Passive YouTube QoE Monitoring for ISPs

Raimund Schatz*, Tobias Hoßfeld†, and Pedro Casas*

*Telecommunications Research Center Vienna - FTW
A-1220 Vienna, Austria - Email: surname@ftw.at

†University of Würzburg, Institute of Computer Science, Chair of Communication Networks
D-97074 Würzburg, Germany - Email: hossfeld@informatik.uni-wuerzburg.de

***Abstract*—Over last decade, Quality of Experience (QoE) has become the guiding paradigm for enabling a more user-centric understanding of quality of communication networks and services. The intensifying competition among ISPs and the exponentially increasing traffic volumes caused by online video platforms like YouTube is forcing service providers to integrate QoE into their corporate DNA.**

This paper investigates the problem of YouTube QoE monitoring from an access provider's perspective. To this end, we present three novel methods for in-network measurement of the QoE impairment that dominates user perception in the context of HTTP video-streaming: stalling of playback. Our evaluation results show that it is possible to detect application-level stalling events at high accuracy by using network-level passive probing only. However, only the most complex and most accurate approach can be used for QoE prediction due to the non-linearities inherent in human quality perception.

***Keywords*-YouTube; video streaming; Quality of Experience; network monitoring; information extraction; passive probing**

I. Introduction

Quality of Experience (QoE) describes the user perception and the resulting satisfaction with service performance in communication networks. QoE modeling and assessment is increasingly gaining attention among Internet Service Providers (ISPs) and operators. This growing interest can be explained in terms of the increased competition and the need for aggregated-value solutions, as well as by the risk of having churning clients for quality dissatisfaction. With HTTP traffic again taking the pole position in residential broadband Internet traffic [1], one might well imagine millions of people sitting in front of their browsers, watching their favorite YouTube videos and surfing in their preferred social networks. However, many users face volatile network conditions, e.g. due to signal-to-noise ratio problems on wireless or DSL links or temporary over-utilization of shared network resources. Such conditions may result in bad QoE, negatively impacting the judgment of users on their ISPs.

For an ISP delivering the service to the end user, it is thus important to understand the relationship between the users' perception and the characteristics of the service provisioning through its networks. Particularly regarding network provisioning, QoE also opens the possibility to save resources, as it is not necessary to provide better Quality of Service (QoS) for maintaining the same QoE level. For example, reserving a bandwidth of 4 Mbps for delivering a 300 Kbps video stream unnecessarily consumes ISP's resources without improving QoE. This is referred to as QoE over-provisioning [2], and shows the potential impact of considering QoE-related concepts in traditional QoS-based network management paradigms.

To avoid QoE over-provisioning and allow for cost savings, an ISP requires proper QoE-based dimensioning (in short QoE dimensioning) with respect to a particular service or a mix of services. For example, for supporting video streaming services, QoE dimensioning aims at determining the minimum bandwidth requirements such that the targeted QoE requirements are met. Thereby, the QoE target level is defined by the ISP. QoE dimensioning for VoD streaming services like YouTube faces several scientific challenges: firstly, the ISP has to identify and monitor the traffic in its network resulting from that service. Then, appropriate QoE models must be conceived, which can map the resulting measurement data into QoE levels. Finally, the ISP has to decide which corrective actions to take so that all users get good QoE levels.

The contribution of this work is on QoE monitoring approaches for the YouTube video streaming service, particularly considering the ISP perspective. By ISP perspective we refer to the fact of monitoring YouTube QoE levels using only network-level data, which can be gathered using ISPs monitoring infrastructures. Having a QoE monitoring approach from network measurements gives to ISPs a paramount advantage: that of detecting user-experience problems from their available monitoring technology, thus managing the corresponding traffic flows and the underlying network infrastructure accordingly. We particularly consider the YouTube video streaming service because it currently represents the most prominent online video portal in the Internet, with more than two billion video streams daily. Our major contribution is dedicated to the investigation of several network-layer monitoring approaches of YouTube QoE, including implementation prospects and an evaluation of their accuracy as compared to application-level monitoring techniques. YouTube QoE is primarily determined by stalling effects on the application layer, as opposed to

UDP-based video streaming common for traditional IPTV services. In this context, the main challenge is on the robust detection of stalling events at the application layer from network-level traffic.

The remainder of this paper is structured as follows. Section II discusses related work in the field of YouTube QoE monitoring. Section III presents three approaches towards passive YouTubE QoE monitoring and discusses their advantages and disadvantages. Section IV then presents a detailed evaluation of the most suitable approach in terms of QoS and QoE approximation accuracy. Finally, Section V provides some conclusions and an outlook on future work in this field.

