Hoßfeld, Tobias; Liers, Florian; Schatz, Raimund; Staehle, Barbara; Staehle, Dirk; Volkert, Thomas; Wamser, Florian:

Quality of experience management for YouTube: clouds, FoG and the AquareYoum

URN: urn:nbn:de:gbv:ilm1-2015210345

Published OpenAccess: January 2015

Original published in:

Praxis der Informationsverarbeitung und Kommunikation : PIK. - Berlin : de Gruyter (ISSN 1865-8342). - 35 (2012) 3, S. 133-143.
DOI: 10.1515/pik-2012-0024
URL: http://dx.doi.org/10.1515/pik-2012-0024
[Visited: 2015-01-20]

PIK, Vol. 35, pp. 133–143 DOI 10.1515/pik-2012-0024

Quality of Experience Management for YouTube: Clouds, FoG and the AquareYoum

Tobias Hoßfeld

University of Würzburg
Am Hubland
97074 Würzburg
Germany
hossfeld@informatik.uni-wuerzburg.de

Florian Liers

Illmenau University of Technology
Integrated Communication Systems
Group
Helmholtzplatz 5
98693 Illmenau
Germany
florian.liers@tu-ilmenau.de

Raimund Schatz

FTW Forschungszentrum Wien
Donau-City-Straße 1
1220 Wien
Austria
schatz@ftw.at

Barbara Staehle

Fraunhofer-Institut für Integrierte
Schaltungen IIS Abteilung
Kommunikationsnetze
Nordostpark 93
90411 Nürnberg
Germany
barbara.staehle@iis.fraunhofer.de

Dirk Staehle

DOCOMO Communications
Laboratories Europe GmbH
Landsberger Str. 312
80687 München
Germany
staehle@docomolab-euro.com

Thomas Volkert

Ilmenau University of Technology
Integrated Communication Systems
Group
Helmholtzplatz 5
98693 Ilmenau
Germany
thomas.volkert@tu-ilmenau.de

Florian Wamser

University of Würzburg Institute of
Computer Science
Am Hubland
97074 Würzburg
Germany
florian.wamser@
informatik.uni-wuerzburg.de

Abstract

Over the last decade, Quality of Experience (QoE) has become a new, central paradigm for understanding the quality of networks and services. In particular, the concept has attracted the interest of communication network and service providers, since being able to guarantee good QoE to customers provides an opportunity for differentiation. In this paper we investigate the potential as well as the implementation challenges of QoE management in the Internet. Using YouTube video streaming service as example, we discuss the different elements that are required for the realization of the paradigm-shift towards truly user-centric network orchestration.

To this end, we elaborate QoE management requirements for two complementary network scenarios (wireless mesh Internet access networks vs. global Internet delivery) and provide a QoE model for YouTube taking into account impairments like stalling and initial delay. We present two YouTube QoE monitoring approaches operating on the network and the end user level. Finally, we demonstrate how QoE can be dynamically optimized in both network scenarios with two exemplary concepts, AquareYoum and FoG, respectively. Our results show how QoE management can truly improve the user experience while at the same time increase the efficiency of network resource allocation.

1 Introduction

In today's Internet, a growing number of users consumes a large variety of different applications and services, with new ones emerging every day. As a result of this growth and the ongoing liberalization of telecommunication markets, end users are in the position to freely choose between different providers. The resulting intensive cost-driven competition among the different players has lead to the commoditization of Internet access services. However, when price levels and pricing schemes become more and more similar, another factor influencing a person's provider choice comes into play: the quality of a service as perceived by the end user, referred to as *Quality of Experience (QoE)*. Here, providers have the opportunity to differentiate themselves from others by explicitly taking into account QoE for user-centric network planning and management – in contrast to the traditional network-centric Quality of Service (QoS) paradigm.

QoE complements QoS with a holistic understanding of the factors that influence the system performance as perceived by the end user. In contrast to QoS, the QoE not only depends on the network's performance but also on a wide range of other factors, including content, user terminal, application, user expectations and goals, and context of use. Understanding QoE demands for a multi-disciplinary research approach that goes beyond the network level.

In this context, *QoE management* of Internet applications is a promising solution which has the potential to resolve the following central dilemma: the delivery of applications to the end user at maximum quality, while at the same time minimizing the costs of the stakeholders involved, i.e. network providers, service providers but also cloud providers. QoE management enables to observe and react quickly to quality problems, at best before customers perceive them and decide to churn. From an economic perspective, an optimal QoE has to be achieved while constraining the application to behave as resource-efficiently as possible in order to minimize operational costs.

In contrast, today's consumer Internet traffic is transmitted on a best effort basis without taking into account any quality requirements. The backbone and the wireless access networks lack service guarantees for the predominant consumer Internet traffic which is composed of applications like P2P or client-server file sharing, web browsing, or video streaming which together make up for more than 80 % of today's traffic [1], [2]. Technical solutions enforcing quality guarantees exist, see e.g. [3], but in general the network does neither know which Internet applications it is carrying nor which quality requirements have to be met.

To be able to meet the demands of applications and users in the network, QoE management requires an information exchange between application and network. A solution for increasing the application-awareness of the network is to apply deep packet inspection (DPI) [4] to each packet transmitted over the network. Aside from legal aspects, this approach has however become rather challenging since the browser tends to become the user's interface to the Internet for an increasing number of applications like watching videos, large file downloads, online office, or gaming. In addition, a large number of applications use port 80 to successfully tunnel through NATs or to hide from detection. The next problem is that the network has to know about appropriate quality parameters. Deriving such application-specific parameters from the observed traffic flow is even more complex than detecting it, in particular for TCP traffic which adapts only to the network conditions. Both tasks could be avoided if Internet applications simply communicate their presence and their quality requirements to the network. The problem with such an approach is however that the applications need to be modified. Furthermore, for many applications like videos with variable bit rate not all QoS requirements are fully known a priori. Thus, scalable dynamic QoE management requires a perpetual *information exchange between application and network.*

In this article, we discuss the prospects for realizing truly user-centric network orchestration. Based on the following two complementary network scenarios, we derive the requirements for QoE management, using YouTube as representative example for the class of online video streaming services:

1. **WMN Scenario.** Firstly, online video platforms make up for roughly 10 % of the traffic volume of private households using a wireless mesh network (WMN) as Internet access network [2]. WMNs are multi-hop networks which require adequate resource management (RM) measures in order to ensure a good QoE for the users. We therefore discuss the Application and Quality of Experience Aware Resource Management for YouTube in Wireless Mesh Networks (*AquareYoum*) idea which uses the example of YouTube video streaming to demonstrate the appeal of application and QoE aware RM for WMNs.
2. **Global Scenario.** Secondly, video streaming dominates global Internet traffic and is expected to account for 57 % of all consumers Internet traffic in 2014. In this context, we discuss how network and application can jointly optimize YouTube QoE by exchanging information over application-network interfaces. Furthermore, we discuss the "Forwarding on Gates" (FoG) stack as possible implementation solution.

