way that the network requirements are less stringent compared to the case of a stationary user. More specifically, the optimization is based on the worst network condition as observed during the past time window of a predefined length. Because a stable WiFi connection may be available inside the train or bus, parameter adjustment based on movement is only applied in case of mobile data networks.

In the third phase, video parameters are further optimized based on the *"goodput"*, which is often lower than the network throughput. Goodput is the throughput on application level, i.e. the number of useful information bits delivered by the network excluding protocol overhead bits as well as retransmitted data packets [11]. The goodput is estimated by measuring the time required to download/upload a file of a known size to/from the mobile device. This file can be dummy, merely for measuring goodput, or in case of a video application, it can be a thumbnail of the video. In the waterfall optimization, goodput is estimated before video playback, as well as at certain moments during playback, and during interactions such as rewind or fast forward of the video to obtain the current value. The goodput thresholds, mapping video parameters such as resolution on a minimum required goodput, are adopted from YouTube [12]. YouTube defines 3 capacity levels: High Definition (HD), Standard Definition (SD), and Low Definition (LD), each characterized by a threshold value for the goodput (> 2.5 Mbps for 720p resolution (HD), between 0.7 and 2.5 Mbps for 360p resolution (SD), < 0.7 Mbps is not sufficient to sustain SD video playback (LD)).

In the fourth phase, the *device characteristics* are taken into account. If the video decoding process is too processor intensive so that the video does not play smoothly, the video parameters are decreased thereby reducing the burden on the processor. By monitoring the processor load during playback, video interruptions due to a processor overload are avoided. In addition to the processor load, also the screen resolution is an important device parameter. If the video resolution is higher than the screen resolution, the video will be downscaled automatically to the screen resolution by the video player. As a result, an obvious optimization is to stream only videos with a resolution that is smaller than or equal to the screen resolution. This will save processing power as well as network bandwidth without sacrificing quality. Also the remaining

battery power capacity is an important device characteristic. The video parameters have a significant influence on the power consumption on mobile devices [13]. To prolong the battery lifetime, the video parameters and the coupled quality level are decreased in case of a battery power capacity below a threshold. This action can enable the user to finish video playback before the battery runs out.

This waterfall optimization, running before and during the video playback, continuously adjusts video parameters to optimize the quality. This enables the QoE agent to respond to changing conditions. If a new parameter configuration is determined as the optimal one, a new video stream is set up which continuous where the previous video stream was stopped. In case of such a video quality change, video playback is not interrupted; users can only notice a small glitch in the video. To avoid quality fluctuations, each parameter adjustment has to be confirmed by subsequent measurements of the changed conditions. This prevents the alternation between two quality configurations, and the coupled switching of the video stream, in case of a borderline condition. For every quality change, the underlying reason is logged for future analysis.

For testing purposes (Section V), YouTube was used as a content source because of its enormous video offer. However, YouTube does not offer all videos in each possible quality level. If the desired quality level is not available on YouTube, the closest quality level just below the desired level is chosen. The QoE agent with waterfall optimization is implemented as an Android service that continuously runs in the background. For the video, a foreground service is used while the video is playing in order to guarantee the continuity of the playback.

In addition, the QoE agent has the following features to adapt the video playback to the environment.

- The *brightness* of the screen is adjusted to the daylight. More direct light on the screen is compensated by a brighter screen. On some device, this adjustment is implemented by default. On other devices, the QoE agent provides this functionality.
- The video player app can be sent to the *background* without interrupting the playback, enabling to listen to music while using other apps. This is not possible with the standard YouTube app, being a source of frustration for many users.
- The proximity sensor detects if another 'object' is close. E.g., if the mobile device is stored in the user's pocket during playback, the screen is automatically locked.
- The *volume* of the player is automatically adjusted to ambient noise measured by the microphone of the device.
- New users receive a short *tutorial* when they use the application for the first time.
- A timer monitors any *timeouts* during application loading or video playback. Such a timeout triggers a short message for the end user reporting the unforeseen delay or potential problem with the internet connection. In the meantime, the task is reloaded until successful, a maximum number of retries is reached, or the user decides to close the application. The timer goes off after a specified time period, which is adapted to the measured goodput. On slower networks, a longer waiting period is tolerated.