II. Background and Related Work

Video streaming allows users to playback media while content is still being downloaded in the background. The advantages is to get almost instant access to content without having to wait for the full download.

A number of studies have investigated the characteristics of online video streaming traffic, which is a prerequisite for monitoring its QoS and QoE. Authors in [3] have studied the network characteristics of YouTube and Netflix streaming services. They showed that depending on the application (e.g., browser, mobile) and the container (e.g., Silverlight, Flash, HTML5), different streaming strategies can be observed. In total, they identified three different streaming strategies causing different traffic patterns which need to be modeled accordingly.

Flash-based YouTube streaming constitutes by far the largest share of online video traffic in current Internet streaming scenarios. In this context, the authors of [4] investigated the interaction between network and application layer, particularly regarding the impact of transport protocol and bandwidth on stalling. Authors showed that in case of TCP, the video playback rather than the audiovisual information itself is disturbed, since the transport protocol cares for the retransmissions of corrupted or lost packets itself. Furthermore, TCP adapts the transmission rate to network congestion in order to minimize packet losses. However, if the available bandwidth is lower than the required video bitrate, the client buffer becomes gradually empty, which ultimately results in stalling of the playback and thus in bad QoE. Based on a considerable body of measurement data, authors showed that the frequency of stalling events can be well approximated with an exponential function

$$F(x) = \alpha\, e^{\beta x} + \gamma \tag{1}$$

being x the normalized video demand $x = {}^{V}\!/_{B}$, defined as the ratio between video bitrate V and network bottleneck capacity B, and α, β, and γ three constant values. Furthermore, the distribution of the duration L of stalling events can be approximated by a t location-scale distribution $L \sim TLOC(\mu, \sigma, \nu)$, with stalling length and stalling frequency being uncorrelated to each other. Figure 1 shows the cumulative distribution function of the length of stalling events. The measurement values taken from [4] are fitted here with a normal distribution and the t location-scale distribution. Later on we shall use the distribution of the stalling length to evaluate the monitoring approaches in Section IV. Results provided at [4] represent therefore a good starting point for the detection of application-level stalling events from network measurements, as it will be discussed discussed in Section III.

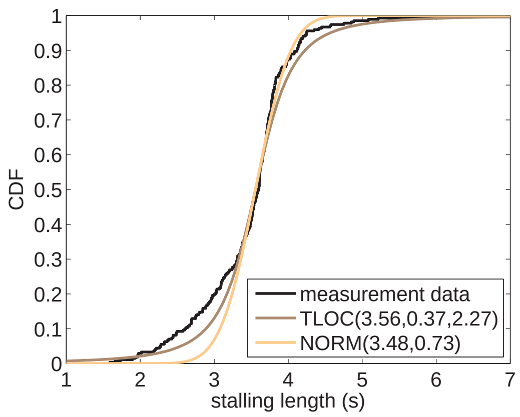

Figure 1. Fitting of the measured stalling length with a t location-scale distribution and a normal distribution.

When it comes to predicting QoE of YouTube, an essential step is determining those key factors that have the strongest influence on the actual experience. To this end, the authors of [5] have conducted a series of YouTube *crowdsourcing* studies; in the QoE context, crowdsourcing consists in outsourcing subjective testing tasks to open YouTube-users' communities. Their results show that only the number of stallings in a given period and the stalling length actually have strong impact on QoE, while many other factors such as type of browser, YouTube usage, and so forth made very little or no difference. Therefore, QoE measurement systems for the YouTube service should at least consider the aforementioned two stalling-related measures. In addition, the QoS – QoE mapping functions derived in [5] show that user perception of stalling events is highly non-linear, with one single stalling event already significantly impairing the overall experience.

Regarding YouTube QoE monitoring, most related work so far has been conducted in the context of QoE optimization and traffic management. The authors of [6] presented a client-side software tool to monitor YouTube traffic at the application level, by estimating buffer levels to predict stalling events. This approach has been successfully applied to the application-aware self-optimization of wireless mesh networks in [7], [8], in such a way that in case of likely stalling, additional resources are provided by the network. In a similar way, the ”Forwarding on Gates” (FoG) approach

[9] is used in [10] to develop a novel dynamic network stack based on functional blocks for optimizing the QoE of YouTube playback.