As major contribution of this work we present the two approaches for YouTube QoE management (AquareYoum in the WMN scenario, FoG in the global scenario) and abstract them into a complete picture as illustrated in Figure 1 This includes the development of a QoE model as well as QoE monitoring solutions for YouTube.

The remainder of this paper is structured as follows. Section 2 elaborates the different elements of QoE management and gives an overview on its application on YouTube in the WMN and global scenarios. The understanding and modeling of YouTube QoE is highlighted in Section 3 which allows deriving appropriate QoE monitoring approaches in Section 4. In particular, YoMo as client side monitoring tool and a passive monitoring approach in the network are introduced. QoE optimization is then considered for the two different scenarios. In the WMN scenario, YoMo allows to select the appropriate resource management mechanism, while for the global scenario a holistic approach based on collaboration between network and application is considered. Finally, Section 6 concludes this work with an outlook on remaining research challenges.

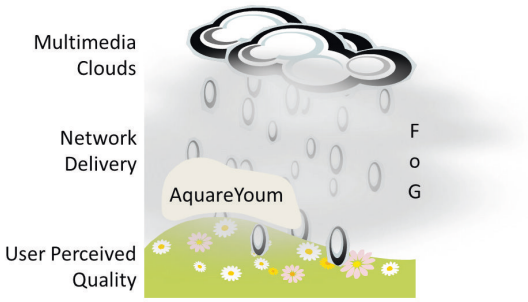

Figure 1 Clouds, Rain, FoG: While AquareYoum provides local QoE management within a wireless mesh network, FoG implements information exchange between application and network in the global scenario.

2 QoE Management Methodology: Modeling, Monitoring, Optimizing

From a conceptual perspective, QoE management requires three basic research steps,

1. modeling QoE,
2. monitoring QoE,
3. optimizing QoE.

This QoE Modeling, Monitoring, Optimization (*MoMO*) approach as introduced for cloud applications in general [5] is applied to video streaming via the YouTube cloud in this section. The 2 × 3 jigsaw puzzle in Figure 2. summarizes the elements required for YouTube QoE management in both scenarios (WMN and FoG). The rows indicate the different MoMO steps, while the columns present different solutions.

Modeling QoE. The fundamental step is understanding the applications' requirements and the impact of disturbances on the user perceived quality. For YouTube video streaming, the application requires a certain amount of network bandwidth to fulfill a smooth video playout. In particular, YouTube

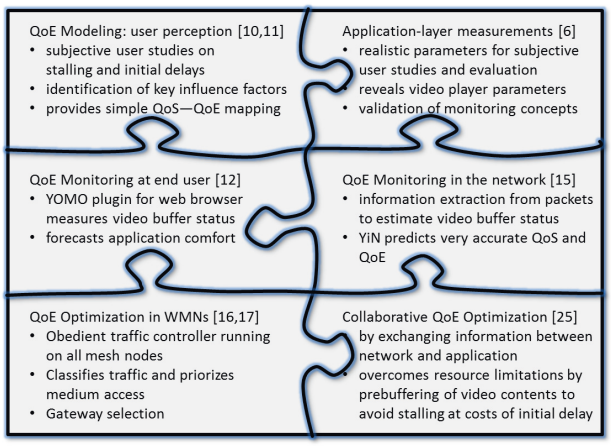

Figure 2 QoE Modeling, Monitoring, Optimization (MoMO) – A framework for QoE management.

QoE is different from traditional UDP-based video streaming, since TCP is used as transport layer protocol and a technique, called pseudo-streaming, is used. Compared to UDP-based streaming, the audiovisual content is not distorted in case of failures since video buffering is used. The video playout is not started until the buffer is filled to a certain level. In particular, the video playback is delayed at the beginning. If now during the playback, the available network data rate is lower than the video bit rate, the video transmission becomes too slow, gradually emptying the playback buffer until this runs empty. Then, the user notices interrupted video playback, commonly referred to as *stalling*.

Hence, YouTube QoE modeling consists of two parts that are (1) the user perception quantifying the impact of a given stalling pattern on the user perceived quality and (2) application layer measurements providing different stalling patterns depending on the actual network conditions. The latter can be used to model application-layer QoS, i.e., how often the service is interrupted and how long. However, in order to quantify the user satisfaction with a service, a user perception model has to be derived by means of subjective user studies. Thereby, the application-layer measurements serve as input for realistic stallings patters [6]. Furthermore, they reveal the video player parameters as implemented by the YouTube player which need to be known for passive YouTube QoE monitoring in the network, cf. Section 4.2. In addition, the application-layer measurements allow to validate the developed monitoring concept.

The YouTube QoE models provided in Section 3 finally map the stalling patterns to QoE, identified as only relevant key influence factor. We additionally investigate the impact of initial delays for filling the YouTube video buffer on QoE. This allows for deciding whether QoE management should avoid stalling at costs of increased initial delays in case of insufficient network resources.

Monitoring QoE. As a result of the QoE modeling process, QoE-relevant parameters are identified which have to be monitored accordingly. In general, monitoring includes the collection of information such as

1. the network environment (e.g., fixed or wireless);
2. the network conditions (e.g., available bandwidth, packet loss);
3. terminal capabilities (e.g., CPU power, display resolution);
4. service and application specific information (e.g., video bitrate, encoding, content genre).

However, the YouTube QoE model in Section 3 identifies stalling as the key influence factor which needs to be mapped to QoE. Hence, QoE monitoring for YouTube requires monitoring or estimating the video buffer status in order to recognize or predict when a stalling occurs.

