Internet Video Delivery in YouTube: From Traffic Measurements to Quality of Experience

Tobias Hoßfeld[1], Raimund Schatz[2], Ernst Biersack[3], and Louis Plissonneau[4]

[1] University of Würzburg, Institute of Computer Science, Germany
tobias.hossfeld@uni-wuerzburg.de
[2] Telecommunications Research Center Vienna (FTW), Vienna, Austria
schatz@ftw.at
[3] Eurecom, Sophia Antipolis, France
erbi@eurecom.fr
[4] Orange Labs, France
louis.plissonneau@orange.com

Abstract. This chapter investigates HTTP video streaming over the Internet for the YouTube platform. YouTube is used as concrete example and case study for video delivery over the Internet, since it is not only the most popular online video platform, but also generates a large share of traffic on today's Internet. We will describe the YouTube infrastructure as well as the underlying mechanisms for optimizing content delivery. Such mechanisms include server selection via DNS as well as application-layer traffic management. Furthermore, the impact of delivery via the Internet on the user experienced quality (QoE) of YouTube video streaming is quantified. In this context, different QoE monitoring approaches are qualitatively compared and evaluated in terms of the accuracy of QoE estimation.

1 Introduction

Quality of Experience (QoE) describes the user perception and satisfaction with application and service performance in communication networks, a topic that has gained increasing attention during the last years. Part of this growth of interest in QoE can be explained by increased competition amongst providers and operators, and by the risk that users churn as they become dissatisfied. However, many users face volatile network conditions, e.g. due to temporary over-utilization of shared network resources such as peering links. Such conditions may result in bad QoE. The focus of this book chapter is on Internet video delivery, since video streaming dominates global Internet traffic and exceeded half of global consumer Internet traffic at the end of 2011 [1]. We differentiate two different types of video content delivery over the Internet, (i) live video streaming with on-the-fly encoding, like IPTV, and (ii) streaming of pre-encoded video, so called Video-on-Demand (VoD). In this chapter, we focus on YouTube, the most popular VoD service in the Internet with more than two billion video streams daily.

The recent surge in popularity of Internet video requires a considerable investment by the operators of these services in order to be able to satisfy the demand. The **delivery infrastructure** (Section 2) is generally distributed all over the world and comprises of tens of different sites. A user must be automatically directed to a nearby site (which is the role of DNS) and must be redirected in case the video is not available on this particular site or if the servers of this site are overloaded. Thus, a cache selection mechanism is implemented within the delivery infrastructure. Today, the delivery of the video data is typically performed via TCP, which is a reliable transport protocol that performs error recovery and congestion control. When using TCP, the transmission can be subject to considerable delay jitter and throughput variations and the client needs to preload a play out buffer before starting the video playback. Various transmission strategies from the server to the client are possible such as client pull or server push mode. Also the size of the video blocks transmitted can vary from 64 Kbytes to a few Mbytes. This is referred to as application-layer traffic management.

For an Internet Service Provider (ISP) providing connectivity to the end user, it is thus important to understand the relationship between Quality of Experience (QoE) of a service and the performance characteristics of the service provisioning through networks, resulting into a so-called **QoE model** (Section 3). In the context of network provisioning, QoE also opens the possibility to save resources by proper QoE management, as it is not economic to invest in better Quality of Service (QoS) for maintaining the same level of QoE. For example, reserving a bandwidth of 16 Mbps for delivering a video stream that has a video bit rate of only 300 Kbps unnecessarily consumes ISP's resources without improving QoE.

To implement a QoE management, the ISP must identify and monitor the traffic in its network that results from that service. These measurement data is used for estimating the QoE by means of an appropriate QoE model. Since YouTube video streams do not change encoding parameters during playback and packet loss artifacts do not occur (due to the use of TCP), the QoE of YouTube is primarily determined by stalling effects on application layer as opposed to image degradation in UDP-based video streaming.

In this book chapter, we investigate **QoE monitoring for YouTube video streaming**. To this end, Section 2 provides the necessary technical background and fundamentals in terms of the delivery infrastructure, cache selection, and traffic management mechanisms that enable such a service. Section 3 then investigates the end-user perspective by identifying the key influence factors for YouTube QoE and developing a model the maps application-layer QoS to QoE. Finally, Section 4 brings these elements together to present various YouTube QoE monitoring approaches both, at the application-layer and the network-layer. The approaches at these two layers are fundamentally different: Application-layer monitoring requires to change the end user application or to install additional monitoring software, but will lead to exact results since performance is directly measured at the user terminal where QoE impairments become directly perceivable. On the other hand, a highly scalable, valid, and robust QoE monitoring approach (including a stalling detector as basic QoE indicator) from measurements within the network is needed by ISPs to detect any problems in the network or to manage the traffic accordingly to overcome networking problems. To this end, several network-layer monitoring approaches of YouTube QoE, as well as an evaluation on its accuracy (compared to application-level monitoring) and implementation prospects are highlighted.