However, the aforementioned approaches are not applicable in the case of an ISP willing to monitor YouTube QoE in its network, since the installation of additional software on the client side as well as the migration to a non-standard network stack or topology are not practical options. For this reason, our work focuses on monitoring YouTube QoE solely on the network level, based on passive probing of TCP flows.

III. Desing of Passive Monitoring Approaches

As already mentioned in the previous section, YouTube QoE is basically determined by the stalling pattern as experienced by the end user. Therefore, the passive YouTube QoE monitoring approaches proposed in this paper rely on the following two-steps procedure. In the first step, the stalling pattern is reconstructed by extracting information from the packet trace within the network. In the second step, the reconstructed stalling pattern is mapped to QoE values by means of an appropriate QoE model, using for example the model provided at [5]. In such a context, the key challenge is to accurately approximate the stalling pattern from network-level measurements. To this end, we shall present and discuss three different passive monitoring approaches, referred to as monitoring approach 'M1', 'M2', and 'M3'. The first approach, namely 'M1', is based on the download time of the YouTube video; 'M2' relies on the end-to-end throughput of the connection; finally, 'M3' consists in approximating the actual video buffer status. All three approaches allow to estimate the stalling pattern without relying on application-level or client-side measurements.

The stalling pattern can be described by the total stalling time T, the number of stalling events N, and the duration or length of a stalling event L. In case of a bottleneck with constant capacity B, regular stalling events are observed as measured in [4]. Thus, if we assume a know distribution for L, we can formulate the first monitoring approach:

M1. Download Time

The basic idea behind M1 is quite simple. The ISP's monitoring system measures the total stalling time T as the difference between the total video download time Y and the video duration D, i.e. $T = \max(Y - D, 0)$. With a given average stalling length L, the number N of stalling events can be roughly approximated as $N = {}^{T}\!/_{L}$. The download time of the video contents can be easily extracted from packet-level traces.

A first problem with M1 arises when trying to obtain the overall duration of the video D. There are several possibilities in the practice. First, this information is available from the YouTube website and can be requested directly via the YouTube API. Therefore, the monitoring system has to extract the YouTube video identifier from the HTTP request containing the url (e.g. http://www.youtube.com/watch?v=75eGXJqsWJI) and the video id (e.g. 75eGXJqsWJI). However, the system then relies on the YouTube API for this information, which may change from time to time. A second option is to extract the information from the video header. YouTube uses for example the FLV container file format, from which meta data like video duration, framerate, and keyframes are specified. In that case, the monitoring system has to parse the network packets and needs to understand the container format. Hence, both option require some extra effort for the system to get the video duration.

A second drawback of M1 is that it considers the total duration of the video as input; if the user does not watch the entire video and aborts the downloading (which in practice frequently happens), this monitoring approach cannot be applied. Therefore, a different approach which avoids having to download the entire video is proposed by M2.

M2. Throughput

Our second passive YouTube QoE monitoring approach is based on the stalling frequency approximation as defined in Eq.(1). In that case, the bottleneck capacity B has to be estimated, which can be easily done from passive monitoring packet traces and throughput measurements [11], [12]. Furthermore, the video bitrate V has to be extracted from the packets by parsing the meta data available at the container file format. From these two values, the normalized video demand $x = {}^{V}\!/_{B}$ is computed. Finally, the number of stalling events can be approximated by $N = \min(D, Y)\, F(x)$, where Y represents in this case the actual download time of the video, and not the total video download time as in M1. Similar to M1, the video duration D has to be extracted from the packet traces.

The computational effort in the practice is comparable between both approaches M1 and M2, but M2 can be applied without completely downloading the whole video content. However, the major problem with M1 and M2 is accuracy. Both approaches estimate either the total stalling time T and/or the number of stalling events N, assuming that the distribution of the stalling length L is known. For example, as we have seen in Section II, L can be approximated by a t location-scale distribution, cf. Figure 1. Although the length of a single stalling event lies between 2 s and 5 s with high probability, this approach leads to inaccurate results and thus QoE estimations with considerable errors.

