Modeling the YouTube stack: From packets to quality of experience

Florian Wamser[a], Pedro Casas[b,c,*], Michael Seufert[a], Christian Moldovan[d], Phuoc Tran-Gia[a], Tobias Hossfeld[d]

[a] *University of Würzburg, Institute of Computer Science, Am Hubland, 97074 Würzburg, Germany*
[b] *FTW – The Telecommunications Research Center, Donau-City-Straβe 1/3, A-1220 Vienna, Austria*
[c] *AIT – Austrian Institute of Technology, Donau-City-Straβe 1/4, A-1220 Vienna, Austria*
[d] *University of Duisburg-Essen, Universitätsstraβe 2, 45141 Essen, Germany*

1. Introduction

YouTube is one of the most popular services in today's Internet and is responsible for more than 20% of the overall Internet traffic [9], including mobile. Every minute, 100 hours of video material are uploaded and more than 1 billion unique users visit YouTube each month [10,11]. YouTube's enormous popularity introduces severe challenges for network operators, who need to design their systems properly in order to cope with the high volume of traffic and the large number of users. Mobile operators are particularly sensitive to these challenges, as YouTube traffic is rapidly increasing in mobile networks, with more than 40% of all YouTube views coming from mobile devices today [10]. Since network operators need to offer satisfying video quality levels to prevent clients from churning, YouTube is an important application which has to be considered by operators, both in current highly competitive mobile and fixed broadband markets.

Consequently, Internet Service Providers (ISPs) do not only try to cope with a service like YouTube in the network. Instead, they actively include it in considerations of network optimization and operations. Thus, Quality of Service (QoS) provisioning is done in the network to meet requirements and provide guarantees, primarily, in order to ensure a good application quality. All these provisions strictly need a comprehensive understanding of the YouTube service in the network. More precisely, not only bandwidth requirements or minimum latency needs to be known, but also the impact of user interaction, the adaptation capabilities of the service, or possible implications for the users are required to be known. In short, for a specific consideration of YouTube in a network, one must understand YouTube at all its layers.

[*] Corresponding author. Tel.: +43 664 672-56-60; fax: +43(0) 50 550-28-13.

E-mail addresses: wamser@informatik.uni-wuerzburg.de (F. Wamser), pedro.casas@ait.ac.at, pecasas@gmail.com (P. Casas), seufert@informatik.uni-wuerzburg.de (M. Seufert), christian.moldovan@uni-due.de (C. Moldovan), trangia@informatik.uni-wuerzburg.de (P. Tran-Gia), tobias.hossfeld@uni-due.de (T. Hossfeld).

In this paper we present three models to explain the main features of YouTube in the network and for the user. We focus our attention on the buffer level of YouTube at the application layer. We show that both the flow control at the network level and the user Quality of Experience (QoE) is directly related to the buffer level, making it to the central point of contact for a possible resources management or possible optimizations [8,12]. The other way round, this also means that with the ability to monitor or estimate the buffer level, a key element of YouTube is known and far-reaching conclusions can be stated about QoE, packet flow, and application properties.

Several works have been carried out in the research community to characterize and investigate YouTube. In [13], authors characterize the YouTube traffic and investigate correlations between network and user behavior; the underlying infrastructure of YouTube is studied in [6,14]; YouTube characteristics in mobile (cellular) networks are investigated in [15], whereas YouTube performance degradation events are detected and diagnosed in [16,17]. Quality of Experience , i.e., the quality perceived by end users of YouTube, was evaluated both in controlled lab studies [18,19] and in field trials [20].

We revisit in this paper the YouTube application through an exhaustive end to end study of the service, considering every layer from the network traffic up to the QoE as perceived by end-users. We start by characterizing the way YouTube servers send video flows to the end-users, describing and modeling the flow control mechanism currently used by YouTube. An initial view to YouTube's flow control was provided by [14] in 2011, but since then the mechanism used by YouTube has much evolved, moving from a server-based control paradigm to a client-based one. To the best of our knowledge, we are the first in characterizing and modeling the new flow control mechanism used by YouTube.

Going a step further from the network to the client, we analyze how the video traffic is consumed and played-back at the client YouTube player, additionally deriving models to assess the quality directly perceived by the end-user from the player state. Previous studies [19,21] have shown that both the number of stalling events and their duration are the most important features influencing the final QoE undergone by the end-user. A stalling event corresponds to the interruption of the video playback due to the depletion of the playback buffer at the YouTube player. When the available bandwidth is lower than the required video bitrate, the playback buffer becomes gradually empty, ultimately leading to the stalling of the playback. Using both controlled lab studies [18] and field studies [20], we conceive a model which can map stallings to end-user QoE.

Finally, at the end of our analysis, a large-scale YouTube QoE monitoring system based on passive traffic analysis is presented. Once we have analyzed and understood how YouTube works from the server to the end-user perception, we devise a real-time monitoring system to extract YouTube performance indicators related to the QoE perceived by end-users, relying exclusively on packet-level measurements.

Overall, this paper provides a complete overview on how today's YouTube application works at both the server and the client side. Table 1 lists the contributions and assigns them to the respective layers, from transport level to the user level. In addition, a list of potential network operational "interests" provides a possible application of the models for network operators in the practice. The main objective of the study is to provide network operators with tools to model YouTube traffic and the client-side player for performance evaluations, and to generally assess how good they are doing in delivering the right QoE to their end-customers watching YouTube videos.

This work is structured as follows: Sections 2 and 3 outline background and related work for YouTube video streaming, and give an overview of different streaming approaches with emphasis on progressive streaming techniques. Among other things, the evolution of progressive streaming is shown, including the paradigm shift and its reasons. Section 4 describes the tools and data sets used in the modeling study at the different layers. In Section 5 we present an overview on the proposed YouTube models at the network, application, and user layers respectively, going into deeper details and analysis of the network layer and YouTube flow control model in Section 6. Sections 7 and 8 focus on the YouTube player and YouTube QoE models respectively; in addition, YouTube QoE-based monitoring in real mobile networks is also discussed in Section 8. Finally, we draw conclusions in Section 9.

2. Starting from scratch: Background on YouTube video streaming

YouTube is a streaming platform mainly offering small to medium-sized video clips to its users. The encoding of the video clips is done according to the H.264/MPEG-4 Advanced Video Coding (AVC) as default video compression format. YouTube uses HTTP(S) as streaming protocol for the videos. The client application is a precompiled Adobe Flash player assembly which runs in the web browser. It essentially downloads the video data over at least one HTTP(S) connection and already starts playing the video while the download is still ongoing. Downloaded data is stored in the memory or in a temporary file which serves as buffer for video playtime. In this context, YouTube employs buffering which means that the client starts playing out data from the buffer only after a certain level of playtime has been stored. The time from requesting a video until the buffer is sufficiently filled to start playing the video is called *initial delay*. While the video is playing the server fills the buffer by transmitting blocks of video data to the client. *Stalling*, i.e., an interruption of the video playback, occurs if the playtime buffer becomes empty during video playback. The actual data arrival at the client is governed by TCP and depends on the available bandwidth. These two transmission phases, the initial filling of the buffer before the video starts playing, and the filling of the buffer while the video is already playing, are dictated by the *YouTube flow control algorithm*. Video streaming over the Internet based on HTTP(S) has evolved considerably in recent years:

Progressive download: in the beginning, HTTP-based video streaming was implemented using a simple file server providing the video file for the users. To avoid the resulting waiting time for the download, the video playback was started already during the download. It was done even though the file was not completely available on the client device. This principle marks the beginning of the so-called *progressive download*. The advantage is that the existing, well-established infrastructure can be used. The disadvantage is however, that a best effort download is not always automatically sufficient for streaming video. The download usually has to be adjusted with complicated means by the server (server-based streaming) to satisfy the video requirements, user demands, and network conditions in order to carry out a useful streaming. There are three essential requirements for a video streaming server to consider for a useful and efficient streaming: first, it has to adjust the download rate according to the video content encoding. Second, it must estimate or know what capabilities (screen size, hardware support, etc.) the client device offers. Third, the server must estimate the current access network conditions of the user and adjust the download accordingly. The crucial point is that, even though the server knows the video content, it does not directly know the conditions at the client side, which can be very different depending on the user preferences, the end device, and the network.

Table 1
Comparison of the potential interests of network operators and the contribution in this paper.

	Network operator interests	Contribution in this work (and reference to the relevant section)	
Network level	**YouTube CDN structure** *Estimate amount of traffic and traffic flow, selective content caching according to network characteristics*	- (see related work in Section 3, references [1–6])	
Transport evel	**YouTube flow control** *Insights into traffic patterns, per-flow optimization for YouTube, flow-level impact on other applications*	**Network traffic model**	Section 6
Application evel	**Video Playback Behavior** *Optimization taking into account the playout behavior of the YouTube player, see e.g.* [7,8], *buffer management, video quality optimizations*	**Video player application model**	Section 7
User evel	**User-perceived quality** *Network performance monitoring,**see** Section 8, user-centric optimizations to avoid user churn*	**User QoE model**	Section 8

YouTube, like many video service platforms, meets these requirements by offering videos in different bit rates, more precisely, in different resolutions. Still, the drawback is that the server is not aware of the user conditions, and must estimate which quality should be delivered in order to achieve a smooth playback. Another problem for video service providers are users which abort the playback of a video [13,22]. In order to fill the client's buffer to compensate for changing network conditions, the server transmits video data in advance. When the video is aborted, all data in the video buffer is discarded, and thus, was transmitted in vain. The video service provider can save resources by (ideally) only transmitting video data that will eventually be watched. This is precisely what is generally referred to as *streaming*.

Client-based streaming with progressive download: To apply the streaming principle for progressive downloads, the servers (which until then controlled the download) are superseded by the clients, which take over control of the streaming and request video data from the servers. Clients only request data if their buffer is below a critical threshold. The other way round, above this threshold, no more data is requested. This limits the amount of video data which is wasted in case of playback abortion, and thus, also the amount of data that is transmitted in vain by the servers. This results in a more efficient resource utilization for the video service provider.