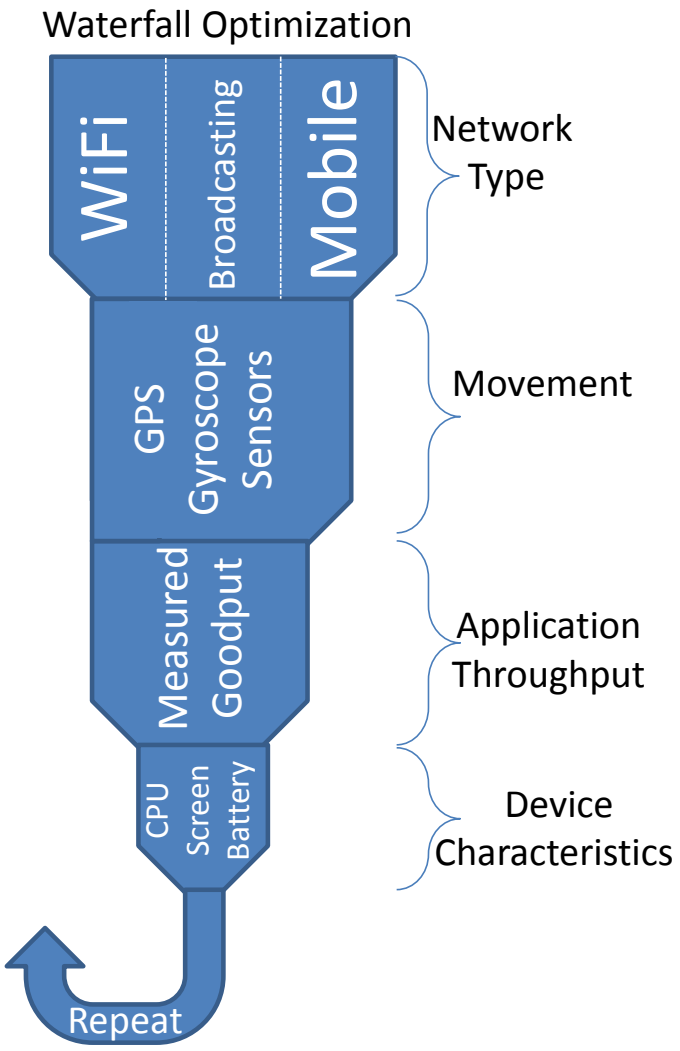

Figure 1. Graphical representation of the waterfall optimization.

IV. Implementation Details

The Android application, containing the video player and the QoE agent, is implemented as a hybrid solution of a native container that encloses a web application. The native container enables the integration of sensor data, while the web application comes with the advantage of platform independence. The web application embeds a video player in an HTML page and controls the player using Google's IFrame API and JavaScript API.

The traditional YouTube API of Android was not used in the developed Android app, since this API did not allow to set the desired quality of the YouTube video and made its own choice regarding the quality depending on the device and network. This limitation should be removed in more recent versions of the YouTube API (June 2014) by allowing users to select the streaming quality [14].

The following Android APIs are used for the development of the application.

- Google's **IFrame API**[3] is used to embed the video player into an Android WebView and control the player through JavaScript. The IFrame API is chosen because it is supported by a wider range of devices compared to alternatives such as Flash objects. For the web page development, the Phonegap framework[4] is used, which ensures an easy migration to other mobile platforms. The source code is written in HTML5, CSS3, and JavaScript and can be compiled for the desired mobile platform, while using native features of this platform is still possible through specific plug-in classes.
- Google's **Data API**[5] is used to query YouTube data, such as a list of video resources. Server responses are in JSON format, which makes it easy to process different attributes such as title, desciption, channel, and thumbnail.
- **ShowcaseView** is an open source library available on GitHub[6] that is used for showing a short tutorial to new users. A semi-transparent overlay is used to indicate and explain important components of the user interface. This will teach users how to interact with the application and the functionality of the various features. In order to show a more detailed tutorial consisting of multiple overlays, we extended ShowcaseView with a ShowcaseManager which enables to display multiple overlays one after the other. If an overlay is closed by the user, the next is automatically shown, to guide the user through the application until the tutorial is finished.
- **Crashlytics**[7] is a library for tracking errors that can easily be integrated into an application without interfering with normal user interaction. Crashlytics enables to follow up crashes and errors of mobile applications after distributing them through the app store in order to resolve problems of active users. If the application crashes, a full report will be send to the Crashlytics servers. The application developers can consult these reports and get an overview of all problems through the web interface of Crashlytics. For each problem, the status can be specified to keep track of the progress of a solution. Crashlytics can be used as a plug-in for an Integrated Development Environment (IDE), so that it can automatically make the necessary additions to the source code for error logging.
- **JSch**[8] is a Java library that allows to make an SSH connection from Android and supports port forwarding. JSch is used to make a connection to a database server and transfer data of the video sessions from the client devices to a central storage server. A connection over SSH is used to overcome port restrictions of the central storage server.
- **JQuery Mobile**[9] is a HTML5-based user interface system designed to make responsive web sites and apps that are accessible on all smart-phone, tablet, and desktop devices. This open source library that is supported by various mobile platforms is used in this application for the interaction with the user.