The QoE monitoring can either be performed a) at the end user or terminal level (see Section 4.1), b) within the network (see Section 4.2), or c) by a combination thereof. While the monitoring within the network can be done by the provider for fast reaction on degrading QoE, it requires mapping functions between network QoS and QoE. When taking into account application-specific parameters additional infrastruc-

ture like DPI is required to derive and estimate these parameters within the network – which is the case for YouTube as we will discuss in Section 4.2.

A better view on user perceived quality is achieved by monitoring at the end user level. However, additional challenges arise, e.g., how to feed QoE information back to the provider for adapting and controlling QoE. In addition, trust and integrity issues are critical as users may cheat to get better performance. Section 4.1 presents the YoMo tool which works at the client and monitors the buffer status of the video player which is referred to as *application comfort* (AC). Monitoring the YouTube AC allows to predict when the user's QoE will be degraded by a stalling event. Thereby, a timely notification can be sent to QoE optimization mechanisms like resource management in WMN. Another advantage of YoMo is moreover that if the user does not simply watch the video but uses the YouTube scrollbar to navigate within the video, a QoE degradation can still be predicted.

Optimizing QoE. The final step of QoE management is the dynamic adaptation and control thereof to deliver optimal QoE so that the user may not get dissatisfied or abandons the service. QoE control aims at reacting before the user encounters problems and uses monitoring information to adjust corresponding impact factors. QoE management addresses the following questions,

1. where to react, i.e., at the edge, within the network, or both;
2. when to react and how often; and
3. how to react and where which control knobs to adjust.

In this paper, we consider the WMN and the global scenarios which feature different QoE optimization possibilities. In the WMN, a network advisor may trigger different resource management tools, as a traffic shaper running on the mesh nodes. The traffic shaper reacts only if it receives messages sent by the QoE monitoring tool (which may be one of the two proposed QoE monitoring mechanisms) and controls the bandwidth of different flows. The key concept is the interaction between network and application for joint QoE optimization. We demonstrate in a congested IEEE 802.11 based mesh testbed how this interaction allows successfully playing back the video playback without stalling by means of traffic shaping guarantees, see Section 5.1. In a similar fashion, in the global scenario, the information exchange between network and application is required to jointly optimize QoE and network costs (in terms of bandwidth). Resource limitations are overcome by adapting the initial delay to avoid stalling. Hence, this QoE optimization scheme takes only place before playing out the video, in contrast to the dynamic traffic shaping concept in the WMN scenario. It has to be clearly stated that both optimization mechanisms are complementary. Thus, in the WMN scenario, the collaborative QoE optimization, cf. Section 5.2, may execute in addition.

3 Modeling YouTube QoE

Generic relationships between measurable technical performance and QoE are a fundamental step towards understanding and modeling QoE. A typical approach for assessing QoE is calculating mean opinion scores (MOS) generated by subjective tests. That is, the opinions of individual users are aggregated and meant to reflect the opinion of an average user for a certain service used under technical conditions. Due to exponential [7] or logarithmic [8] interdependency between QoS and QoE, the QoE tends to be highly sensitive in certain QoS ranges. Particularly in these cases single averaged MOS values are not sufficient for QoE management. Thus, also the user diversity, e.g., reflected by standard deviation of MOS or in terms of distributions, also needs to be taken into account. A generic dependency between user diversity and MOS is proposed in [9], which also provides some concrete values for the user diversity of YouTube QoE.

3.1 Key Influence Factors on YouTube QoE

For deriving the key influence factors on YouTube QoE, we conducted subjective tests by means of the crowdsourcing platform Microworkers.com at University of Würzburg and in the 'i:lab' laboratory at FTW in Vienna, see [10]. In the context of QoE management, we are mainly interested in relating the stalling pattern to YouTube QoE. Based on the measurements in [6], we varied the following parameters:

1. the number of stalling events as well as
2. the length of a single stalling event, resulting in
3. different total stalling times. We also considered the influence of the test video id in order to take into account the
4. type of video as well as the
5. resolution,
6. used codec settings, etc. Further, we asked the users to additionally rate
7. whether they liked the content (using a 5-point ACR scale). We collected additional data concerning the background of the user by integrating demographic questions including
8. age,
9. gender,
10. family situation,
11. education,
12. profession,
13. home country, and
14. home continent. We also asked questions regarding their
15. Internet application usage habits in the survey. Furthermore, we additionally collected data such as
16. access network speed and
17. browser used in order to identify potential influence factors on YouTube QoE.

Finally, the key influence factors on YouTube QoE are identified by means of (a) correlation coefficients and (b) support vector machine (SVM) weights in [10]. From the results we can clearly observe that the stalling parameters dominate and are the key influence factors. Surprisingly, the user ratings are statistically independent from the video parameters (like resolution, video motion, type of content like news or music clip, etc.), the usage pattern of the user, as well as its access speed to reflect the user's expectations.

3.2 QoE Model for Stalling Pattern and Initial Delays

The analysis in the previous subsection has shown that YouTube QoE is mainly determined by stalling frequency and length only. Concrete mappings functions are provided in [10] and used in this section later on. Further on, we take a closer look at initial delays which may be accepted by the user for filling up the video buffers to avoid stalling. In case of bad network conditions, providers have to trade off between these two impairment types, i.e. stalling or initial delays. This understanding allows QoE management for YouTube video streaming clouds, see Section 5.2. Therefore, we have to answer the question whether initial delays are less harmful to QoE than stalling events for YouTube.

To this end, we analyze the subjective user ratings for the initial delay tests [11] and stalling tests [10]. The injected waiting times, either in terms of initial delay or in terms of stalling duration, range from 0 s until 32 s. We consider YouTube videos of different contents with duration 30 s and 60 s, respectively. Furthermore, we investigate single stalling events as well as several stalling events with fixed length L each.