Figure 2 shows the complementary cumulative distribution function of the number of stallings N estimated for given total stalling times T, which are varied from 2 s to 16 s. The estimation is obtained by using the t location-scale approximation depicted in Figure 1, and the approximation applied in M1, i.e., $N = {}^{T}\!/_{L}$. This estimation of N clearly exhibits a large variance, particularly for longer total stalling time values. For example, for a total stalling time

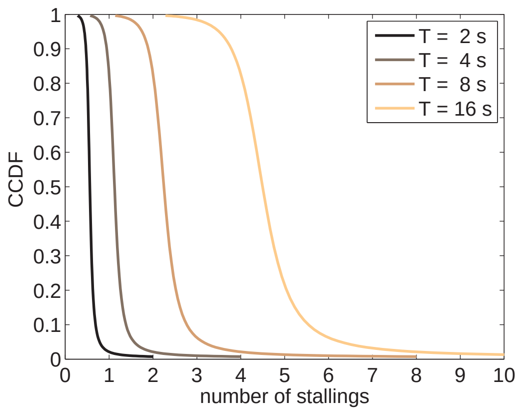

Figure 2. Monitoring approaches M1 and M2 can only estimate the number of stallings with a certain probability.

of 8 s, the number of stallings lies between 2 and 4 with high probability. However, this range already has a strong impact on the actual QoE, ultimately deciding between good and bad quality: as shown in [5], the number of stalling events is crucial for the end-user experience. Thus, the QoE differences for N and $N+1$ stalling events may be dramatic. For example, for $N = 1$ the difference is about 0.7 – 0.8 MOS in a 5-point MOS-scale. Consequently, in the practice an ISP can use both approaches only for upper or lower bound estimations of QoE.

M3. Video Buffer

In order to improve the accuracy of the YouTube QoE monitoring, we present a third approach which estimates the current video buffer status of the client player from network-level measurements. The basic idea is to compare the playback times of the video frames and the time stamps of the received packets. The costs for the improved accuracy are higher efforts in extracting information from the packet traces. In particular, the size of each video frame and the video frame rate have to be extracted from the meta data contained in the FLV container.

Let us define some additional parameters that compose the M3 approach. The first and most important parameter is the total downloaded video duration τ_i, which is updated from every new TCP acknowledgment received at time t_i. The value of τ_i is estimated from the total number of bytes downloaded until time t_i, considering both the size of video frames (I, B, P frames), and the video frame rate. In simple terms, if we know the size of the received video frames, we can estimate the number of frames to playback from the total number of bytes downloaded; then, using the frame rate of the video, we get the value of τ_i.

We additionally define the play time ρ_i and the stalling time σ_i, which are the user experienced video play time and stalling time after the i-th TCP acknowledgment. The amount of buffered video time is indicated as β_i, and it

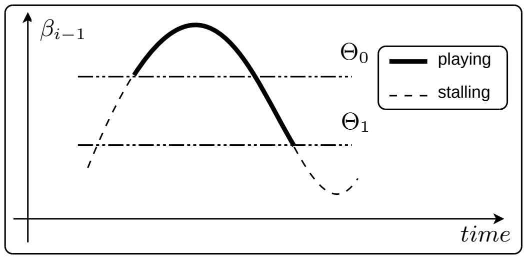

Figure 3. Management of the playback buffer in YouTube.

corresponds to the difference between downloaded video duration τ_i and played time ρ_i, i.e., $\beta_i = \tau_i - \rho_i$. We also consider a boolean stalling variable ψ_i, which indicates whether the video is currently playing ($\psi_i = 0$) or stalling ($\psi_i = 1$).

The last part of the M3 approach is composed by the way the YouTube player buffers and displays the video. The YouTube player uses two different playing and stalling thresholds to control the way it consumes video frames from the playback buffer, see Figure 3. The first threshold Θ_0 defines the minimum amount of buffered video time that has to be exceeded to start playing a stalled video; the second threshold Θ_1 specifies the minimum amount of buffered video time necessary to continue playing a video once the playback has started. So if we consider the video buffer size β_{i-1} at time t_{i-1}, then we get that if β_{i-1} exceeds Θ_0, then the video starts playing; on the other hand, if the video buffer falls bellow Θ_1, then the video stalls. Hence, stalling occurs if the following condition is true: $(\psi_{i-1} \wedge (\beta_{i-1} < \Theta_0)) \vee (\neg\psi_{i-1} \wedge (\beta_{i-1} < \Theta_1))$. Our measurement studies performed in the following section revealed that these two buffer thresholds can be reasonably taken as $\Theta_0 = 2.2\,\text{s}$ and $\Theta_1 = 0.4\,\text{s}$. Using these definitions, the stalling pattern of a YouTube video over time can be obtained as follows:

$$\psi_i = \psi_{i-1} \wedge (\beta_{i-1} < \Theta_0) \vee \neg\psi_{i-1} \wedge (\beta_{i-1} < \Theta_1) \quad (2)$$

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

$$\rho_i = \rho_{i-1} + \begin{cases} 0, & \text{if } \psi_i \\ t_i - t_{i-1}, & \text{if } \neg\psi_i \end{cases} \quad (4)$$

$$\beta_i = \tau_i - \rho_i \quad (5)$$

Finally, as depicted in Eq. (3) and Eq. (4), the time elapsed between the previous acknowledgment at time t_{i-1} and current acknowledgment at time t_i increases the play time ρ_i or the stalling time σ_i, depending on the resulting video state (i.e., playing or stalling).