[3]https://developers.google.com/youtube/iframe_api_reference
[4]http://phonegap.com
[5]https://developers.google.com/youtube/v3/
[6]https://github.com/amlcurran/ShowcaseView
[7]https://try.crashlytics.com
[8]http://www.jcraft.com/jsch/

V. User Testing

A. Setup

In order to compare video quality selection and playback using the waterfall optimization with YouTube's traditional video selection and playback, a user test was performed. In this study, 15 users participated by watching videos on their own mobile device. They were stimulated to watch these videos in various contextual situations: different locations, times, networks, etc., in order to exploit the potential of the waterfall optimization. Because the main purpose of the study was to gather qualitative feedback regarding users' experience with the waterfall optimization rather than a comprehensive statistical analysis of the two quality selection methods, we do not consider the limited number of participants as an issue.

To make a user evaluation of the two quality selection methods feasible, a video player application, very similar to the YouTube app, was developed. This application enabled test users to search for, browse, and watch any YouTube video of their choice while it also enabled us to experiment with the waterfall optimization as an alternative quality selection method.

During the start up of the video application, users are randomly assigned to one of two test groups: the group using YouTube's traditional quality selection or the group using the waterfall optimization. To avoid any biases, users were not informed about these two different video quality selection methods.

Compared to the traditional YouTube app, our video application contains some additional features to gather feedback from the test users. The user interface, which is shown in Figure 2, contains a quality feedback option, visualized as red, semi-transparent overlay button, enabling the users to report technical problems during video playback. In the application's tutorial, users get clear instructions about the usage of this button. They are encouraged to push this button in case they experience the quality as unacceptable, or in case of playback issues, such as video interruptions due to rebufferings or too long loading times. Users can move and remove this button freely if it is in an inappropriate position for video watching. By logging all technical parameters of the video playback when this button is pushed, detailed information is available to draw conclusions regarding the underlying reason for the user's dissatisfaction.

After the video playback, users are asked to answer a short questionnaire consisting of a few multiple choice and open questions to assess their experience with the playback quality. The answers to the questionnaire are firstly stored

[9]http://jquerymobile.com

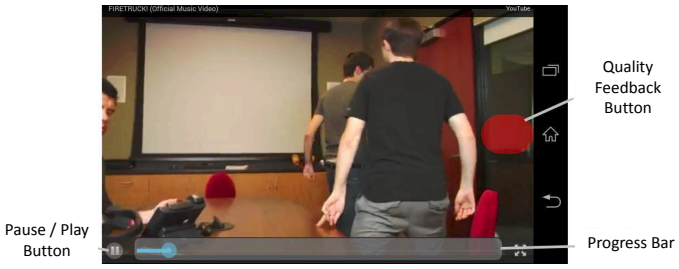

Figure 2. The user interface of the video application with the quality feedback button.

locally on the user's mobile device in an SQLite database. This local storage ensures that in case of technical problems, such as a network disconnection, technical data are logged for future analysis. As soon as a stable network connection is available, the application will transfer the technical data to a remote MySQL service in order to make the data available for further investigation and comparison with similar cases. If necessary, various attempts are being made to transfer the data until successful.

B. Users' video interactions

If video playback is stopped early by the user before the end of the video, the user receives a question regarding the reason for this. The answers revealed that is most cases the reason is non-technical: users are not interested to watch the full video or do not have time.

If the reason of the early stop is because of technical issues, users could specify their *complaint*. The waterfall optimization and YouTube's video selection method both received one complaint regarding a low video quality. So, users do not experience the waterfall optimization as different in terms of delivered video quality.