Figure 3 shows the curve fitting functions as provided in [10], [11] for the QoE in terms of mean opinion scores depending on the total delay T. The MOS can take the following values: (1) bad; (2) poor; (3) fair; (4) good; (5) excellent. First, we consider the impact of *stalling* on YouTube QoE. It has to be noted that the MOS values in case of no stalling, i.e. $T = 0$ s, are around 4.5 due to rating scale effects in subjective studies. Some users tend to not completely utilize the entire scale, i.e. avoiding ratings at the edges. For the mapping functions in case of multiple stallings the values are close to 5 due to applied curve fitting. Nevertheless, the results clearly show hat users tend to be highly dissatisfied with two ore more stalling events per clip, i.e. when the total stalling duration exceeds a few seconds only. The concrete MOS values depend on the video duration and the actual stalling pattern, as the curves differ for single stalling events as well as multiple stalling events of length $L = 1$ s and $L = 3$ s, respectively. Hence, it is not possible to characterize the stalling pattern by a simple total stalling duration figure only. From a QoE management perspective, stalling has to be avoided to keep YouTube users satisfied. Hence, YouTube QoE monitoring has to proactively detect imminent stalling, so that QoE control mechanisms are triggered timely in advance.

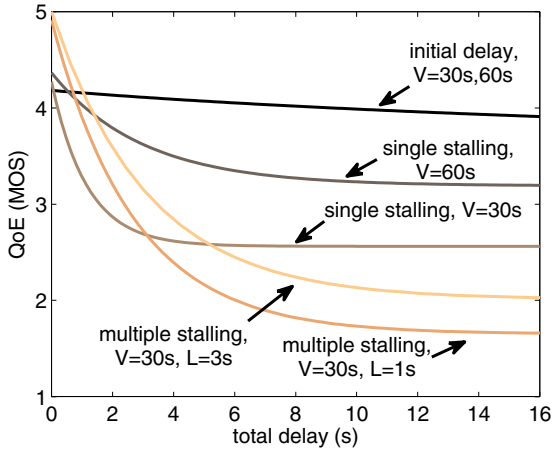

Figure 3 Stalling vs. initial delay for YouTube QoE for videos of duration V. Single and multiple stalling events (with fixed length L per event) are considered separately.

Next, we consider initial delays and compare them with stalling events of same duration. [11] shows that no statistical difference are observed for video clips of 30 s and 60 s regarding the QoE impact of initial delays. Furthermore, the results in Figure 3 clearly show again that service interruptions have to be avoided in any case from a user-centric point of view. Even very short stalling events of a few seconds already decrease user perceived quality significantly. However, initial delays are tolerated up to a reasonable level. Thus, QoE management may utilized this accordingly to overcome insufficient network resources for smoothly playing out the video.

4 Monitoring YouTube QoE

This section discusses two YouTube QoE monitoring approaches which differ in terms of measurement point and layers on which information is captured. The YouTube monitoring tool YoMo [12] measures the video player buffer status directly at the end user site on application layer. This allows to predict an imminent stalling taking into account user interactions as pausing the video or jumping within the video, but requires an instance of YoMo running at the user device. Additional challenges of monitoring at the end user level address privacy concerns of users as well as trust and integrity issues due to cheating users to obtain better performance by fraud. However, monitoring at the end-user gives the best view on user perceived quality and YoMo itself is a lightweight Java tool cooperating with a Firefox plugin. Implementation details on YoMo are given in Section 4.1.

In contrast, monitoring within the network leads to opposite disadvantages and advantages. The YouTube in-network (YiN) monitoring approach aims at detecting and measuring stalling of the video playback by approximating the video buffer status (which cannot be measured directly from network data). Thus, the main challenge is the accurate reconstruction of the stalling events that arrive at the application layer, using network packet traces only. Due to the reverse engineering of YouTube player parameters in [6], an accurate reconstruction of stalling events just on behalf of network-level measurement data is possible and will be explained in Section 4.2. However, this requires a deep packet inspection of the packet flow and hence additional costs. Furthermore, this requirement also limits scalability in terms of the number of YouTube video streaming flows that can be actually monitored by a probe. On the positive side, an ISP can monitor independently the QoE of YouTube consumers in his network on his own without any user involvement.

4.1 YoMo: Monitoring at the End User

The YouTube monitoring tool YoMo performs several tasks [12].

1. Detect a YouTube flow and forward this information to the mesh advisor which is able to trigger adequate RM tools in order to avoid a QoE degradation (cf. 5.1).
2. Analyze the packets of the YouTube flow to calculate the video buffer status β.
3. Constantly monitor β and raise an alarm if this falls below a critical threshold.

The YouTube player is a proprietary Flash application which concurrently plays a FLV file and downloads it via HTTP. A new TCP connection is opened each time a new video is downloaded or if the user jumps to another time in the video. YoMo monitors the clients incoming traffic and identifies a YouTube video flow by detecting the FLV signature. The data of each YouTube video flow is continuously parsed in order to retrieve the information embedded in the FLV tags. In particular, each FLV tag includes the time when the frame is to be played out, allowing to derive the currently available playtime T preloaded in the video buffer.

To monitor the YouTube AC, YoMo constantly calculates the amount of playtime β buffered by the YouTube player giving the amount of time, the player can continue playing if the connection to the server is interrupted. It is calculated as the difference between T and the current time instant of the video t. If β falls below an alarm threshold, a stalling event impends. Hence, YoMo has to e.g. notify the mesh advisor which applies a suitable resource management action as discussed in Section 5.1. Another option would be to decrease the video quality in order to reduce the required bandwidth (see [13] in case of scalable video codec streams) which could be easily implemented at the end user site by utilizing the YouTube API.

The current video position t can not be obtained from the FLV tags, but from the YouTube player only. The YouTube player can be accessed by the YouTube API with scripting languages only. Therefore, YoMo uses a simple Firefox extension which retrieves t from the YouTube player and sends it to the YoMo software. In [12] further technical details are described extensively. It is moreover shown that in the case of a sudden connection interruption, YoMo predicts the time of the video stalling time with an accuracy of about 0.1 s.

What makes YoMo especially suitable for QoE management is its ability to predict the time of stalling in advance. If YoMo is hence e.g. used for radio resource management as discussed in the WMN scenario in Section 5.1, it allows a network operator to react *prior* to a QoE degradation and thereby to avoid unsatisfied customers. YoMo and the Firefox plugin may be downloaded from the G-Lab website[1].