Since YouTube first starts buffering (i.e., stalling state) until the threshold Θ_0 is exceeded, the iterative computation of the different variables is initialized in the following way:

$$\sigma_0 = 0, \quad \rho_0 = 0, \quad \psi_0 = 1\,. \tag{6}$$

Obviously, the monitoring approach M3 is the most complex and requires the highest computational effort, since all packets have to be analyzed and the information about the frame sizes has to be extracted. However, the approaches M1 and M2 are not accurate enough for QoE monitoring in the practice. For this reason, we consider only the M3 approach for evaluation in the rest of the paper.

IV. QoS and QoE Evaluation of the Monitoring Approach

The evaluations of the M3 approach for passive YouTube QoE monitoring were performed by measuring the stalling events on the application layer, and by capturing the packets on the network layer within a controlled environment. Then, the M3 approach was applied to this data to estimate the stalling pattern; finally, the obtained stalling pattern was compared to the actual stalling as observed in the application, which served as ground truth. This comparison of stalling patterns represents a QoS-based evaluation, because the user experience is not directly involved. In order to evaluate the QoE monitoring and estimation properties of M3, we map both the ground truth application layer stalling patterns and the estimated stalling patterns to QoE.

A. Measurement Setup

Our measurements took place from June 2011 to August 2011 in a laboratory at FTW in Vienna. The controlled environment was realized by emulating network conditions resulting into stalling on application layer. We used existing network emulation tools like Dummynet [13] to control the network conditions in terms of packet loss, delays, and throughput in uplink and downlink direction. Then, we implemented a script which simulates a user watching YouTube videos in his browser. Therefore, a local web server was configured and web pages were dynamically generated, which call the YouTube API for embedding and playing the YouTube video. The video player status and the used buffer size were monitored within the generated web page using Javascript. At the end of the simulation (i.e. when the simulated user completely watched the video, after a certain timeout, or in case of any player errors), the stalling monitoring information and the buffer status were written to a logfile via cgi scripts. In addition, the network packet traces were captured with wireshark, in particular tshark as its command line implementation. As a result, we obtained network-level QoS parameters (from the packet traces) and application-level QoS parameters (the stalling patterns).

Next, the monitoring approach M3 was applied to the packet traces. This means that video content packets were identified and relevant information extracted. As a result of this step, information about the playback times of the video frames and the timestamps of the received network packets were obtained. This approach can be applied online during network operation, or offline using captured packet traces. For the evaluation of M3, we analyzed the packets offline.

The identification of video content from a given packet trace was achieved by analyzing the HTTP requests. The YouTube API specifies a set of calls for requesting videos through HTTP. Using pattern matching, these HTTP requests and the corresponding HTTP objects were identified. Furthermore, YouTube uses DNS translation and URL redirection, as the actual video contents are located on various caching servers. This facilitates the identification of YouTube flows. Our implementation of the video identification and extraction tool, written in Perl, returns the following information and data for each requested video within a packet trace: (a) timestamps of HTTP requests, redirected HTTP requests, and the HTTP response, (b) YouTube video identifier, and (c) video data of the HTTP response i.e. a FLV file. The video file itself, which was identified at the trace, was parsed by implementing a Perl module which analyzes the video contents. As a result, video information like video bit rate, video resolution, used audio and video codecs, or video size and duration were extracted. Furthermore, for each video frame in the video stream, information about the video playback times of frames, the size of the video frames, as well as the type of frames (key frame or interframe) were extracted. Network statistics (with respect to TCP retransmissions, estimated round trip times, throughput. etc.) were also obtained for the HTTP objects and for the entire TCP stream. Finally, based on the YouTube video identifier (b), additional information available at YouTube were downloaded and analyzed, including the popularity of files, the user ratings, etc.

B. QoS: Stalling on Application Layer

Based on the playback times of video frames and the time stamps of received packets, the stalling patterns for individual flows were approximated according to Eq.(5). The approximation of the stalling pattern depends on additional information like the threshold Θ_0, which makes YouTube start playing back the video if the video buffer exceeds this value, $\beta_i > \Theta_0$. The threshold Θ_1 indicates that YouTube stops playing if the video buffer is below this value, $\beta_i < \Theta_1$. After that, YouTube restarts playing if $\beta_i > \Theta_0$ again. Since we also monitored the application-level QoS parameters beside the network-level QoS parameters, we were able to deduce these additional parameters.