Adaptive client-based streaming (HTTP adaptive streaming): To be more flexible and to avoid the need to estimate the client's network conditions, the paradigm change evolved even more. Nowadays, video files are no longer stored as a whole file for each quality level (i.e., bitrate). Instead, they are stored as many files, so-called chunks or segments, consisting only of a few seconds of playback time each. The mapping of files to video parts is specified in a dedicated media description file. This allows for a more fine-grained quality selection (adaptive streaming). In combination with client-based streaming, the benefit arises since the client directly knows about its current network condition. It is aware of its physical layer for data transmission. Further on, it also knows about the video streaming options due to the media description file. Consequently, adaptive client-based streaming can seamlessly adapt to changing network conditions by requesting video chunks of appropriate bit rates from the server. This results in the highest possible playback quality and reduced number of stalling events which increases users' QoE [23]. Many proprietary solutions implemented this new client-based adaptive streaming paradigm. Recently, also YouTube followed the current trend by integrating the standardized adaptive streaming technology MPEG Dynamic Streaming over HTTP (DASH) [24].

3. Related work

Related work for YouTube can be divided into different areas of research. There is, first of all, work on the content delivery infrastructure at network level of YouTube [1–6]. In order to avoid bottleneck links in the networks and to bring video content closer to end users, video service providers federate many servers and data centers to form a content delivery network. The videos are distributed among the servers based on certain criteria (local popularity, time of day, etc.), and when a user requests a video, a server close to him transmits the video. For example, it is shown in [3], how the YouTube video player selects the content servers. Based on the insights about the current CDN structure, in [25,26], it is considered how an efficient caching for YouTube can be achieved.

In terms of our work, particularly the transmission characteristics of YouTube for the end-to-end transport is interesting. In [27], network characteristics of the two most popular video streaming services, Netflix and YouTube are presented. The authors show that the streaming strategies vary with the type of the application (web browser or native mobile application), and the type of container (Silverlight, Flash, or HTML5). In particular, they identify three different streaming strategies that produce traffic patterns from non-ack clocked ON-OFF cycles to bulk TCP transfer. Furthermore, they present an analytical model to study the potential impact of these streaming strategies. In [28], a YouTube server traffic generation model is proposed. The derived characterization and model are based on experimental evaluations of traffic generated by the application layer of YouTube servers. Another well-known paper about the YouTube flow control is [29]. Alcock introduces here in detail the streaming behavior of the YouTube throttling algorithm. In [30], the characteristics of mobile YouTube traffic are analyzed.

Going a step further, the question is how the CDN structure and the flow control finally influences the user perceived quality at the end device. In both works [19,21], the authors state that the impact on the user quality depends on the stalling duration and the number of stalling events. Consequently, for an optimization in the sense of the user, the focus must be on the buffer of the video player at the individual user. In [18], moreover, the influence of the initial waiting for filling the buffer is discussed.

Several works propose optimization for a network, based on the previous insights [8,12,31–34]. In [12], both network-level resource management for YouTube as well as content and client-based control for access network with YouTube traffic are proposed. The authors of [8], propose a special QoE-oriented scheduling in the air interface of OFDMA mobile communication networks. Prerequisites for all optimization is the knowledge of how well YouTube currently behaves in a network. Various monitoring solutions have been presented such as [35–38].

Finally, there are numerous statistics and characterizations of the YouTube content and the usage of YouTube, both in the cellular environment as well as for normal computers [13,15,39–41]. For example, in [13] usage patterns, file properties, popularity characteristics, and transfer behaviors of YouTube are presented, and compared to traditional web and media streaming workload characteristics. In [41], the video traffic generated by three million users across one of the world's largest 3G cellular networks is measured.

General papers with guidelines about monitoring, management of YouTube, and optimization for YouTube within the network are presented in [7,12,33,42–44].

As we explain in next sections, our paper builds on top of our many previous studies on YouTube at the multiple layers, from network traffic to QoE. While some of the results of the individual studies presented in this paper have been partially presented in previous work [18,20,43–45] (in particular, the YouTube QoE models presented in Section 8), the additional value of this paper is to provide a single source of information compiling all these studies into a single and complete modeling effort of probably the greatest and more complex Internet-scale service ever, which is changing even the structure of the whole Internet. We expect this paper to be highly useful for ISPs willing to provision YouTube and future similar video streaming services through their networks, allowing them to better understand how to dimension and manage their networks in order to correctly provision this surge of services.

4. Measurement tools and data sets

For the characterization of YouTube, three different types of measurements are used. Each type of measurement operates on different layers and measures different properties from YouTube. First of all, network packet traces were carried out while playing YouTube videos. Second, video and player information during playback were retrieved at the client using the official YouTube player API. They were stored in a second data set. Third, the results from a set of QoE subjective tests performed in our previous work, as well as measurements performed at the core of a mobile network were used to model and quantify the overall user satisfaction of YouTube. Table 2 enumerates the list of conducted measurements and assigns them to the models described in this study.

4.1. Packet traces

For comprehensive packet traces, measurements in two different time periods were performed. The first measurements were carried out in November and December in 2012 (22.11.2012–22.12.2012) at the University of Würzburg. Here 84 videos were measured, and the packet arrival times and packet sizes were passively monitored. In order to investigate in this work as many, but also popular videos, the YouTube videos were selected as follows. A YouTube search was done according to a random dictionary entry. The first two search results are accepted for each search. That way, only popular videos get selected. This method is used until 100 different videos are selected. Some of the videos did not exist anymore when we tried to watch them or could not be replayed for different reasons. This reduced the number of videos that are finally measured to 84, which were replayed several times at the Campus network of the University of Würzburg. The network bandwidth was not limited.

The second series of measurements were carried out in April 2015 (23.03.2015–15.04.2015).[1] Here exactly 1060 videos were randomly selected and watched. Some videos were selected twice, which resulted in 1002 different videos. This data set includes approximately 13 million log items. The network bandwidth was limited to 3 Mbit/s, 1 Mbit/s, 700 kbit/s. These values were selected empirically, to specifically cover the different events of the YouTube streaming under study. There are 384 videos measured with a limitation of 3 Mbit/s, 331 videos were measured at a limit of 1 Mbit/s, and 345 videos were measured at a limitation of 700 kbit/s. There are about 16 GB transferred video data and about 258 h of video content.

Finally, the study focuses on three standard video resolutions: 240p, 360p and 480p. The first two resolutions correspond to the most popular ones in the 2012 dataset, whereas the 480p resolution was added for completeness. While it is true that higher resolutions were available for the 2015 measurement campaign, we took a conservative approach to keep consistency among both datasets.

4.2. Video player measurements

YoMo [35,36,46] is a YouTube monitoring tool for the YouTube video player developed at the University of Würzburg. It is used to analyze the buffered playtime and the playout behavior of YouTube. YoMo uses a Mozilla Firefox extension which retrieves data from the YouTube player. In general, it works as follows. Since the buffered playtime is monitored, it knows if a video is playing or stalling. If a new video is requested, the YouTube player opens a new TCP connection to download the video file. A signature is contained within the header of the video file which is detected by YoMo. As YoMo looks also at all other TCP flows to a YouTube server, the flow that contains the signature can be identified. YoMo then investigates all of the data of this specific TCP stream. Since the video tags are parsed in real-time, YoMo can return the buffered playtime of the video. For the new YouTube algorithm, YoMo especially parses the packets that contain the HTTP-request over a certain range of bytes. The playtime is sent to YoMo from the plugin which in return retrieves it from the YouTube player API. To overcome end-to-end encryption, a simple state of the art man-in-the-middle attack was done at the measurement client. A more detailed insight of YoMo is given in [32,36].

YoMo was used to study YouTube videos. The application data were collected in conjunction with the first measurement of the packet traces. The scope and duration of the measurement corresponds exactly to the specified information in the previous subsection. The measurement was conducted in the period 22.11.2012 to 22.12.2012 for 84 videos. Several samples had measurement errors, e.g., some HTTP requests for parts of the video content were not received or were received multiple times. We omitted these runs from the analysis and ended with 778 measurement samples.

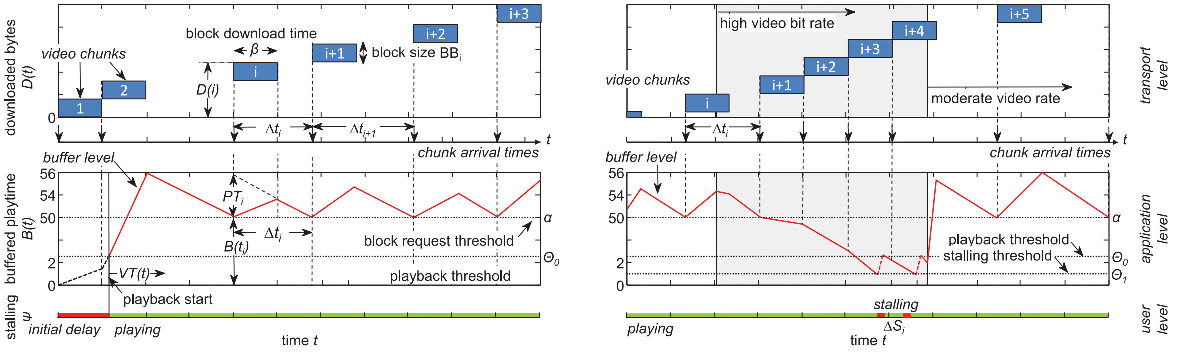
4.3. YouTube QoE and Performance monitoring studies

The third dataset is composed of the results obtained from several subjective YouTube QoE studies we have conducted between 2011 and 2014 [18,20,21,43], which address the influence of the most relevant characteristics of the YouTube service on the quality as perceived by the end users. This dataset is complemented with real network and QoE-based measurements we have performed at a nationwide mobile operator [43]. These are composed of YouTube videos streamed through cellular connections, which are passively captured at the packet level in the core of a mobile network, and additionally stamped with the real QoE feedback provided by the users actually watching the videos.

[1] Dataset is available at http://youtubedb.informatik.uni-wuerzburg.de.