4.2 YiN: Passive YouTube QoE Monitoring in the Network

The passive YiN monitoring approach detects YouTube video flows similar as described in Section 4.1. It extracts video information from network packet data, referred to as monitoring approach 'M3: Video Buffer' in [14]. In particular, the size and time stamps of (audio and video) frames are retrieved by means of deep packet inspection. Together with the YouTube video player parameters, in particular the playing threshold Θ_1 and the stalling threshold Θ_0, the video buffer status is estimated almost exactly on behalf of network data only. As soon as the YouTube video buffer exceeds Θ_1, the player starts the video playback. If the buffer underruns Θ_0, the video stalls. The player parameters are determined in [14] based on the application-layer measurements in [6]. However, it has to be noted that there may small deviations of these values from video to video in practice, since the player takes into account the actual structure of the video codec for optimized video playout. Consequently, such small errors may propagate and lead to inaccuracies.

The basic idea of YiN is to compare the playback times of video frames and the time stamps of received packets. We define the frame time τ_i as follows. After receiving the i-th acknowledgment on TCP layer at time t_i, a total amount of $\nu = \sum_{j=1}^{i} \nu_i$ bytes has been downloaded. Together with the size of each video frame and the video frame rate – typically around 25 $^{\text{frames}}/_{\text{s}}$ –, the frame time τ_i corresponds to the downloaded video 'duration' so far. Then, we define the play time ρ_i and the stalling time σ_i to be the user experienced video play time and stalling time after the i-th TCP acknowledgment. The actual amount of buffered video time is indicated by β_i. The boolean stalling variable ψ_i indicates whether the video is currently playing ($\psi_i = 0$) or stalling ($\psi_i = 1$).

On behalf of these measures the stalling pattern over time, i.e. over the TCP acknowledgments, can be computed as follows [15].

$$\psi_i = \psi_{i-1} \wedge \beta_{i-1} < \Theta_0 \vee \neg\psi_{i-1} \wedge \beta_{i-1} < \Theta_1 \quad (1)$$

$$\sigma_i = \sigma_{i-1} + \begin{cases} t_i - t_{i-1}, & \text{if } \psi_i \\ 0, & \text{if } \neg\psi_i \end{cases} \quad (2)$$

$$\rho_i = \rho_{i-1} + \begin{cases} 0, & \text{if } \psi_i \\ t_i - t_{i-1}, & \text{if } \neg\psi_i \end{cases} \quad (3)$$

$$\beta_i = \tau_i - \rho_i \quad (4)$$

The actual video buffer can then be approximated by the difference between the frame time τ_i and the actual play time ρ_i. The iterative computation of the different variables is initialized in the following way, since YouTube first starts playing until the threshold Θ_1 is exceeded to fill the video buffer.

$$\sigma_0 = 0, \quad \rho_0 = 0, \quad \psi_0 = 1\,. \quad (5)$$

To evaluate the accuracy of the YiN monitoring approach, the estimated and the actual video buffer measured on application layer are compared. Furthermore, we map finally the stalling patterns to QoE according to the YouTube QoE model (Section 3) and compare the difference between 'measured' and 'estimated' QoE based on the reconstructed stalling patterns. The results in [15] show that the stalling pattern is almost exactly predicted with a coefficient of correlation between measured and estimated values about 0.9998. Nev-

[1] http://www.german-lab.de/go/yomo

ertheless, these differences may lead to strong QoE differences due to the non-linear perception of stalling. As a result, for 80 % of the videos investigated the QoE difference is almost zero. However, differences can be as large as one step on the MOS scale, as observed for 10 % of the videos. Thus, the monitoring approach may estimate good quality (MOS 4), while the users actually only experience a fair quality (MOS 3). The main reason for these inaccuracies is – as described above – error propagation. However, since according to [10] end-user quality perception and the underlying mapping from stalling QoS to YouTube QoE are highly non-linear, a relatively small measurement error can result in aforementioned MOS differences. For example, when the number of stalling events is very low, one stalling more or less already makes a huge difference in QoE. As a consequence, an ISP has to take these error margins into account and set his alarm thresholds accordingly.

5 Optimizing YouTube QoE

The final step of QoE management aims at optimizing QoE in a controlled fashion. We first consider the WMN scenario in Section 5.1 to show how YouTube QoE can be maintained by cooperation between QoE monitoring and appropriate RM mechanisms. The subsequently described AquareYoum suite is an implementation of the more general *Aquarema* concept which is short for Application and QoE Aware Resource Management [16]. Aquarema enables application specific network resource management and thereby improves the user QoE in all kinds of networks for all kinds of applications. This is achieved by the interaction of a QoE monitoring tool, e.g. YoMo or YiN, with a network advisor acting upon an imminent QoE degradation. For the case of YouTube running in a WMN AquareYoum is a concrete implementation of the Aquarema idea. In many experiments in congested WMN testbeds, the interaction of YoMo, the mesh advisor, and various resource management actions has enabled a stalling-free YouTube video playback [16]–[18].

In the complementary global scenario, we optimize YouTube QoE by avoiding stalling at costs of initial delay for prebuffering. This is necessary in the presence of insufficient network resources. The question arises how to set up the initial delay in such a way that stalling occurs with low probability. In Section 5.2, we derive an approximation for levels of initial delay that are just high enough so that stalling is unlikely to occur. However, since information from the network (e.g. available bandwidth) and the application (e.g. the video bitrate) has to be exchanged, we discuss how to exchange required information using the G-Lab application-to-network interface (GAPI) and propose the FoG stack as possible implementation solution. This is currently work in progress, but proof-of-concept implementations have already been demonstrated in [19], [20].

5.1 AquareYoum

QoE aware resource management is especially promising for wireless networks in general and in IEEE 802.11 multi-hop networks (WMNs) which are increasingly used as Internet access networks, in particular. Due to the multi-hop structure, enforcing strict QoS guarantees in WMNs is difficult. Moreover, a link between two nodes has not a constant capacity, but has to share its access time with surrounding links. QoE based resource management see e.g. [21] therefore continuously adapts the network resources to quality feedback from the application and is well suited for WMNs.