Figure 4 shows the relative error between the approximated stalling patterns and the measured stalling pattern on the application-layer, depending on the video buffer threshold Θ_1. From this evaluation, we found the best values to be $\Theta_0 = 2.2\,\mathrm{s}$ and $\Theta_1 = 0.4\,\mathrm{s}$, corresponding to the minimal relative errors. Figure 4 justifies the election for the value of Θ_1; the value of Θ_0 was obtained in a similar way.

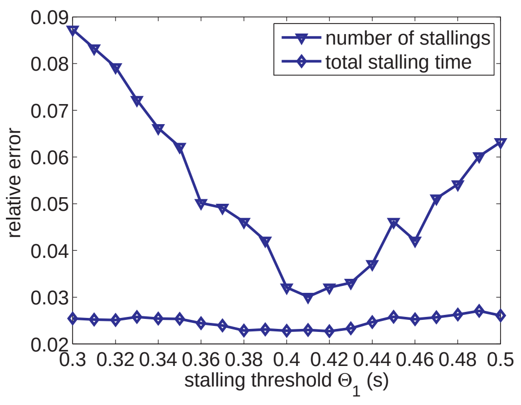

Figure 4. Reverse engineering of the video player buffer thresholds and derivation of optimal values yielding $\Theta_0 = 2.2\,\mathrm{s}$ and $\Theta_1 = 0.4\,\mathrm{s}$.

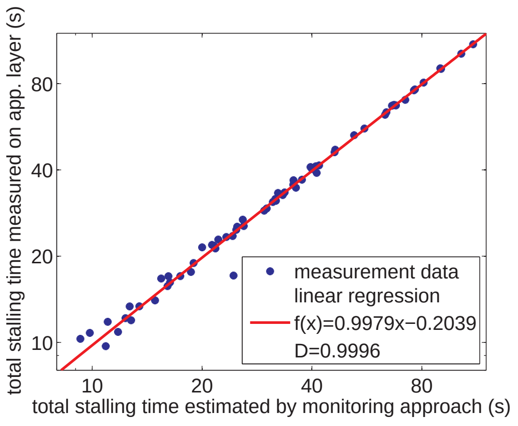

Figure 6. Comparison of total stalling times measured on application layer and estimated stalling times by YouTube monitoring approach M3.

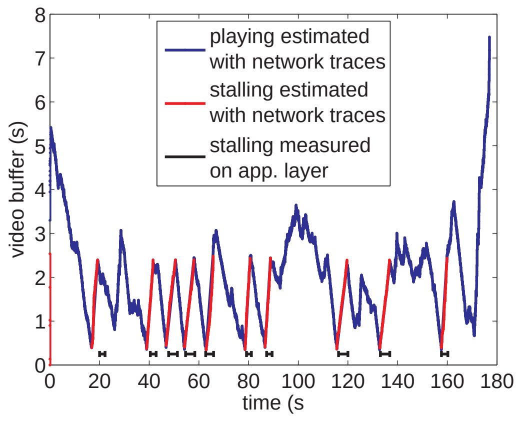

Figure 5. YouTube monitoring approach M3 estimates the video buffer in order to infer stalling.

Based on these thresholds, we were able to reconstruct the stalling pattern from the network traces. Figure 5 shows exemplary the estimated video buffer size over time. It can be seen that the video starts playing as soon as Θ_0 is exceeded (indicated by the blue line). However, when the buffer is below Θ_1 the video stalls (indicated by the red line). In addition, the stalling pattern as measured on the application layer is plotted as vertical black lines. It can be seen that the estimated and the actually observed stalling fits well. However, there are some small differences caused by various aspects. Firstly, we rely on TCP acknowledgments, but there might be network fluctuations from the client to the measurement point in the network. Next, the video buffer thresholds are average values over a large set of videos, considered in our measurements. However, the actual thresholds for an individual video depend also on the video structure, i.e. the sequence of I-, B-, and P-frames of the video. Hence, small differences to the optimally found values may emerge for some videos. Then, the small error in estimating the video buffer propagates from frame to frame.