Table 2
List of measurements and underlying data sets.

	Data set	Corresponding model
Transport level	(1) Packet traces	*Network Traffic model*
Application evel	(2) Player measurements [35,36,46]	*Video player model*
User evel	(3) Stalling and performance monitoring studies [18,20,21,43]	*user QoE Model*



(a) The normal streaming case with a sufficient buffer level: actual video rate is smaller than the download rate.

(b) Video rate is higher than the download rate, resulting in stalling.

Fig. 1. Modeling of YouTube at the transport, application, and user level.

Table 3
Input parameters and variables used in the model.

Name	Description	Unit
VS	Size of video	[B]
r	Resolution of video	[p]
$br(VT)$	Video bit rate at playtime VT	[B/s]
n	Number of blocks	-
BS_r	Maximum size of a block for resolution r	[B]
BB_i	Size of block i	[B]
C	Bandwidth/Download capacity	[B/s]
β_i	Download time of block i	[s]
t_i	Request time of block i	[s]
Δt_i	Inter-arrival time between block i and block $i+1$	[s]
PT_i	Playtime of block i	[s]
$D(t)$	Downloaded bytes at time t	[B]
$B(t)$	Buffered playtime at time t	[s]
$DT(t)$	Downloaded playtime at time t	[s]
$VT(t)$	Video playtime at time t	[s]
Θ_0	Playing threshold	[s]
Θ_1	Stalling threshold	[s]
α	Block request threshold	[s]
ψ	Stalling indicator	{0, 1}
N	Number of stalling events	-
$S(t)$	Stalling time at time t	[s]
ΔS_i	Stalling time between t_i and t_{i+1}	[s]

5. Modeling YouTube

We now introduce first a model for YouTube considering different layers. We describe the model components, the variables, and the input parameters that are needed for the overall model.

In order to take the *traffic on transport layer*, the *YouTube application* in the browser as well as the *quality perception of the user* into account, we divide our model into three parts. All the parts are interconnected via a central parameter, the buffer level at application layer, which is discussed below. The input parameters and variables which are used in the model are summarized in Table 3. They are introduced in detail in the following paragraphs:

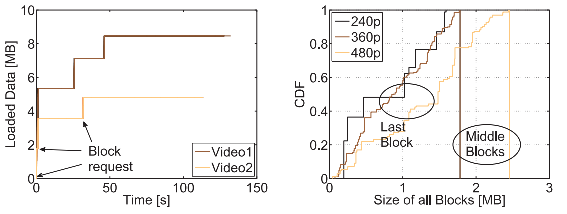
Transport level: At the transport level, we describe the transfer of data over time. Since YouTube uses a block-wise transmission, we specify here when a block is transmitted and how large it is. We consider a video of size VS which consists of n blocks. As depicted in the upper part of Fig. 1, the following parameters are used: (1) downloaded bytes $D(t)$, (2) block size BB_i, and (3) block inter-arrival time Δt_i between two blocks. The downloaded bytes $D(t)$ depend on the time t and are shown on the y-axis in the subfigures in the upper part of Fig. 1. Furthermore, block i is BB_i bytes in size and is requested at time t_i. All blocks are downloaded over a link with capacity C in bytes per second, such that β_i is the download time of block i in seconds with $\beta_i = \frac{BB_i}{C}$.

Application level: At application level, we describe the playout behavior of the YouTube player at the client, particularly the buffered video playtime $B(t)$, the video thresholds Θ_0 and Θ_1, and parameter α are introduced. $B(t)$ is illustrated on the y-axis in the middle part of Fig. 1, as well as in Fig. 4 later on. In order to calculate $B(t)$, the model takes into account the downloaded playtime $DT(t)$ and the video playtime $VT(t)$, such that the following holds at any time:

$$B(t) = DT(t) - VT(t) \tag{1}$$

The downloaded playtime $DT(t)$ is calculated from the sum of downloaded playtime PT_i per block i. In particular, PT_i depends on the video bit rate $br(VT)$, which is a continuous function of the video playtime that determines the amount of bytes needed to play out the video at time VT. If $B(t)$ is larger than Θ_0 the video playback starts, cf. Fig. 1(a). Blocks are requested steadily as long as the buffered playtime is smaller than the threshold α. Then, no blocks are requested until the buffered playtime drops below α again. This means the application parameter α controls the generated network traffic. Once the buffered playtime is lower than Θ_1 the playback is interrupted until enough data is in the buffer, i.e., stalling occurs if Θ_1 is reached.

User level: Based on the findings of [47], we use a simplified approach in order to model the impact on the user. We quantify the impact as number of stalling events and total stalling duration while watching the video. We introduce a boolean variable ψ which indicates whether the video is playing ($\psi = 0$, green) or stalling ($\psi = 1$, red). $S(t)$ indicates the total stalling duration until time t. The stalling is illustrated in the bottom subfigure of Fig. 1(b). Fig. 1 illustrates two cases and outlines (1) the course of the downloaded bytes, (2) the course of the buffer level of the video player, and (3) whether the video is stalling or not.



(a) Bytes loaded over time for two example videos with 12 samples. Notice that corresponding samples of each video are on top of each other.

(b) CDF of the block size of the last block (Last Block) and CDF of the block size of all blocks except for the last block (Middle Blocks).

Fig. 2. Study on the download behavior. Downloaded data over time and cumulative distribution function for the block size.

Fig. 1(a) shows the general case if a sufficient downloading speed is available, i.e., the current video bit rate is smaller than the maximum download rate. The figure shows an exemplary download of a YouTube video over time with a fixed download capacity C. The upper part shows the downloaded bytes $D(t)$, the middle part shows the buffered playtime $B(t)$, and in the bottom part, ψ indicates whether the video is playing. The first blocks of the video are downloaded steadily and during the initial delay phase, in which the video is not playing, each block adds its contained playtime to the buffer. After a certain amount of playtime is stored in the buffer, the video playback starts and the buffered playtime increases more slowly, as data is played out of the buffer at the same time. After the download of Block 2, the buffered playtime has surpassed α and thus no immediate block request occurs. Only after $B(t)$ drops down to α due to the playback, a new block is requested at transport layer. This oscillating behavior of $B(t)$ continues until the end of the video. In Fig. 1(b), the contained playtime in the blocks 5–8 is lower than time needed to download the blocks. Thus, the playout buffer empties which eventually results in periodic stalling. Finally, Block 9 then contains enough playtime again, in order to present the video without any interruptions. In the following sections, each level is described in detail.

6. YouTube flow control: A network traffic model

Since early 2012, YouTube uses a new algorithm to transfer videos to the users. We investigate this algorithm in the following as the basis for subsequent modeling for the transport level traffic. The performed measurement is combined with a second measurement of the buffer level at the application layer, which gives insights into the behavior of the video player and the quality presented to the user.

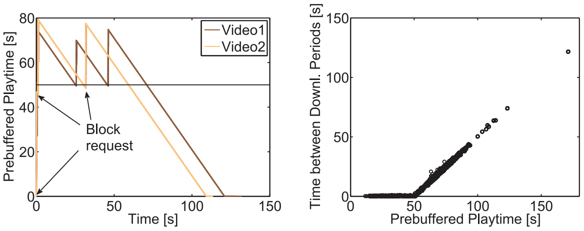
6.1. Download behavior

In this section, we describe the general findings about the YouTube player and streaming. Our results show that YouTube employs a block-wise download behavior. First of all, we define and measure the block size. Next, we investigate the exact points in time when blocks are requested. For this purpose, in particular the measurement of application parameters such as buffer level and video resolution is required.

6.1.1. Blockwise download

The flow control algorithm manages the way data is downloaded. Fig. 2(a) shows the cumulative data downloaded over time for 12 samples of two example videos (Video1, Video2).

There are major differences compared to the download of videos with the old algorithm [29]. For example, there is no longer a consistent download of the video data. Instead, Fig. 2(a) shows that there are long periods where no data is downloaded. Furthermore, there are short periods of time where a lot of data is downloaded very fast. We call the data that is downloaded in such a short period a *block*. From our measurements, we see in Fig. 2(a) that each video consists of different blocks which are downloaded at different times. However, the measurement curves of a specific video are located one above the other, which means that the same block of a video was always downloaded at approximately the same time. In order to initiate the download of a block, a HTTP-request for a certain range of bytes is sent to the YouTube video server. In the following, we use the terms *GET-request* or *block request*. No such request of the same resolution is sent as long as a block of one resolution is still being downloaded.



(a) Buffered playtime over time for two 360p videos.

(b) Time between two consecutive block downloads at different buffer levels.

Fig. 3. Studies on the request time of the blocks.

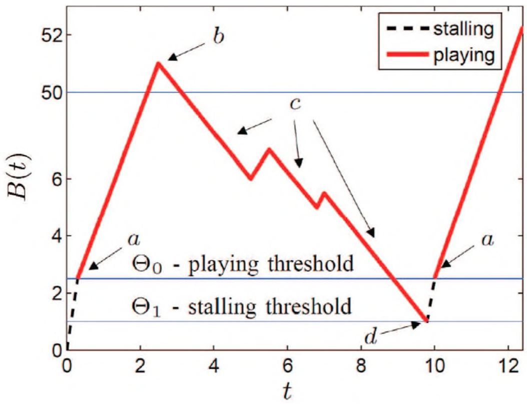


Fig. 4. Management of the playback buffer in YouTube.

6.1.2. Block size

Having analyzed the blockwise download behavior, we now discuss the size of the blocks. In Fig. 2(b), we see how the size of the blocks is distributed. For a resolution of 240p and 360p, a block of 1.78 MB (rounded value) is requested. For 480p, a block of 2.46 MB (rounded value) is requested. The header of the video container, which has a size of 13 B, is not downloaded because it is the same for every video and the player already has this information. Hence, the size of the first block is 13 B smaller than the standard block size. In our measurements the last block is always smaller than the other blocks since the size of our selected videos is not an exact multiple of the normal block size. Instead, the size of the last block is equal to the video size minus all previous blocks minus 13 B. The resulting value follows a uniform distribution as expected because all of our videos contain more than one block.