As already discussed, stalling is the main negative influence factor on the YouTube user QoE. The goal of the AquareYoum suite is therefore to avoid a stalling of the YouTube video. Resource management in IEEE 802.11 WMNs covers a multitude of possibilities. Therefore, the core component of AquareYoum is the *mesh advisor* which is informed about the YouTube flows routed over the network and the current network status. Based on this holistic view, it is able to trigger a suitable *resource management (RM) tool* if this is necessary to avoid a YouTube QoE degradation. The network-application interaction is assured by the already described *application monitor* YoMo. It informs the mesh advisor via detection messages *DM* and alarm messages *AM* if it detects a YouTube flow and if an imminent QoE degradation is imminent. The mesh advisor evaluates a warning together with the information about the network status it periodically receives from the *mesh monitor* tools and decides about the appropriate RM action.

The interaction of the AquareYoum components is depicted in Figure 4. Note that in dependence of the used RM tool the mesh advisor requires different information from the mesh monitoring tools. A holistic implementation integrating different monitoring and resource management tools and especially policies for selecting the right tool is the subject of our current work. Subsequently we will describe a number of already implemented solutions where

1. traffic shaping [16],
2. gateway selection [17], and
3. the steering of a mobile mesh node[18]

are used as resource management actions.

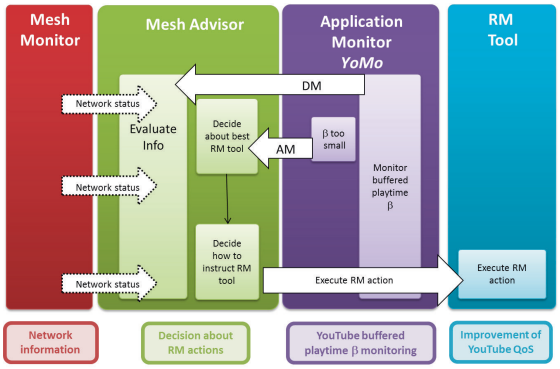

Figure 4 Interaction between the AquareYoum components.

5.1.1 Traffic Shaping

When a YouTube video is about to stall, the corresponding download lacks bandwidth. The most probable reason for this is that the links it is using are overloaded or that the nodes forwarding the flow can not access the channel often enough as neighboring nodes are highly loaded and thereby cause too much interference. The first problem is known from the wired domain and caused by *in-band* cross traffic. The second problem is wireless specific and due to *out-band* cross traffic.

In both cases, the YouTube flow gets more bandwidth and the video playback continues smoothly if the bandwidth of the cross traffic is reduced. In [16] we described a very straightforward AquareYoum implementation which does not require the presence of a mesh advisor. Instead, a good YouTube QoE is achieved by forwarding YoMo's DM and AM messages to instances of a traffic shaping tool running on mesh nodes which are forwarding, or which are interfering with the YouTube flow. If the traffic shapers receive an alarm message the maximum bandwidth of both inband, and outband best effort cross traffic will be reduced. The YoMo signaling mechanism allows to recognize that the network situation improves, and that bandwidth available for the best effort traffic can be increased again.

Implementation details are provided in [16]. The results in this paper also demonstrate that the cooperation of YoMo with the traffic shapers successfully avoids stallings for both in-band and out-band cross traffic. Different parametrization for the alarm threshold and for the quantity of the traffic reduction allow to adapt the mechanism to the network configuration.

5.1.2 Gateway Selection

For assuring the YouTube QoE in large WMNs with more than one Internet gateway we implemented a more complex concept which was one winner of the 2011 KiVS communication software award [17]. In this setup, the mesh advisor constantly receives information about the amount and type of traffic on the different backhaul links from mesh monitoring instances running on each gateway node. The mesh advisor consequently always knows which gateway is the momentarily least congested one. Again, YoMo signals the presence and an imminent QoE degradation via DM and AM messages to the mesh advisor. If the mesh advisor receives an alarm message, it uses its knowledge about the WMN to find the least congested gateway. Subsequently, it instructs the gateway selection tool to move the YouTube flow to this gateway and thereby avoids the stalling and the QoE degradation.

The gateway selection RM tool is implemented to allow for a seamless relocation of a YouTube TCP flow. For details refer to [17]. The functionality of this setup was evaluated in the DES-testbed of the Free University of Berlin[2]. The Internet gateways were located in the G-Lab experimental facility[3] in order to emulate different realistic Internet qualities of the backhaul links. Our extensive experiments proved that under various conditions, the cooperation of the four different AquareYoum components is able to improve the QoE of a YouTube user by seamlessly moving the YouTube flow to the least congested gateway if necessary.

5.1.3 Steering of A Mobile Mesh Node

For networks with mobile mesh nodes like the previously introduced DES-testbed, a variant of AquareYoum triggering a mobile mesh node was implemented [18]. In this setting, the *DES-SERT* framework [22] takes the role of the mesh monitor and provides network status information to the mesh advisor. In particular, the mesh advisor constantly queries the *DES-SERT* routing daemon for information about the state of the network. As before, the application monitor tool YoMo monitors the YouTube AC and sends DM and AM messages to the mesh advisor. The RM tool which can be triggered in this setting, is a robot carrying a mesh node on its back. If advised so, it will move to a nonfunctional part of the network and is consequently able to compensate for node failures.

During a repeated number of experiments, this variant of the AquareYoum idea has proved to ensure a smooth video playback in cases where the self-organizing routing protocol implemented in *DES-SERT* is not able to care for enough bandwidth for the YouTube video after a node failure. The mesh advisor recognizes this problem and triggers the robot to move to the location of the nonfunctional mesh node. After another routing reorganization, the YouTube flow is moved this path. As a result the AC of the YouTube video increases again and the video playback continues without stalling.

5.2 Collaborative Optimization between Network and Application

This section focuses on optimizing QoE for YouTube video streaming by prebuffering as much video data until no stalling occurs. Let us consider in the following a network with capacity B. When downloading a video which is encoded with a certain video bit rate $R < B$, stalling may occur. However, stalling may also occur if the network data rate is sufficient on average to download the video during the playout time because of the variable video bit rate [6]. To compensate such effects, a video player typically implements a video buffer. Thus, if the video is buffered long enough, no stalling will occur. From the end user's point of view, it is more convenient to experience no stalling at all during the playout even at the cost of an increased initial delay, than having small initial delays but also stalling. The question arises how to set up the initial delay in such a way that stalling occurs with low probability. Therefore, we approximate the sizes of the video (key and inter-) frames with a t-location scale distribution [23] and approximate the video buffer status.