However, a comparison of the estimated stalling time from the network traces with the actual stalling time from the application-level monitor revealed that the coefficient of determination ($R^2 = 0.9996$) was almost equal to a perfect match ($R^2 = 1$). The same quality of match was found regarding the number of estimated and monitored stalling events. Figure 6 shows the total stalling time estimated by the monitoring approach on the x-axis versus the total stalling time as measured on the application-layer. It can be seen that the estimation is very accurate with respect to the total stalling time.

However, as we have seen for the two other monitoring approaches M1 and M2, it is not sufficient to estimate the total stalling time only. Hence, we also take a closer look at the estimated number of stalling events, which we can easily obtain with M3. Figure 7 shows the absolute difference ΔN between the number of stallings N_a measured on the application layer and the estimated N_e by M3. It can be seen that about 50 % are exactly measured. However, for about 30 % of the videos, there is a difference of one stalling event. Although this does not seem to be much, it may have a strong influence on QoE. Therefore, we additionally investigate the relative difference $\Delta N^* = \frac{|N_e - N_a|}{N_a}$. From Figure 8, we can see that the relative difference is quite small and below 0.2 for 90 % of the videos.

C. QoE: Overall Quality as Experienced by the End User

The main goal of the monitoring approach is to estimate the YouTube QoE. Therefore, we finally map the stalling patterns to QoE according to the model provided at [5], and compare the difference between the ’measured’ QoE and the ’estimated’ QoE, based on the reconstructed stalling patterns. For the comparison, we consider (a) a worst case estimation to get an upper limit of the QoE difference and (b) a best QoE estimation, which uses the ’best’ QoS – QoE mapping function available in [5].

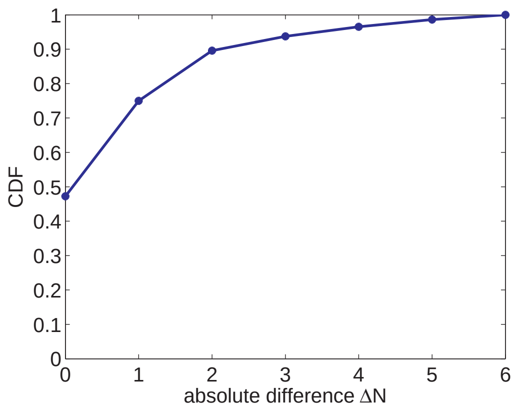

Figure 7. Absolute difference $\Delta N = |N_a - N_e|$ between the number of stallings N_a measured on application layer and N_e estimated by YouTube monitoring approach M3.

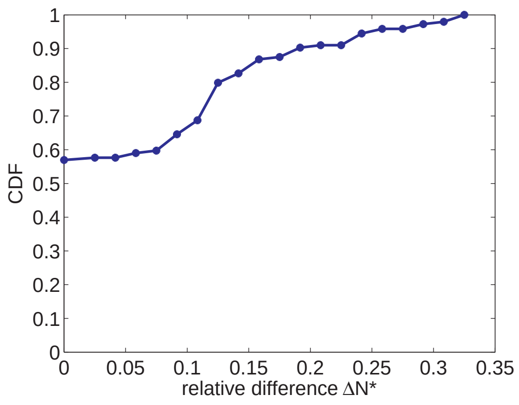

Figure 8. Relative difference $\Delta N^* = \frac{|N_a - N_e|}{N_a}$ between the number of stallings N_a measured on application layer and N_e estimated by YouTube monitoring approach M3.

For the worst case estimation, short stalling events of 1 s length are considered, which sum up to the total stalling time. This is a worst case estimation, as it leads to a higher number of stalling events than really observed. Although the QoS comparison in Figure 6 only leads to small differences between the measured total stalling time and the estimated total stalling time, these differences may lead to strong QoE differences due to the non-linear perception of stalling. Thus, the worst case estimation leads to an upper bound regarding QoE differences.

For the best case estimation, the measured number of stallings for measured lengths of single stalling events are mapped to QoE. This measured QoE value is compared to the estimated QoE, obtained from the estimated number of stallings and the corresponding lengths of single stalling events, and using the aforementioned mapping function.