6.1.3. Request time of blocks

Next, we analyze the exact time when a block is requested. Fig. 3(a) shows the buffered playtime in seconds over time for two example videos. We observe that in the first two seconds, a big part of the video (over half of the total playtime) is downloaded. The requests for the remaining data are sent each time the buffered playtime drops to approximately 50 s.

To investigate this in more detail, we measure the buffered playtime right after the block download is completed. In Fig. 3(b), this value is compared to the time between the end of a block download and the next request. The results can be separated into two parts at around 50 s. If the buffered playtime is under 50 s, the time between two downloading periods is independent of the buffered playtime as the next block is requested almost instantly after the previous block download finishes. If greater than 50 s, the buffered playtime seems to correlate linearly with the time until the next block is requested.

In order to interpret the results at about 50 s accurately, we further consider in the following the results in detail. After fitting these values for videos with 240p, 360p, and 480p separately, we notice that all resolutions return very similar results. For a buffered playtime greater than 52 s, we fit the values to a linear function with an average slope of $a = 1.00$ and an average y-intercept of $b = -49.82$ s. Here, only an insignificant deviation of less than 2% between the different resolutions is observed. We assume that this value is pre-configured by YouTube. Hence, it seems reasonable to accept that the next block is requested in (bufferedplaytime) – 50 s in average after the previous block download. For a buffer level of under 48 s, the average time between two requests is 0.3 s in our measurements. Since blocks of one resolution are always downloaded successively, it takes at least the download time of a block until a new block is requested again. Therefore, the time between two downloading periods is influenced by the download speed of the Internet connection at our local network.

6.1.4. *Dynamic adaptive streaming over HTTP*

Since 2013, YouTube supports the dynamic adaptive streaming over HTTP (DASH) protocol, which was first presented in [48]. With DASH, the player switches to a lower video resolution if there is little data in the buffer and if it is not growing quickly. Next, if the buffer contains a lot of data or if it is growing quickly, the video resolution may be increased. A first complete analysis of DASH in YouTube was done in [49]. There are many possibilities on how to optimize adaptive streaming. A comprehensive study that discusses relevant adaptation algorithms is given in [50]. However, exact video adaptation strategies are out of scope of this paper and will not be discussed any further. Nevertheless, any future adaptive approach may easily be integrated into the model presented in this paper.

6.2. *Flow control model*

Finally, we create a model of the Range algorithm for the flow control of a YouTube video. With this model, the network traffic can be simulated, while a video is viewed. Each video is defined by its size *VS*, its resolution $r \in \{240\text{p}, 360\text{p}, 480\text{p}\}$, and its bit rate *br*(*VT*). Based on these three characteristics, we define the flow control model for YouTube traffic, i.e., we describe when and how much traffic is generated by streaming this video. For the sake of simplification, we assume that the download capacity *C* does not change during a video download and that the YouTube servers are always able to saturate the users' download capacity. Further, we consider the bit rate to be constant within a block. Finally, we assume that the download of the first packet of a block starts without delay after the request. Based on these assumptions, the model can be described in detail.

The resolution determines the maximum size *BS* of a block according to

$$BS = \begin{cases} 1.78\,\text{MB} & \text{if } r = 360\text{p or } 240\text{p}, \\ 2.45\,\text{MB} & \text{if } r = 480\text{p}. \end{cases} \tag{2}$$

The maximum block size *BS* is the size of all blocks except for the first and last block. This is due to the fact that the first 13 B of a video file are always known, hence, they need not be downloaded. The last block usually is smaller than the maximum block size, as it contains the remaining bytes of the video. This gives the block sizes for all *n* blocks of the video:

$$\begin{array}{ll} BB_1 = BS - 13\,\text{B} & \\ BB_i = BS & \text{for } 2 \le i < n \\ BB_n = VS - \sum_{j=1}^{n-1} BB_j & \end{array} \tag{3}$$

Next to the block size, the request times t_i of block *i* are important to fully describe the network traffic. Therefore, we update several variables at each block request t_i. We assume the first block is requested at time $t_1 = 0$ and consider in the following the time difference $\Delta t_i = t_{i+1} - t_i$. From our measurements, we found that the next block is downloaded if the buffered video time $B(t)$ is less or equal to a threshold α. In order to compute the next block request t_{i+1} from the previous request time t_i it is necessary to consider the amount of playtime PT_i contained within block *i*. If the buffered playtime is lower than α after the download of block *i*, the next block is requested immediately. This means, the time between block *i* and $i+1$ is only the download time β_i of block *i*. If the buffered playtime is larger than α after the download of block *i*, the next block request occurs when the buffered playtime decreases down to α. Thus, the time Δt_i between the requests of block *i* and block $i+1$ can be computed as described in Eq. (4):

$$\Delta t_i = \begin{cases} \beta_i & \text{if } B(t_i) + PT_i < \alpha + \beta_i, \\ B(t_i) + PT_i - \alpha & \text{otherwise}, \end{cases} \tag{4}$$

where the download time of block *i* $\beta_i = BB_i/C$ is negligible for very high download capacities. While α lies between 48 s and 52 s in almost any download sample, we recommend an approximation of $\alpha = 50\,\text{s}$ according to our investigations in Section 6.1.3.

To compute the buffered playtime, we use the formula

$$B(t_i) = DT(t_i) - VT(t_i) \tag{5}$$

$DT(t_i)$ refers to the sum of the playtime which is contained in the previously downloaded blocks and can be calculated recursively as

$$DT(t_i) = \begin{cases} 0\text{s} & \text{if } i = 1, \\ DT(t_{i-1}) + PT_{i-1} & \text{otherwise}. \end{cases} \tag{6}$$

$VT(t_i)$ refers to the amount of time from the video that has been played out until block *i* is requested. While the video time is initially 0, at each block request t_i, $i \ge 2$, the video time can be calculated by adding the time between the block requests Δt_{i-1} but subtracting the time during which the video was not playing (i.e., initial delay and stallings) in that interval. As $S(t)$ is the total stalling duration until time *t*, $\Delta S_{i-1} = S(t_i) - S(t_{i-1})$ is the sum of the length of all stalling events (including startup delay) between t_{i-1} and t_i.

$$VT(t_i) = \begin{cases} 0 & \text{if } i = 1, \\ VT(t_{i-1}) + \Delta t_{i-1} - \Delta S_{i-1} & \text{otherwise}. \end{cases} \tag{7}$$

We consider the model for the flow control at distinct time points when a new block is requested. Thus, stalling can only occur if the video playtime at the next block request time t_{i+1} is larger than the downloaded playtime at that time $DT(t_{i+1})$. In this case, the previous stalling time can be computed as the difference between the needed playtime $VT(t_i) + \Delta t_i$ and the actual available playtime $DT(t_{i+1})$:

$$\Delta S_i = \begin{cases} 0\,s & \text{if } VT(t_i) + \Delta t_i < DT(t_{i+1}), \\ VT(t_i) + \Delta t_i - DT(t_{i+1}), & \text{otherwise.} \end{cases} \tag{8}$$

The presented model allows for a simulation setup which computes the time and amount of traffic generated by a YouTube video. Moreover, the occurred stalling time can be calculated which can be used to estimate the perceived quality of the video streaming. The needed input parameters are video size VS, resolution r, contained video playtime per block PT_i, and download capacity C.

7. Video player application model: Measuring and modeling the YouTube player

In previous sections we have explored YouTube from a network perspective, measuring and modeling the flow control mechanisms it uses to deliver the videos through the network. Grasping such mechanisms is paramount for ISPs, both to understand the impact of YouTube traffic on their networks, as well as to assess the traffic delivered to the customers' end-devices. With a proper assessment, traffic bottlenecks in the network can be avoided and network issues can be resolved. Let us now turn from the *network* to the *application* and shed light on how the video flows are consumed at the end devices, which ultimately defines how the customers perceive the YouTube service. This is in fact the most important part of the end-to-end YouTube provisioning for a network operator: how good or bad is the YouTube quality as experienced by the customers.

For doing so, we measure and model the YouTube player buffering and playback behavior. Later on in Section 8 we investigate how to estimate the quality experienced by YouTube users, providing a QoE model translating the behavior of the YouTube player into a measure of user satisfaction.

As explained in Section 5, the YouTube player works with an internal playback buffer where the chunks of video being downloaded are stored at and played from. During the simultaneous downloading and playback, the buffer grows and shrinks depending on the download bandwidth and the video bitrate. Intuitively, when the download bandwidth is lower than the video bitrate, the playback buffer becomes gradually empty, ultimately leading to the stalling of the playback. When the buffer runs empty, the video stalls and the YouTube player state changes from “playing” to “stalling”, until more video chunks are received and buffered.

The YouTube player model uses the parameters presented in Section 5: it considers the buffering-based thresholds Θ_0 and Θ_1 to control the way video frames are consumed from the playback buffer. To further explain the YouTube player behavior in the practice, Fig. 4 describes an artificially generated video playback scenario, in which a video is displayed under heavy downlink congestion, forcing the player to visit all its states (note that this figure is a particular case of Fig. 1). At time $t = 0$ the player buffer is empty, and video blocks are requested to the server. The video starts playing immediately after the buffered video playtime $B(t)$ exceeds the *playing threshold* Θ_0, flagged as event (a). Video blocks are downloaded from the server as long as $B(t)$ is below 50 s (see Section 6.1.3 for a detailed explanation). Event (b) flags the end of this buffering/blocks-request period. The playback of the video continues without additional buffering activity as long as $B(t)$ is above the aforementioned 50 s threshold. Events flagged as (c) correspond to a heavy downlink congestion situation, in which the player requests additional video blocks but the video content gets to the player only sporadically, i.e., the video buffer slightly increases. The video playback continues until $B(t)$ falls below Θ_1, flagged as event (d). The player remains in stalling state until $B(t)$ exceeds Θ_0.