Figure 5 shows the numerical results for the approximated initial delay T_0 compared with the optimal value T_0^* derived by analyzing the video file frame-by-frame for the YouTube video contents provided in [6]. Although in most cases the approximation is close to the optimum, there are several cases (about 25 %) which still lead to stalling. The reason behind this phenomenon lies in scene changes in the video clips, after which the characteristics of the content and thus bandwidth requirement can change considerably. Thus, the required parameters for the approximation, i.e. mean and vari-

[2] http://www.des-testbed.net
[3] http://www.german-lab.de

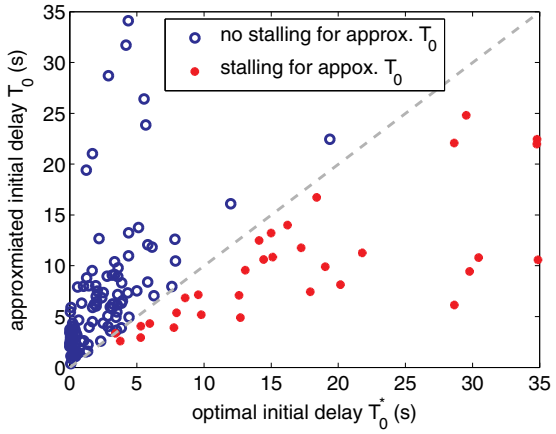

Figure 5 Scatter plot of the optimal initial delay T0*and the approximated initial delay T0.

ance of key frame and inter-frame sizes, have to be specified for each scene to improve the approximation. This is only necessary for a fraction of the YouTube videos, as they often consist only of a single scene (in terms of video encoding). Another option is to send the size for each frame ($M = 1$) or aggregated for M consecutive frames before the video is transmitted. This information enables the computation of the optimal initial delay. Since there is an upper limit on the duration of YouTube videos to be uploaded, which is now 15 min, the number of maximum frames per video is below 27000 at a frame rate of 30 $^{\text{frames}}/_{\text{s}}$. The parameter M adjusts the trade-off between signaling overhead and accuracy of the approximation.

To realize the information exchange between network and application to compute optimal initial delays T_0^* before video playout, several solution approaches exist. For example, the video frame structure is sent as meta-data before the transmission of the video data to the application. Another option is that the application (or some network entity) signals the video server the currently available network capacity, such that the video server (having the entire video structure information) computes T_0^* and sends it back to the client.

Such requirements cannot be passed to the network stack with today's APIs. Therefore, new APIs like the GAPI [24] are required. The GAPI was developed especially to provide applications a way to specify their requirements for communication associations. With the help of the GAPI function, the player is able to specify the name of the server and its list of requirements. Finally, the network stack must be able to react to these requirements dynamically. On the one hand, the stack must be able to buffer data locally, in order to sort them and to reduce the variance of the data rate. On the other hand, the network must be able to reserve data rates and to fast retransmit lost or corrupted packets. Both must be done in a scalable way in order to support the large amount of YouTube users.

One possible dynamic stack is provided by the FoG framework [19], [20]. It is a scaling inter-network system, based on dynamic composition of functional blocks. An application is enabled to define special requirements for a data transmission as described before. The network stack of FoG uses this information to select appropriate existing functions for the upcoming transmission. If needed functionality isn't available yet, FoG's routing directs each packet to the next intermediate node where new function instances can be placed in order to fulfill at least one of the desired transmission requirements. Packets are used as data input for the placed functions. In general, this system for placing functional blocks in the network can be used to direct packets through a chain of function instances, needed for video transcoding or buffering.

In addition to the creation of new instances, the system is also able to re-use existing function instances and their states for multiple connections in order to improve scalability. Finding existing and creating new function instances is done during the signaling process for setting up a communication association. The requirements are described in the header of the first signaling packet. A demo has shown that the reuse is possible without per-connection state information on the hosts which provide the desired functions. This proof-of-concept for automatic function placement places functions on the first node along the communication route, whose policy allows this. A more sophisticated placement algorithm that places functionality with focus on potential reuse is developed in current work.

6 Conclusions and Outlook

In this article we have investigated the constituents and prospects of QoE management using YouTube video as example for multimedia cloud services. Utilizing the *MoMO* approach [5] as guiding framework we showed that QoE management requires three inter-dependent fundamental elements: QoE models, QoE monitoring and QoE optimization. Regarding to YouTube QoE modeling we quantified the QoE impact of initial delay and stalling which are the two most relevant influence factors for this service category. Our analysis not only reveals the highly non-linear relationship between technical impairment level and quality perception. It also shows that stalling has strong QoE impact and should be avoided by all means, e.g. by increasing initial delay to fill the video buffer. The next step, QoE monitoring was discussed in the context of the two approaches YoMo (mesh networking scenario) [12] and YiN (global scenario), which run on the end user level and the network level respectively. Our results show that while YoMo is able to predict a stalling of the video with high accuracy, the accuracy of network level monitoring is still high enough for practical usage, with fully accurate QoE estimations for 80 % of the videos tested. Finally, we presented two approaches for QoE optimization. The first approach, AquareYoum [17], targets wireless mesh networks and relies on the mesh advisor as core component. In dependence on the current network condition, the mesh advisor is able to trigger different resource management tools in order to avoid a stalling of the YouTube video. The second approach is based on collaboration between application and network in order to minimize stalling probability by optimizing the initial delay. We showed that this can only be realized by information exchange between application and network as has been implemented by GAPI [24] and FoG [19]. Both ap-

proaches, AquareYoum and FoG have been already successfully implemented and demonstrated (cf. [16]–[20] and [25]).

For future work, we envisage extending the proposed QoE management approaches towards application to further multimedia cloud services such as Hulu and Netflix. Beyond online video streaming, we also foresee addressing other Internet-based applications such as web-browsing, file downloads and VoIP. Finally, we plan to validate the QoE optimization approaches presented in the article in the context of a large field trial as an important step towards realizing the vision of truly user-centric network and service quality management.