The cumulative distribution function of the QoE difference is depicted in Figure 9 for both estimations. It can be seen that for about 80 % of the analyzed videos, the QoE

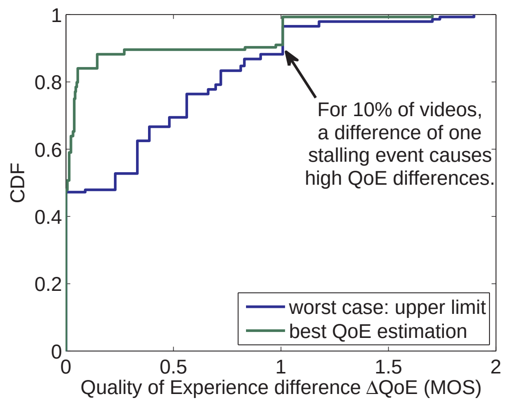

Figure 9. Quality of Experience difference ΔQoE in terms of MOS between measurements on application layer and estimation by YouTube monitoring approach M3.

difference is almost zero (best estimation). As upper bound, $^2/_3$ of the analyzed videos show a QoE difference below 0.5, which may be acceptable for an ISP. 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 the aforementioned error propagation. According to [5], end-user quality perception and the underlying mapping from stalling QoS to YouTube QoE are highly non-linear; therefore, a relatively small measurement error can result in the aforementioned MOS differences. For example, when the number of stalling events is very low, one stalling already makes a huge difference in QoE. As a consequence, an ISP has to take these error margins into account and set its alarm thresholds accordingly.

V. Conclusion

This paper has investigated the feasibility of in-network YouTube QoE monitoring via passive probing on ISP level. We showed that in the context of YouTube, stalling of the video playback is the main impairment that needs to be detected and measured. Consequently, the main challenge for YouTube QoE monitoring on ISP level is the accurate reconstruction of the stalling events that arrive at the application layer, using network packet traces only. To this end, we introduced three different monitoring approaches and briefly compared them in terms of computational effort and accuracy.

As a main result, we showed that accurate reconstruction of stalling events just on behalf of network-level measurement data is possible, and that YouTube QoE monitoring on ISP-level is thus feasible. However, we found that only the most accurate, and unfortunately the most complex approach, can be actually used for QoE monitoring purposes, since (a) stalling frequency and stalling duration both need to be measured, and (b) the non-linearity of human perception demands for high QoS measurement accuracy, particularly in those cases where stalling frequency is low.

In this paper we claim that the developed approach can be used for QoE monitoring from an ISP perspective, but we did not perform any type of scalability evaluation regarding the number of YouTube video streaming flows that can be actually monitored and assessed. This analysis depends on the particular characteristics and capacity of the monitoring infrastructures used by the ISP, but in any case, we plan to conduct evaluations on this direction, at least using standard monitoring equipment. Continuing with future work, we also plan to validate the M3 monitoring approach within the context of a field trial. Furthermore, our current approach has only been used for flash-based YouTube Streaming using the FLV container. Therefore, we envisage to extend our approach towards inclusion of HTML5 as well as mobile clients.

Acknowledgment

The research has been supported by COST TMA Action IC0703, COST QUALINET Action IC1003, and by the European FP7 Network of Excellence ”Euro-NF” through the Specific Joint Research Project “PRUNO”. 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] G. Maier, A. Feldmann, V. Paxson, and M. Allman, “On dominant characteristics of residential broadband internet traffic,” in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, ser. IMC ’09. New York, NY, USA: ACM, 2009, pp. 90–102.

[2] T. Hoßfeld, “Performance Evaluation of Future Internet Applications and Emerging User Behavior,” Ph.D. dissertation, University of Würzburg, aug 2009.

[3] A. Rao, Y.-S. Lim, C. Barakat, A. Legout, D. Towsley, and W. Dabbous, “Network characteristics of video streaming traffic,” *CoRR*, vol. abs/1111.0948, 2011.

[4] 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.

[5] 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.

[6] 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.

[7] ——, “Aquarema in Action: Improving the YouTube QoE in Wireless Mesh Networks,” in *Baltic Congress on Future Internet Communications (BCFIC)*, Riga, Latvia, feb 2011.

[8] 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.

[9] F. Liers, T. Volkert, and A. Mitschele-Thiel, “Demonstrating forwarding on gates with first applications,” in *Proceedings of EuroView2010*, 2010.

[10] T. Hoßfeld, F. Liers, T. Volkert, and R. Schatz, “FoG and Clouds: Optimizing QoE for YouTube,” in *KuVS 5thGI/ITG KuVS Fachgesprch NG Service Delivery Platforms*, Munich, Germany, oct 2011.

[11] K. Lai and M. Baker, “Nettimer: A tool for measuring bottleneck link,” in *Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems*, 2001.

[12] T. En-Najjary and G. Urvoy-Keller, “Pprate: A passive capacity estimation tool,” in *Proceedings of E2EMON*, 2006.

[13] M. Carbone and L. Rizzo, “Dummynet revisited,” *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 12–20, April 2010.