7.1. The model in the practice

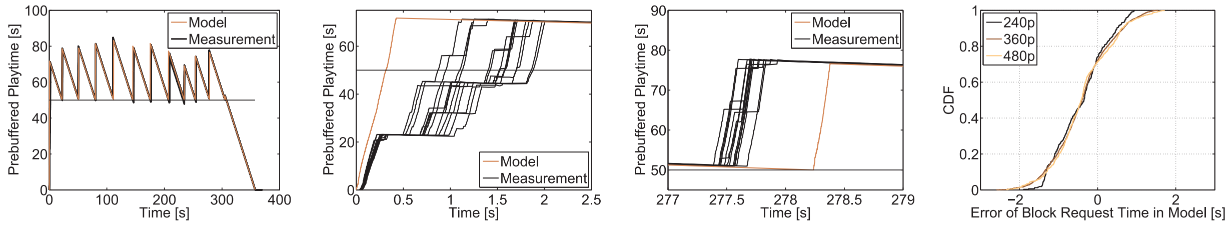
To verify the applicability of the proposed model, we compare the buffered playtime $B(t)$ as measured for a video watched in the desktop YouTube player with the buffered playtime $\hat{B}(t)$ estimated by the model through a simulation. For this purpose, the data set and the measuring methodology described in Section 4.2 are used. The same video is downloaded/replayed 14 times under perfect network conditions (i.e., bandwidth is high enough to avoid stalling), to account for potential network performance variations. Fig. 5 shows the buffered playtime over time. Colored in black, we see the result of 14 measurements of the same video on top of each other. Colored in brown, we see the result of the simulation. To simplify the evaluation, we assume that the startup delay of the video playback is zero in the model, i.e., we take $\Theta_0 = 0$. Given that no stalling is observed for this video, we also set the stalling threshold $\Theta_1 = 0$.

There is only one curve for the simulation since it is deterministic. We first notice that the measurement and the simulation seem to be very close to each other, as depicted in Fig. 5(a). However, there are some deviations which are visible if we look at the curves in detail in the following two figures. In Fig. 5(b), the buffered playtime over time in the initial phase is depicted for the measured data and the simulation. Here, it can be seen that the first blocks are requested faster in the simulation since we ignore the waiting time between requests in the initial phase. Therefore, data is downloaded faster in the initial phase and the buffered playtime increases faster as compared to the real measurements. In addition, a new block is requested in our model when the buffered playtime drops to 50 s. In our measurement results, these values vary slightly, cf. Fig. 5(c). Thus, the point in time when a block is requested deviates by up to two seconds compared to the measured values. These variations do not add up but instead, they are memoryless. Furthermore, ignoring the startup delay (i.e., till $B(t) > \Theta_0$) causes the playback to be started up to one second before it starts according to our measurement. Fig. 5(d) shows a CDF of the difference in block request time between the simulation and the measurement results, considering different video resolutions. For 360p, the mean is at around −0.40 s with a variance of 0.55 and a mean squared error of 0.71. Similar values are observed for other resolutions. The main cause for the negative offset is the zero playback delay considered in the model parametrization. Furthermore, the mean buffered playtime at which blocks are requested is 49.8 s for the samples, in contrast to 50 s that are used in the model.

7.2. Using the player model To extract stallings

The YouTube player conditions the experience of the user watching a video, specially because of its influence on the stalling pattern of a video (i.e., the number and duration of stalling events), which ultimately determines the YouTube QoE. We therefore propose to use the described YouTube player model to extract the stallings of a video. In [44] we have introduced a very simple technique that permits to reconstruct the stalling patterns of a video from the aforementioned player model, following a similar approach to the one considered to the YouTube flow control model as presented in Section 6.2.

The main difference we consider now wrt the flow control model is the temporal-granularity: instead of updating the different parameters for every new requested block at time t_i, the technique works at the packet time level, updating the state with every new video packet TCP ACK received at time τ_i. This provides a much finer granularity to estimate not only the complete stalling



(a) Buffered playtime $B(t)$. (b) $B(t)$ – initial phase. (c) $B(t)$ – detailed view. (d) CDF of errors.

Fig. 5. Comparison of the buffered playtime for different measurements of the same video and the estimations provided by the video player model by simple simulation. The estimation is almost perfect, but there are some small misalignments in the specific times when additional video blocks are requested, both at the initial pre-buffering phase as well as doing the video playback. Still, differences are almost negligible, being always below 2 s.

time of a video, but also the individual length and time of single stalling events during the video playback. To work at such low temporal granularity, we resort to the analysis of the metadata contained in the flow of YouTube packets.

The playback times of the video frames composing the video can be obtained by dissecting the metadata present in the so-called *video container* (e.g, MP4, VP9 etc.). Each YouTube video is compressed and encoded as a MP4, VP9, etc. file which is a container format for media files. The container includes the compressed video and audio, as well as the information needed by the YouTube player to decode and display the video content. The header of these media files starts with a well-defined signature identifying the corresponding container format, and contains metadata information such as the times when the video frames have to be actually displayed. The developed technique consists of identifying the beginning of a new YouTube video flow as marked by the signature of its container, and extracting the corresponding play times of the downloaded content to estimate the accumulated video play time at the buffer.

Let us describe the parameters which are used in this technique, recalling their definitions from Table 3. The first and most important parameter is the total downloaded video play time at time τ_i, namely $DT(\tau_i)$, which is updated from every new TCP ACK received at time τ_i. As we said before, the value of $DT(\tau_i)$ is obtained by parsing the video container metadata. We additionally consider the video play time $VT(\tau_i)$ and the stalling time $S(\tau_i)$, which are the user experienced video play time and stalling time after the reception of the ith TCP ACK. The buffered video playtime at time τ_i is indicated as $B(\tau_i)$, and it corresponds to the difference between the downloaded video play time $DT(\tau_i)$ and the actually played time $VT(\tau_i)$, i.e., $B(\tau_i) = DT(\tau_i) - VT(\tau_i)$ (cf. Eq (1)). We also consider the boolean stalling variable ψ, which indicates whether the video is currently playing ($\psi = 0$) or stalling ($\psi = 1$), depending on the relations between the buffered video playtime and the playing/stalling thresholds, Θ_0 and Θ_1 respectively. The measurement studies performed in [44] revealed that these two buffer thresholds can be reasonably taken as $\Theta_0 = 2.2$ s and $\Theta_1 = 0.4$ seconds for YouTube players running on laptops and desktop PCs. While these two thresholds are not strictly constant and might depend on the specific characteristics of a video, results shown next suggest that the estimations are highly accurate with these approximations. Even more, other studies such as [28] have estimated these thresholds in very similar values ($\Theta_0 = 1.9$ s and $\Theta_1 = 0.5$), reinforcing the selection done in this paper. Using these definitions, the stalling pattern of a YouTube video over time can be obtained as follows:

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

$$S(\tau_i) = S(\tau_{i-1}) + \begin{cases} \tau_i - \tau_{i-1}, & \text{if } \psi_i \\ 0, & \text{if } \neg\psi_i \end{cases}$$

$$VT(\tau_i) = VT(\tau_{i-1}) + \begin{cases} 0, & \text{if } \psi_i \\ \tau_i - \tau_{i-1}, & \text{if } \neg\psi_i \end{cases}$$

$$B(\tau_i) = DT(\tau_i) - VT(\tau_i)$$

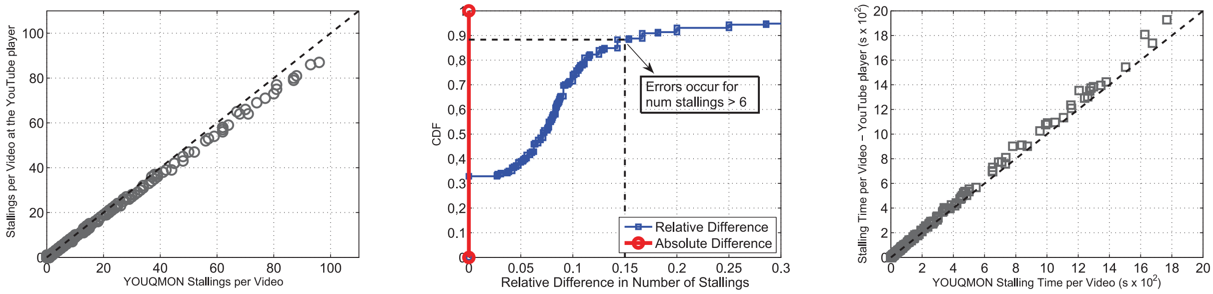
Finally, the time elapsed between the previous ACK at time τ_{i-1} and current ACK at time τ_i increases the stalling time $S(\tau_i)$ or the play time $VT(\tau_i)$, depending on the resulting video state (i.e., stalling or playing). Since YouTube first starts buffering (i.e., stalling state) until the threshold Θ_0 is exceeded, the iterative computation of the different variables is initialized with $S(\tau_0) = VT(\tau_0) = 0$ and $\psi_0 = 1$.

Fig. 6 reports validation results for the proposed technique. Results correspond to 386 YouTube videos streamed from `youtube.com` through a bottleneck link of controlled capacity (from 128kbps to 20Mbps). Fig. 6(a) and 6(c) show that the number and total duration of stallings per video computed by the aforementioned technique are highly consistent with the stallings measured at the YouTube player. Fig. 6(b) shows that for 131 videos, the number of stallings is zero and the absolute difference between the estimated (n_e) and the real (n_a) number of stallings is 0. For the 255 remaining videos, the relative difference $\frac{|n_e - n_a|}{n_a}$ is still 0 for 30% of the cases, and below 15% for about 90% of the videos. Hence, for more than 93% of the 386 tested videos, the estimation is either exact or there are errors for $n_a > 6$. According to the QoE model we described in Section 8 next, MOS differences for $n > 4$ are negligible.

These results show that the YouTube player model and the presented technique can actually be used to extract the stalling patterns that occur during the streaming of a YouTube video, which can then be mapped to QoE values by applying the models presented next. The main limitation of this estimation technique as presented so far is that it has not been conceived as a tool for monitoring the QoE of YouTube from the perspective of an operator, who actually needs to run such estimations in the core or close to it to have an idea of the overall quality his customers are experiencing. The last step to achieve such a monitoring system is described in the last part of the paper.