Acknowledgment

This work was funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (Förderkennzeichen 01 BK 0800, GLab), COST TMA Action IC0703 and by the European FP7 Network of Excellence "Euro-NF" through the Specific Joint Research Project "PRUNO: Prospects for Realizing User-centric Network Orchestration". In addition, this work has been supported within the projects ACE 2.0 and U-0 at the Telecommunications Research Center Vienna (FTW) and has been funded by the Austrian Government and the City of Vienna within the competence center program COMET. The authors alone are responsible for the content of the paper.

References

[1] Cisco Systems Inc., *Cisco Visual Networking Index - Forecast and Methodology, 2008-2013*, White Paper, 2009.

[2] F. Wamser, R. Pries, D. Staehle, K. Heck, and P. Tran-Gia, „On traffic characteristics of a broadband wireless internet access," *Special Issue of the Telecommunication Systems (TS) Journal*, 2010.

[3] DSL Forum, *TR-059: DSL Evolution – Architecture Requirements for the Support of QoS-Enabled IP Services*, 2003.

[4] *Traffic inspection for visibility, control and new business opportunities*, Ericsson White Paper, 2008.

[5] T. Hoßfeld, R. Schatz, M. Varela, and C. Timmerer, „Challenges of QoE Management for Cloud Applications," *IEEE Communications Magazine*, April 2012.

[6] T. Hoßfeld, T. Zinner, R. Schatz, M. Seufert, and P. Tran-Gia, „Transport Protocol Influences on YouTube QoE," University of Würzburg, Tech. Rep. 482, Jul. 2011.

[7] M. Fiedler, T. Hossfeld, and P. Tran-Gia, „A generic quantitative relationship between quality of experience and quality of service," *IEEE Network Special Issue on Improving QoE for Network Services*, 2010.

[8] P. Reichl, S. Egger, R. Schatz, and A. D'Alconzo, „The logarithmic nature of qoe and the role of the weber-fechner law in qoe assessment," in *ICC*, IEEE, 2010, pp. 1–5, ISBN: 978-1-4244-6402-9.

[9] T. Hoßfeld, R. Schatz, and S. Egger, „SOS: The MOS is not enough!," in *QoMEX 2011*, Mechelen, Belgium, Sep. 2011.

[10] T. Hoßfeld, R. Schatz, M. Seufert, M. Hirth, T. Zinner, and P. Tran-Gia, „Quantification of YouTube QoE via Crowdsourcing," in *IEEE International Workshop on Multimedia Quality of Experience - Modeling, Evaluation, and Directions (MQoE 2011)*, Dana Point, CA, USA, Dec. 2011.

[11] T. Hoßfeld, R. Schatz, S. Egger, M. Fiedler, K. Masuch, and C. Lorentzen, „Initial delay vs. interruptions: between the devil and the deep blue sea," QoMEX 2012, Yarra Valley, Australia, Jul. 2012.

[12] B. Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle, „YoMo: A YouTube Application Comfort Monitoring Tool," in *New Dimensions in the Assessment and Support of Quality of Experience for Multimedia Applications*, Tampere, Finland, Jun. 2010.

[13] T. Hoßfeld, M. Fiedler, and T. Zinner, „The QoE Provisioning-Delivery-Hysteresis and Its Importance for Service Provisioning in the Future Internet," in *Proceedings of the 7th Conference on Next Generation Internet Networks (NGI)*, Kaiserslautern, Germany, Jun. 2011.

[14] R. Schatz, S. Egger, and K. Masuch, „The Impact of User Fatigue and Test Duration on the Reliability of Subjective Quality Ratings," *JAES – Journal of the Audio Engineering Society*, 2012.

[15] R. Schatz, T. Hoßfeld, and P. Casas, „Passive YouTube QoE Monitoring for ISPs," in *Workshop on Future Internet and Next Generation Networks (FINGNet-2012)*, Palermo, Italy, Jul. 2012.

[16] B. Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle, „Aquarema in Action: Improving the YouTube QoE in Wireless Mesh Networks," in *Baltic Congress on Future Internet Communications (BCFIC)*, Riga, Latvia, Feb. 2011.

[17] B. Staehle, F. Wamser, M. Hirth, D. Stezenbach, and D. Staehle, „AquareYoum: Application and Quality of Experience-Aware Resource Management for YouTube in Wireless Mesh Networks," *PIK - Praxis der Informationsverarbeitung und Kommunikation*, 2011.

[18] B. Staehle, F. Wamser, S. Deschner, A. Blenk, D. Staehle, O. Hahm, N. Schmittberger, and M. Günes, „Application-Aware Self-Optimization of Wireless Mesh Networks with AquareYoum and DESSERT," in *Euroview 2011*, Würzburg, Germany, Aug. 2011.

[19] F. Liers, T. Volkert, and A. Mitschele-Thiel, „Demonstrating Forwarding on Gates with First Applications," in *EuroView2010*, Würzburg, Germany, Aug. 2010.

[20] F. Liers, T. Volkert, and A. Mitschele-Thiel, „Scalable Network Support for Application Requirements with Forwarding on Gates," in *EuroView2011*, Würzburg, Germany, Aug. 2011.

[21] R. Pries, D. Hock, N. Bayer, M. Siebert, D. Staehle, V. Rakocevic, B. Xu, and P. Tran-Gia, „Dynamic bandwidth control in wireless mesh networks: a quality of experience based approach," in *18th ITC Specialist Seminar on Quality of Experience*, Karlskrona, Sweden, 2008.

[22] B. Blywis, M. Günes, F. Juraschek, P. Schmidt, and P. Kumar, „DES-SERT: A Framework for Structured Routing Protocol Implementation," in *IFIP WD'09*, Paris, France, 2009.

[23] T. Hoßfeld, F. Liers, T. Volkert, and R. Schatz, „FoG and Clouds: Optimizing QoE for YouTube," in *KuVS 5thGI/ITG KuVS Fachgespräch NG Service Delivery Platforms*, Munich, Germany, Oct. 2011.

[24] F. Liers, T. Volkert, D. Martin, H. Backhaus, H. Wippel, E. Veith, A. A. Siddiqui, and R. Khondoker, „GAPI: A G-Lab Application-to-Network Interface," in *EuroView2011*, Würzburg, Germany, Aug. 2011.

[25] F. Liers, T. Volkert, T. Zinner, and T. Hoßfeld, „Prospects for Realizing User-Centric Network Orchestration: Demonstration of FOG for Use Case of Video Streaming," under submission, 2012.