In terms of loading time (i.e. the time between selecting a video and the moment when the video starts playing), YouTube's video selection method received four complaints regarding an excessively long loading time in comparison with two complaints for the waterfall optimization. This indicates that the waterfall optimization makes a better assessment of the optimal video quality level. E.g., in case of a user moving at a high speed, the video parameters are adjusted to the network conditions to ensure a fluent video playback, as explained in Section III. In this case, lowering the video parameters will improve the loading time. If the network conditions improve during video playback, the waterfall optimization can increase the quality level. In contrast, YouTube's video selection method is static and assesses the optimal quality level only one time, just before video playback. Changing network conditions during playback are not taken into account by YouTube. The logging data of the technical parameters illustrate this. E.g., occasionally a low-definition quality was initially chosen by YouTube while the network goodput was > 5 Mbps during playback. In this case, increasing the quality level to high-definition was possible to improve the user experience and the waterfall optimization has an advantage compared to YouTubes method.

In addition, the static optimization was often the reason of interactions with the quality feedback button. The logging data showed that in case of variable network conditions, YouTube's choice of high-resolution quality may cause playback interruptions. During the user test, several cases have been observed in which YouTube selected high-resolution quality based on a high network goodput before playback, but in which the goodput dropped to 2 Mbps or lower during playback. The static optimization is not able to respond to the insufficient goodput for high-resolution video. As a result, rebufferings during video playback are inevitable, resulting in playback interruptions. In contrast, the waterfall optimization can respond to changing network conditions by switching to a lower quality level thereby preventing playback interruptions due to rebufferings.

C. Users' quality feedback

After every video, test users received a short *questionnaire* to evaluate their experience with the video playback, as explained in Section V. The first question was: *"How do you evaluate the playback quality (e.g., loading time, playback interruptions)? "* Through this subjective question, users could specify how they perceived the video playback on a 5-point rating scale. Although loading time and interruptions can be measured objectively, the user's opinion is important since not all playback interruptions are perceived as equally disturbing [15], [10].

Table I lists the mean value (MOS) and the standard deviation of the obtained ratings. The results show that the waterfall optimization obtained a higher mean rating (4.4) than YouTube's video selection method (3.8) in terms of playback quality. This can be explained by the dynamic quality selection of the waterfall process. If the network conditions deteriorate during playback, the waterfall method will lower the quality level thereby minimizing playback interruptions due to rebufferings. In contrast, since YouTube does not adjust the quality level during playback, interruptions are more likely in case of deteriorating conditions. Logged data confirmed that especially in cases with a dropping goodput, the Waterfall optimization scores better than YouTube's method. These cases are also the reason why the playback quality ratings for YouTube have a higher standard deviation (0.94) compared to the Waterfall optimization (0.45).

The second question of the questionnaire was: *"How do you evaluate the video quality?"*. Again, the offered video quality could be evaluated objectively. But user expectations and contextual influences on the perception can only be taken into account by subjective evaluations of actual users in a realistic environment [6].

The mean value (MOS) and the standard deviation of users' rating for the video quality are listed in Table I. In terms of perceived video quality, the waterfall optimization obtained a mean score of 4.6 whereas YouTube's method scored 3.8. This improvement can be attributed to the dynamic quality selection of the waterfall optimization. In case network conditions are bad during the start of the video, but gradually improve during video playback, the waterfall optimization switches to a higher quality level if possible. In contrast, YouTube will stick to the low quality version till the end of the video. During the user test, various data samples were obtained in which YouTube maintains a sub-optimal quality during playback.

Metric	YouTube	Waterfall
Mean playback quality rating	3.8	4.4
Standard deviation playback quality rating	0.94	0.45
Mean video quality	3.8	4.6
Standard deviation video quality	1.11	0.45
Processor load	0.15	0.45 %

Table I. COMPARISON OF THE WATERFALL OPTIMIZATION AND YOUTUBE'S METHOD

As a result, the standard deviation of the video quality is higher for YouTube's method (1.11) compared to the waterfall optimization (0.45).

As shown in Table I, these improvements in perceived playback and video quality of the waterfall optimization are obtained at the price of an increased processor load (0.45% for the waterfall optimization compared to 0.15% for YouTube's method). The reason for this is two-fold. Firstly, the waterfall optimization can switch to a higher quality level if network conditions improve during playback. These high quality videos typically require more processing power during playback. Secondly, the waterfall optimization itself requires some additional processing power to estimate the optimal video quality, not only before the video, but also continuously during playback. Compared to YouTube, the waterfall optimization has some additional criteria to make a decision such as the movement of the user. The monitoring and processing of these sensor data (GPS, gyroscope) require extra processing power.

VI. CONCLUSION

This paper proposed a dynamic optimization (called the waterfall method) of the video configuration during video playback with the aim of improving the Quality of Experience for end users. YouTube has a static quality configuration by determining the video quality before video playback, and maintaining this quality level during video playback, even in the case of changing network conditions. In contrast, the waterfall optimization method can dynamically decide on the optimal quality configuration and adapt its choice to the current network conditions during playback. In addition, the waterfall optimization method bases its decisions on more criteria such as the battery level and the user's movement.