8. Quality of experience in YouTube: From packets to user perception

The experience of a user with any application is conditioned by multiple influence parameters, including dimensions such as technical characteristics of the application, user personality and expectations, user demographics, device usability, and usage context among others. In the case of YouTube, the most relevant parameters defining its QoE are the stallings of the video playback. Authors in [18] show that, while initial playback delay has also an influence in QoE for video streaming, most users tolerate it because they are used to them. Stalling, on the other hand, has a huge impact as already little stalling severely degrades the QoE.



(a) Estimated number of stallings.

(b) Estimation error in number of stallings.

(c) Estimated stalling time.

Fig. 6. Estimated (a) number of stallings, (b) distribution of errors, and (c) duration of stallings for 386 YouTube videos.

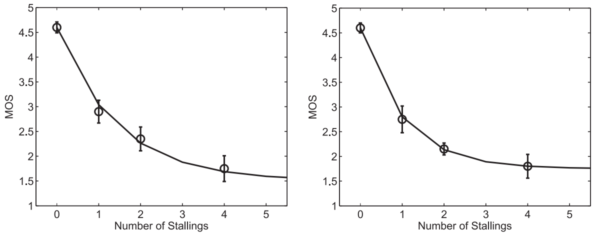


Fig. 7. MOS vs number of stallings from Lab and Crowdsourcing measurements: stallings of 2 (left) and 4 (right) seconds of duration.

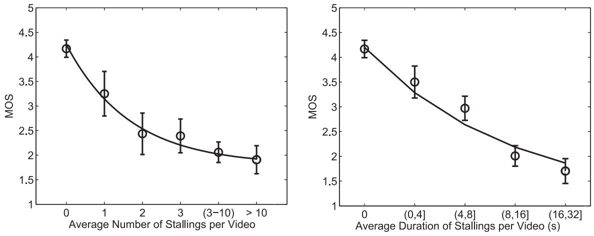


Fig. 8. MOS vs average number (left) and average duration (right) of stallings per video from field measurements.

In this section we study the relations between both the number and the duration of these stalling events and the users' perception. Having a model which can map stallings to QoE has a very powerful advantage, that of becoming independent of the underlying specific characteristics of the network in which the YouTube QoE will be evaluated.

Figs. 7 and 8 depict these relations for both controlled studies (lab and crowdsourcing) and field experiments we have performed in [18,20,21]. In the case of lab and crowdsourcing studies, 37 participants watched different YouTube videos for which a fully controlled stalling pattern was applied (i.e., number and duration of stalling events were perfectly defined), and then rated the perceived overall quality according to an ordinal ACR mean opinion score (MOS) scale [51], ranging from "bad" (MOS=1) to "excellent" (MOS=5). The 37 users were adults aged between 20 and 72 years (18 female, 19 male, average age of 39 years old), and about 65% had a daily Internet usage between 1 and 5 hours. The obtained results are depicted in Fig. 7.

In the case of field studies, a group of 33 participants used mobile broadband 3.5G modems connected to the network of a mobile network operator to watch their preferred YouTube videos on their own laptops, rating the overall perceived quality. In this study, the average age of participants (12 female, 21 male) was 32 years old, and more than 70% had a daily Internet usage between 1 and 5 hour. Stalling patterns can not be controlled in field studies; for this reason, participants' traffic was rate-limited to different down-link bandwidth values, and the resulting stallings were measured at the application layer using the aforementioned YoMo tool. Fig. 8 shows the results.

Both lab and field studies show that user perception of stalling events is highly non-linear, with one single stalling event already significantly impairing the overall experience. In both cases, a single stalling event reduces the video quality from excellent to fair (i.e., 1 MOS point in the scale). Note that the maximum ratings provided by users in both Fig. 7 and Fig. 8 are never 5 but somewhere between 4.3 and 4.6. This is a well known phenomenon in QoE studies, where users hardly employ the limit values of the scale for their ratings [52]. A second stalling event has also a strong influence on YouTube QoE, but saturation already starts after 2 stallings, as even getting more than 4 stallings slightly reduces the QoE from around 2 to 1.6. Stallings duration also plays an important role in YouTube QoE, but shows to be less critical in this case. For example, doubling the stalling duration from 2 to 4 seconds in the lab studies has a limited impact, but increasing its value to more than 8 seconds shows degradation of the user experience in the field.

8.1. Combining models for QoE-based traffic monitoring

In this section we combine the results obtained from previous QoE studies into a single YouTube QoE model. By coupling this model with the YouTube player model and the stalling reconstruction technique so far presented, we have designed an on-line passive monitoring system for assessing, in real time, the QoE undergone by customers watching YouTube videos in the mobile network of a major European ISP. The system is known as YOUQMON [43].

The proposed monitoring system consists of passive data analysis of the traffic observed in the well known Gn data interface of a mobile operator. YouTube flows are identified on the fly using pattern matching and deep packet inspection techniques [43], and stalling patterns are extracted for every observed YouTube video, producing a per-video report in a time-slotted temporal basis (in the practice, every minute). To map the extracted number and duration of stalling events into MOS values, we have adapted the datasets and curves presented before to the specific slotted time functioning of the monitoring system. In particular, we have considered a new mapping function where we take the ratio λ between the total stalling time and the total video elapsed time (i.e., playing + stalling time) in the corresponding time slot as a better image of the impacts of stalling time on YouTube QoE. This permits to limit the effects of videos with different durations, as we are now considering the stalling time relative to the length of the evaluation (i.e., the length of the time slot). The resulting YouTube stallings–QoE mapping model depicted in Fig. 9 is decomposed in five different functions, depending on the value of λ computed in the time slot of length T ($T = 60$ s). The five functions have all the

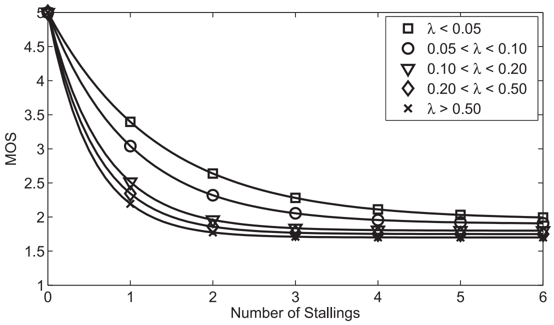


Fig. 9. MOS vs number of stallings, depending on the fraction λ of total stalling duration.

Table 4
MOS vs. # stallings operational model parameters.

i	λ	a_i	b_i	c_i
1	$\lambda < 0.05$	2.97	0.74	2.03
2	$0.05 < \lambda < 0.10$	3.07	0.96	1.93
3	$0.10 < \lambda < 0.20$	3.17	1.55	1.83
4	$0.20 < \lambda < 0.50$	3.21	1.66	1.79
5	$\lambda > 0.50$	3.24	1.79	1.76

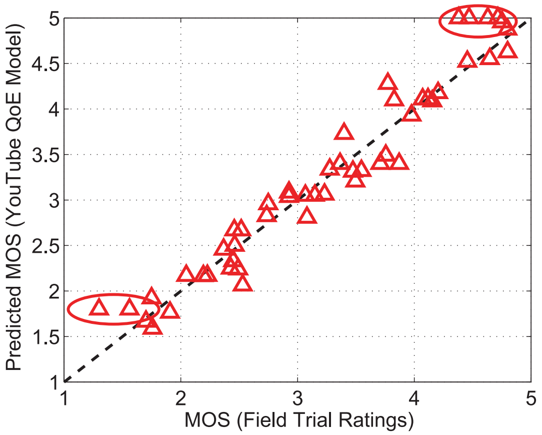


Fig. 10. On-line QoE monitoring results. Validation with real traces from the field trial.

same shape, in the form of

$$\mathrm{MOS}(n)_i = a_i \cdot e^{-b_i \cdot n} + c_i, \quad \forall i = 1, 2, 3, 4, 5. \tag{9}$$

where n is the number of stalling events estimated on the time slot of length T and $\{a_i, b_i, c_i\}$ depend on the computed value for λ, see Table 4. At every new time slot where a YouTube video is detected, the value of λ is obtained as follows: first, we compute the total stalling time σ and the total play time ρ for this time slot; then, if the total video elapsed time $\rho + \sigma$ is smaller than the length of the time slot T, then we compute $\lambda = \sigma/(\sigma + \rho)$; otherwise, $\lambda = \sigma/T$. The curves depicted in Fig. 9 deserve some clarifications: firstly, the MOS value computed for $n = 0$ stallings only makes sense for the curve in which $\lambda < 5\%$; in all the other cases, $n > 0$. Secondly, the curves only show mappings for up to $n = 6$ stallings; this is because a YouTube video with more than such a number of stalling events can be directly declared as very bad quality (cf. Figs. 7 and 8), and no extra mapping is therefore required.

To validate the QoE estimation properties of the proposed system, we replay some of the network packet traces captured in the field trial study conducted in [20], for which we have the MOS values declared by the users as ground truth. Fig. 10 compares both the declared MOS and the predicted MOS values for 50 different videos which experienced different stalling patterns in the field trial. All the considered videos have a total duration of less than 60 seconds, just to avoid any biased comparison due to the different evaluation procedure used in the field trial and on this evaluation. Obtained results are very accurate and close to the MOS values actually declared by the participants, but some strange deviations occur at the edges of the rating scale, both at very low or very high MOS values. This difference comes from the edge-ratings phenomenon previously mentioned. In the field study, ratings for 0 stallings correspond to MOS values around 4.5, while the model depicted in Fig. 9 gives a MOS value of 5 on these situations. Similarly, the limit values for very bad quality provided by the model are slightly higher than the actual opinion of the users; for this reason, the model provides a MOS value around 1.8 when users actually rate around 1.5. In any case, the reader should note that none of both identified differences are an issue to consider, as they occur so at the edges of the scale.