A user study, probing for qualitative feedback, showed that the waterfall optimization can improve the perceived playback and video quality, especially under varying network conditions. Although, the dynamic quality optimization comes at the expense of an increased processor load due to monitoring the conditions, it showed to have a positive influence on users' experience with mobile video watching.

In future work, we will investigate the possibility to fall back to an audio-only version of the video in case of very low application throughput. Offering the audio track of the video can be interesting for video genres such as (live) news reporting, sporting events (for the commentator's voice) or music videos. Also a text version consisting of the closed captions of the videos is an alternative, low bandwidth version of the video.

REFERENCES

[1] Cisco visual networking index, "Global mobile data traffic forecast update, 2014-2019, white paper," 2013.

[2] V. Vassiliou, P. Antoniou, I. Giannakou, and A. Pitsillides, "Requirements for the transmission of streaming video in mobile wireless networks," in *Artificial Neural Networks ICANN 2006*, ser. Lecture Notes in Computer Science, S. Kollias, A. Stafylopatis, W. Duch, and E. Oja, Eds. Springer Berlin Heidelberg, 2006, vol. 4132, pp. 528–537. [Online]. Available: http://dx.doi.org/10.1007/11840930_55

[3] International Telecommunication Union, "E.800: Terms and definitions related to quality of service and network performance including dependability," ITU-T, International Telecommunication Union, ITU-T Recommendation, 1994, updated September 2008 as Definitions of terms related to quality of service.

[4] S. Chikkerur, V. Sundaram, M. Reisslein, and L. Karam, "Objective video quality assessment methods: A classification, review, and performance comparison," *IEEE Transactions on Broadcasting*, vol. 57, no. 2, pp. 165–182, June 2011.

[5] International Telecommunication Union , "Definition of Quality of Experience (QoE)," ITU-T, International Telecommunication Union, Liaison Statement, 2007, ref.: TD 109 rev 2 (PLEN/12).

[6] T. De Pessemier, K. De Moor, W. Joseph, L. De Marez, and L. Martens, "Quantifying subjective quality evaluations for mobile video watching in a semi-living lab context," *Broadcasting, IEEE Transactions on*, vol. 58, no. 4, pp. 580–589, 2012.

[7] L. Karam, T. Ebrahimi, S. Hemami, T. Pappas, R. Safranek, Z. Wang, and A. Watson, "Introduction to the issue on visual media quality assessment," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 2, pp. 189–192, April 2009.

[8] T. de Koning, P. Veldhoven, H. Knoche, and R. Kooij, "Of mos and men: bridging the gap between objective and subjective quality measurements in mobile tv," in *Multimedia on Mobile Devices*, February 2007, pp. 65 070P–65 070P–11. [Online]. Available: http://dx.doi.org/10.1117/12.704159

[9] S. Winkler and P. Mohandas, "The evolution of video quality measurement: From psnr to hybrid metrics," *Broadcasting, IEEE Transactions on*, vol. 54, no. 3, pp. 660–668, Sept 2008.

[10] T. De Pessemier, K. De Moor, W. Joseph, L. De Marez, and L. Martens, "Quantifying the influence of rebuffering interruptions on the user's quality of experience during mobile video watching," *Broadcasting, IEEE Transactions on*, vol. 59, no. 1, pp. 47–61, March 2013.

[11] K. Pentikousis, M. Palola, M. Jurvansuu, and P. Perala, "Active goodput measurements from a public 3g/umts network," *Communications Letters, IEEE*, vol. 9, no. 9, pp. 802–804, Sep 2005.

[12] Google, "Video Quality Report - How we verify that an Internet Provider can consistently serve YouTube in HD." 2014, available at www.google.com/get/videoqualityreport/#methodology Accessed on December 2014.

[13] R. Trestian, A.-N. Moldovan, O. Ormond, and G. Muntean, "Energy consumption analysis of video streaming to android mobile devices," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, April 2012, pp. 444–452.

[14] Nguyen, C., "Youtube update allows users to select streaming quality," 2014, available at http://www.androidcentral.com/youtube-update-allows-users-select-streaming-quality.

[15] X. Tan, J. Gustafsson, and G. Heikkilä, "Perceived video streaming quality under initial buffering and rebuffering degradations," in *MESAQIN Conference (June 2006)*, vol. 90, 2006.