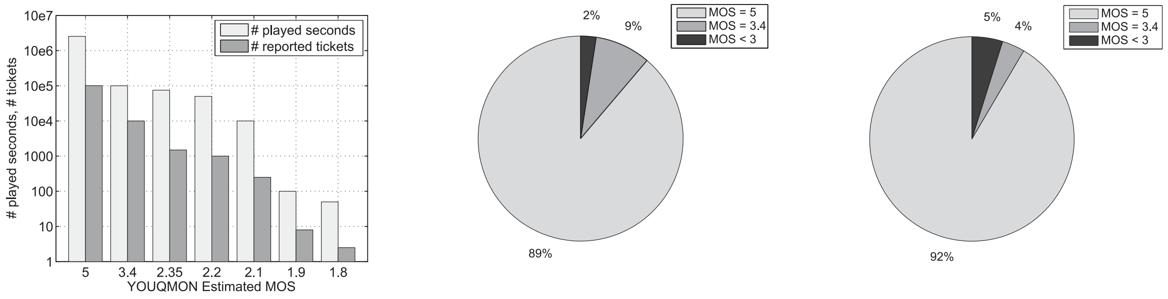
To conclude, we present the YouTube QoE monitoring results obtained by using this system with the real mobile broadband traffic of the aforementioned operator. Fig. 11(a) depicts an histogram on the number of reported *tickets* (a ticket reports the QoE estimation results for every video and every time slot T) and the total played seconds of YouTube videos at the different estimated QoE levels, for one hour of real traffic monitored at the live network. As reported by the pie charts in Figs. 11(b) and 11(c), the resulting YouTube QoE in this network is excellent (i.e., MOS = 5) for about 90% of the issued tickets and of the video time consumed during the analyzed hour. For 9% of the issued tickets and 4% of the total video time, the quality achieved was average (i.e., MOS = 3.4 in this case). Regarding bad quality events, one of the main limitations of doing only monitoring is that the system can not say whether bad quality events come from problems on the network or in any other part of the end-to-end path (the customer terminal, the YouTube servers, a bad SNR, etc). Still, the level of visibility the operator gets by following such an approach is solely by itself a great asset.

8.2. YouTube QoE in DASH

So far we have focused the QoE analysis on the fixed-quality video streaming approach normally followed by YouTube. Still, the massive application of YouTube DASH, as well as its growing usage in mobile networks and end devices, introduces some interesting aspects from the QoE perspective that we discuss next. Indeed, whilst adaptive streaming concepts are known for a long time, their broad commercial usage has only risen recently, and the topic is getting more and more attention within the research community. In the case of adaptive streaming, a new KPI becomes relevant in terms of QoE: quality switches. It is well-known that dynamic quality adaptation can dramatically reduce stalling when bandwidth decreases in a mobile environment, but at the same time, quality switches might have an important impact on QoE, as they increase or decrease the video quality during the playback.

Fig. 12 reports the overall quality results obtained for YouTube in mobile devices (i.e., smartphones) in subjective lab tests we have recently conducted in [53], where we have compared two different flavors of the YouTube smartphone application. Whereas in one case we fix the quality of the watched videos to constant HD quality, in the other one we configure the application to use adaptive streaming (i.e., DASH). In the DASH case, videos are also requested in HD quality, but the service itself adapts the subsequent video quality resolutions to throughput variations.

Figs. 12 (a) and 12(b) compare the QoE experienced by the participants using the fixed HD quality configuration against the DASH configuration, assuming always a constant downlink bandwidth value during the video display. It is quite impressive to

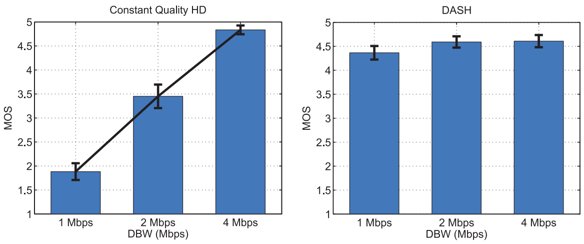


(a) # of QoE tickets and played minutes vs. MOS scores.

(b) Distribution of # of QoE tickets in terms of MOS scores.

(c) Distribution of # of played minutes in terms of MOS scores.

Fig. 11. YouTube QoE-based monitoring in a real network. The monitoring is performed at the Gn interface of the mobile network of a leading European network operator, on a period of one hour.



(a) YouTube QoE – HD fixed.

(b) YouTube QoE – DASH.

Fig. 12. Overall QoE for YouTube in smartphones, considering both DASH and non-DASH applications. Videos are UHD 4k, but due to the device capabilities, there are re-scaled to 720p.

appreciate how the DASH approach results in a nearly optimal QoE for all the tested conditions (from 1 Mbps to 4 Mbps), whereas the fixed HD quality approach results in poor QoE for downlink bandwidth values below 4 Mbps. The main difference here is that DASH changes the video quality without incurring in playback stallings, whereas the fixed quality configuration definitely results in video stallings.

The main takeaway of these simple evaluations is that, whatever new YouTube QoE-based monitoring systems relying on network throughput measurements, it must definitely address the definition of new metrics considering the characteristics of YouTube DASH. Indeed, results from the end-user perspective in the case of DASH are completely different from the traditional bandwidth-QoE relations observed in the past. We have recently presented some first promising results in terms of modeling QoE for DASH [54], but there is still a long way to go when it comes to live monitoring systems as the one we described before.

9. Conclusion

This paper characterized and modeled YouTube, the most popular and volume-dominant service in today's Internet. Going from the generated network traffic to the Quality of Experience perceived by the users watching YouTube videos, we have investigated and derived different models to better understand the functioning of YouTube. In particular, we introduced a network traffic model for the new YouTube flow control mechanism, which permits to understand how YouTube provisions the video traffic flows to the users. We have also investigated how the traffic is consumed at the client side, and derived a simple model for the YouTube application. Finally, we analyzed the operation of YouTube from an end-user perspective, presenting a model for the quality perceived by them. All in all this paper provides objective tools and models to network operators to better understand the YouTube traffic in their networks, to predict the playback behavior of the video player, and to assess how well they do with the YouTube traffic in terms of the satisfaction of their customers.

Acknowledgments

This work has been partially supported by the German Research Foundation (DFG) in the context of the project QoE-DZ (TR 257/41). The work has been partially performed within the framework of the projects ACE 3.0 and N-0 at the Telecommunications Research Center Vienna (FTW), and has been partially funded by the Austrian Government and the City of Vienna through the program COMET. Further work was carried out in the course of the H2020 project INPUT (Call H2020-ICT-2014-1, Grant no. 644672).

References

[1] V.K. Adhikari, S. Jain, Z.-L. Zhang, YouTube traffic dynamics and its interplay with a tier-1 isp: an isp perspective, in: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, ACM, 2010, pp. 431–443.

[2] L. Plissonneau, E. Biersack, P. Juluri, Analyzing the impact of YouTube delivery policies on user experience, in: Proceedings of the 24th International Teletraffic Congress, International Teletraffic Congress, 2012, p. 28.

[3] V.K. Adhikari, S. Jain, Y. Chen, Z.-L. Zhang, Vivisecting YouTube: an active measurement study, in: INFOCOM, 2012 Proceedings IEEE, IEEE, 2012, pp. 2521–2525.

[4] V.K. Adhikari, S. Jain, Z.-L. Zhang, Where do you YouTube? uncovering YouTube server selection strategy, in: Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on, IEEE, 2011a, pp. 1–6.

[5] V.K. Adhikari, S. Jain, Y. Chen, Z.-L. Zhang, Reverse engineering the YouTube video delivery cloud, in: Proceedings of IEEE Hot Topics in Media Delivery Workshop, 2011b.

[6] R. Torres, A. Finamore, J.R. Kim, M. Mellia, M.M. Munafo, S. Rao, Dissecting video server selection strategies in the YouTube cdn, in: Distributed Computing Systems (ICDCS), 2011 31st International Conference on, IEEE, 2011, pp. 248–257.

[7] F. Wamser, D. Hock, M. Seufert, T. Zinner, P. Tran-Gia, Demonstrating the benefit of joint application and network control within a wireless access network, IEEE INFOCOM 2013, Demo Session (2013).

[8] F. Wamser, D. Staehle, J. Prokopec, A. Maeder, P. Tran-Gia, Utilizing buffered youtube playtime for QoE-oriented scheduling in OFDMA networks, in: International Teletraffic Congress (ITC), Krakow, Poland, 2012.

[9] Sandvine - Intelligent Broadband Networks, Global Internet Phenomena Report, 2014.

[10] YouTube Press Room - Statistics, 2015, October 2015. [Online]. Available: https://www.youtube.com/yt/press/statistics.html.

[11] F. Wamser, R. Pries, D. Staehle, K. Heck, P. Tran-Gia, Traffic characterization of a residential wireless internet access, Telecommun. Syst. 48 (1-2) (2011) 5–17.

[12] F. Wamser, D. Hock, M. Seufert, B. Staehle, R. Pries, P. Tran-Gia, Using buffered playtime for QoE-oriented resource management of youtube video streaming, Trans. Emerg. Telecommun. Technol. 24 (2013).

[13] A. Finamore, M. Mellia, M.M. Munafò, R. Torres, S.G. Rao, YouTube everywhere: impact of device and infrastructure synergies on user experience, in: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, ACM, 2011, pp. 345–360.

[14] S. Alcock, R. Nelson, Application flow control in YouTube video streams, ACM SIGCOMM Comput. Commun. Rev. 41 (2) (2011) 24–30.

[15] P. Casas, P. Fiadino, A. Sackl, A. D'Alconzo, YouTube in the move: understanding the performance of YouTube in cellular networks, in: Wireless Days (WD), 2014 IFIP, 2014a, pp. 1–6, doi:10.1109/WD.2014.7020798.

[16] P. Casas, A. D'Alconzo, P. Fiadino, A. Bar, A. Finamore, T. Zseby, When YouTube does not work: analysis of QoE-relevant degradation in google cdn traffic, Netw. Serv. Manage IEEE Trans. on 11 (4) (2014b) 441–457, doi:10.1109/TNSM.2014.2377691.

[17] P. Ameigeiras, A. Azcona-Rivas, J. Navarro-Ortiz, J.J. Ramos-Munoz, J.M. López–Soler, A simple model for predicting the number and duration of rebuffering events for YouTube flows, Commun. Lett. IEEE 16 (2) (2012) 278–280.

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

[19] R.K.P. Mok, E.W.W. Chan, R.K.C. Chang, Measuring the quality of experience of http video streaming., in: N. Agoulmine, C. Bartolini, T. Pfeifer, D. O'Sullivan (Eds.), Integrated Network Management, IEEE, 2011, pp. 485–492.

[20] P. Casas, A. Sackl, S. Egger, R. Schatz, YouTube & facebook quality of experience in mobile broadband networks, in: Globecom Workshops (GC Wkshps), 2012 IEEE, IEEE, 2012, pp. 1269–1274.

[21] T. Hoßfeld, R. Schatz, M. Seufert, M. Hirth, T. Zinner, 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, 2011.

[22] L. Chen, Y. Zhou, D.M. Chiu, Video browsing:a study of user behavior in online VoD services, in: Computer Communications and Networks (ICCCN), 2013 22nd International Conference on, IEEE, 2013, pp. 1–7.

[23] J. Yao, S.S. Kanhere, I. Hossain, M. Hassan, Empirical evaluation of HTTP adaptive streaming under vehicular mobility, in: NETWORKING 2011, Springer, 2011, pp. 92–105.

[24] ISO/IEC 23009-1:2012 information technology - dynamic adaptive streaming over HTTP (DASH) - part 1: Media presentation description and segment formats, 2012,

[25] S. Khemmarat, R. Zhou, D.K. Krishnappa, L. Gao, M. Zink, Watching user generated videos with prefetching, Signal Process. Image Commun. 27 (4) (2012) 343–359.

[26] X. Cheng, C. Dale, J. Liu, Understanding the characteristics of internet short video sharing: YouTube as a case study, arXiv preprint arXiv:0707.3670 (2007).

[27] A. Rao, A. Legout, Y. Lim, D. Towsley, C. Barakat, W. Dabbous, Network characteristics of video streaming traffic, in: Conference on Emerging Networking Experiments and Technologies, ACM, 2011, p. 25, doi:10.1145/2079296.2079321.

[28] P. Ameigeiras, J.J. Ramos-Munoz, J. Navarro-Ortiz, J.M. Lopez-Soler, Analysis and modelling of YouTube traffic, Eur. Trans. Telecommun. 23 (4) (2012) 360–377.

[29] S. Alcock, R. Nelson, Application flow control in YouTube video streams, ACM SIGCOMM Comput. Commun. Rev. 41 (2) (2011) 24–30.

[30] J.J. Ramos-Munoz, J. Prados-Garzon, P. Ameigeiras, J. Navarro-Ortiz, J.M. Lopez–Soler, Characteristics of mobile YouTube traffic, Wireless Commun. IEEE 21 (1) (2014) 18–25.

[31] B. Staehle, F. Wamser, M. Hirth, D. Stezenbach, D. Staehle, AquareYoum: application- and quality of experience-aware resource management for youtube in Wwireless mesh networks, Praxis der Informationsverarbeitung und Kommunikation 34 (3) (2011a) 144–148.

[32] B. Staehle, M. Hirth, R. Pries, F. Wamser, D. Staehle, Aquarema in action: improving the youtube QoE in wireless mesh networks, in: Baltic Congress on Future Internet Communications (BCFIC), Riga, Latvia, 2011b.

[33] T. Hoßfeld, F. Liers, R. Schatz, B. Staehle, D. Staehle, T. Volkert, F. Wamser, Quality of experience management for youtube: clouds, FoG and the AquareYoum, Praxis der Informationverarbeitung und -kommunikation (2012).

[34] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, P. Tran-Gia, Sdn-based application-aware networking on the example of YouTube video streaming, in: Software Defined Networks (EWSDN), 2013 Second European Workshop on, IEEE, 2013, pp. 87–92.

[35] B. Staehle, M. Hirth, R. Pries, F. Wamser, 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, 2010a.

[36] B. Staehle, M. Hirth, F. Wamser, R. Pries, D. Staehle, YoMo: A YouTube Application Comfort Monitoring Tool, Technical Report 467, University of Würzburg, 2010b.

[37] R. Schatz, T. Hoßfeld, P. Casas, Passive YouTube qoe monitoring for isps, in: Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, IEEE, 2012, pp. 358–364.

[38] P. Casas, R. Schatz, T. Hoßfeld, Monitoring YouTube QoE: is your mobile network delivering the right experience to your customers? in: Wireless Communications and Networking Conference (WCNC), 2013 IEEE, IEEE, 2013, pp. 1609–1614.

[39] P. Gill, M. Arlitt, Z. Li, A. Mahanti, YouTube traffic characterization: a view from the edge, in: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, ACM, 2007, pp. 15–28.

[40] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, S. Moon, I tube, YouTube, everybody tubes: analyzing the world's largest user generated content video system, in: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, ACM, 2007, pp. 1–14.

[41] J. Erman, A. Gerber, K. Ramadrishnan, S. Sen, O. Spatscheck, Over the top video: the gorilla in cellular networks, in: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, ACM, 2011, pp. 127–136.

[42] T. Hoßfeld, R. Schatz, E. Biersack, L. Plissonneau, Internet video delivery in youtube: from traffic measurements to quality of experience, in: M.M. Ernst Biersack (Ed.), Data Traffic Monitoring and Analysis: From Measurement, Classification and Anomaly Detection to Quality of Experience, Computer Communications and Networks series, Springers, Berlin, 2012.

[43] P. Casas, M. Seufert, R. Schatz, YOUQMON: a system for on-line monitoring of youtube QoE in operational 3G networks, SIGMETRICS Perform. Eval. Rev. 41 (2) (2013) 44–46, doi:10.1145/2518025.2518033.

[44] R. Schatz, T. Hoßfeld, P. Casas, Passive youtube QoE monitoring for ISPs, in: Workshop on Future Internet and Next Generation Networks (FINGNet-2012), Palermo, Italy, 2012.

[45] T. Hoßfeld, R. Schatz, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, Quantification of youtube QoE via Crowdsourcing, in: IEEE International Workshop on Multimedia Quality of Experience, Dana Point, CA, USA, 2011.

[46] B. Staehle, M. Hirth, R. Pries, F. Wamser, D. Staehle, Aquarema in action: improving the youtube QoE in wireless mesh networks, in: Baltic Congress on Future Internet Communications (BCFIC), Riga, Latvia, 2011.

[47] T. Hoßfeld, R. Schatz, M. Seufert, M. Hirth, T. Zinner, 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, 2011.

[48] C. Timmerer, C. Müller, Http streaming of mpeg media, Streaming Day (2010).

[49] D.K. Krishnappa, D. Bhat, M. Zink, Dashing YouTube: an analysis of using dash in YouTube video service, in: Local Computer Networks (LCN), 2013 IEEE 38th Conference on, IEEE, 2013, pp. 407–415.

[50] C. Sieber, T. Hoßfeld, T. Zinner, P. Tran-Gia, C. Timmerer, Implementation and user-centric comparison of a novel adaptation logic for dash with svc, in: Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on, IEEE, 2013, pp. 1318–1323.

[51] International Telecommunication Union, Methods for subjective determination of transmission quality, ITU-T Recommendation P.800 (1996).

[52] P. Casas, R. Schatz, Quality of experience in cloud services: survey and measurements, Comput. Netw. 68 (2014) 149–165. http://dx.doi.org/10.1016/j.comnet.2014.01.008. Communications and Networking in the Cloud.

[53] M. Seufert, F. Wamser, P. Casas, R. Irmer, T.-G. Phuoc, R. Schatz, YouTube QoE on mobile devices: subjective analysis of classical vs. adaptive video streaming, in: Proceedings of the 6th International Workshop on Traffic Analysis and Characterization (TRAC), 2015.

[54] T. Hoßfeld, M. Seufert, C. Sieber, T. Zinner, Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming, in: 6th International Workshop on Quality of Multimedia Experience (QoMEX), Singapore, 2014.

Florian Wamser is a Research Associate at the Chair of Communication Networks at the University of Wrzburg, Germany. He leads the group on cloud networks and Internet Applications at the Chair of Prof. Dr.-Ing. Phuoc Tran-Gia. Florian studied at the University of Wrzburg and at the Helsinki University of Technology, Finland. He received his diploma in computer science in 2009. During the thesis he worked on the topics characterization and modeling of application Internet traffic in broadband wireless access networks. In 2015 he received his PhD. The title of his dissertation is Performance Assessment of Resource Management Strategies for Cellular and Wireless Mesh Networks. His current research is focused on the analytical and simulative performance evaluation and optimization of cloud networks and related fields.

Pedro Casas is Scientist at the Austrian Institute of Technology (AIT) in Vienna. He received an Electrical Engineering degree from the University of the Republic, Uruguay in 2005, and a Ph.D. degree in Computer Science from Tlcom Bretagne, France in 2010. He held a Research and Teaching Assistant position at the University of the Republic between 2003 and 2012, and was at the french research lab LAAS-CNRS in Toulouse as a Postdoctoral Research Fellow between 2010 and 2011. Between 2011 and 2015 he was Senior Researcher at the Telecommunications Research Center Vienna (FTW). His main research areas span the monitoring and analysis of network traffic, network security and anomaly detection, QoE modeling and automatic assessment, as well as machine-learning and data mining based approaches for Networking. Dr. Casas has published more than 80 Networking research papers (50 as main author) in major international conferences and journals, and has received seven best paper awards in the last 6 years.

Michael Seufert studied computer science, mathematics, and education at the University of Würzburg. In 2011, he received his diploma degree in Computer Science, and additionally passed the state examinations for teaching Mathematics and Computer Science in secondary schools. From 2012 to 2013, he was with FTW Telecommunication Research Center, Vienna, Austria, working in the area of user-centered interaction and communication economics. He is currently a researcher at the Chair of Communication Networks, University of Würzburg, working toward the Ph.D. degree. His research mainly focuses on QoE of Internet applications, social networks, performance modeling and analysis, and traffic management solutions.