2 Delivery Infrastructure, Cache Selection and Application-Layer Traffic Management

Recent studies show that videos streaming sites represent about half of the Internet data volume, both for mobile access [2] and for fixed access [3]. Moreover, video streaming today produces a traffic volume that is more than double the one due to peer-to-peer. In the case of peer-to-peer the server resources to satisfy the demand come from the peer themselves, while in the case of video streaming a content distribution network must be built that consists of geo-distributed servers and caches. It is therefore instructive to study in detail the organization of a video streaming service and its evolution over time. In the following, we will present the design principles of the YouTube delivery architecture and its performance.

2.1 Basic Facts about YouTube

YouTube, which was created in 2005, allows users to upload and share video content. Its success was immediate, resulting in spectacular growth ever since. For instance, the *number of videos viewed per day* has increased from around 200 Million in 2007 to more than 4 Billion in 2012 [4]. Since YouTube was acquired in late 2007 by Google, its infrastructure has been in constant evolution and the delivery architecture that initially used third party content distribution network services is now fully operated and managed by Google. Not much about YouTube has been disclosed by Google itself [4–6]. However, in the last couple of years YouTube has been extensively investigated [7–12] by academia via active and/or passive measurements, which are often carried out from multiple vantage points. While such measurements can reveal certain aspects of YouTube, many details are still unknown. Our description of YouTube is based on the results published in literature and on recent studies of YouTube carried out by ourselves. Describing a system like YouTube that constantly evolves is challenging. However, we believe that the underlying *design principles* will most likely stay the same for some time. We sometimes give real figure to indicate the size of YouTube with the aim to give an idea of the order of magnitude and to provide a reference point for future comparison.

The number of videos hosted by YouTube was estimated in mid-2011 [13] to be about 500 million, which represents about 5 PetaBytes of data considering an average size of 10 MBytes per video. Taking into account replication of videos and multiple formats, this makes a total of about 50 PetaBytes of data. In 2012, an average of *one hour of video was uploaded every second*[0], which is a three-fold increase as compared to 2009/ The number of videos downloaded per day has been evaluated in 2011 to be between 1.7 and 4.6 Billion representing a 50% increase over the previous year, which results in tens of PetaBytes of traffic per day. While YouTube originally started its service in the USA, it has become truly international in the meantime with caches in many countries. Today only 25% of the views are generated in the USA, followed by countries such as UK, Japan, Germany or Brazil, which each generate between 3–7% of all the views [5]. Geographic request locality is high, with around two thirds of the views per video coming from a single region, where a region represents a country or another political, geographic or linguistic entity [5]. Request locality also varies between countries and is highest in Japan and Brazil.

In the following, we focus on the process of downloading the videos. We are going to review the major steps in accessing a YouTube video before we describe the server infrastructure and explain cache selection and cache redirections, which are used to balance the load among caches.

When a user uploads a video to YouTube, the video is stored on a server in one of Google **backend data centers**. YouTube supports multiple video formats. Each video may be transcoded in all the different formats, which can happen pro-actively or on the fly. As we will see in the following, a user that requests a YouTube video will never directly interact with the servers in backend data centers. Instead, the videos will be delivered to the users from so called **caches**.

2.2 Steps in YouTube Video Download

Watching a video on YouTube involves a different set of servers. Initially, the embedding Web page is delivered through front end YouTube web servers, whereas the video content is itself delivered by YouTube video cache servers. YouTube currently supports two containers for video streaming, Flash and HTML5 [14]. At the time of writing, the adoption of HTML5 for YouTube playback on PCs is still in an early phase, and almost all browsers use Flash technology as the default to play the YouTube videos [10]. When the container is Flash, a dedicated Shockwave Flash player must be downloaded to control the Flash plugin in the browser.

Simplified Steps in Accessing a YouTube Video. As shown in Figure 1, the process of accessing a YouTube video can be summarized as (numbers correspond to the graph):

(1) The user requests a video on the YouTube webpage: `http://www.youtube.com/watch?v=videoID` and gets to the Web server that delivers the YouTube HTML page;
(2) After downloading the embedding HTML web page, the other contents are requested in particular the Shockwave Flash Player (embedded in a HTML object that contains the video parameters);
(3) The actual video content is requested from a cache server (`lscache` server); if this cache is over-loaded, it sends a redirect (HTTP 302) message to the client indicating another cache server;
(4) The client sends a request the other cache server (`tccache` server) for the video, and the FLV file is delivered to the client while being played in the Flash player (Progressive Download).

The details of the redirections depend on the load of the cache servers and are explained in the following.

We now focus on the video content delivery, and more specifically on the architecture and the interaction with the cache server infrastructure.

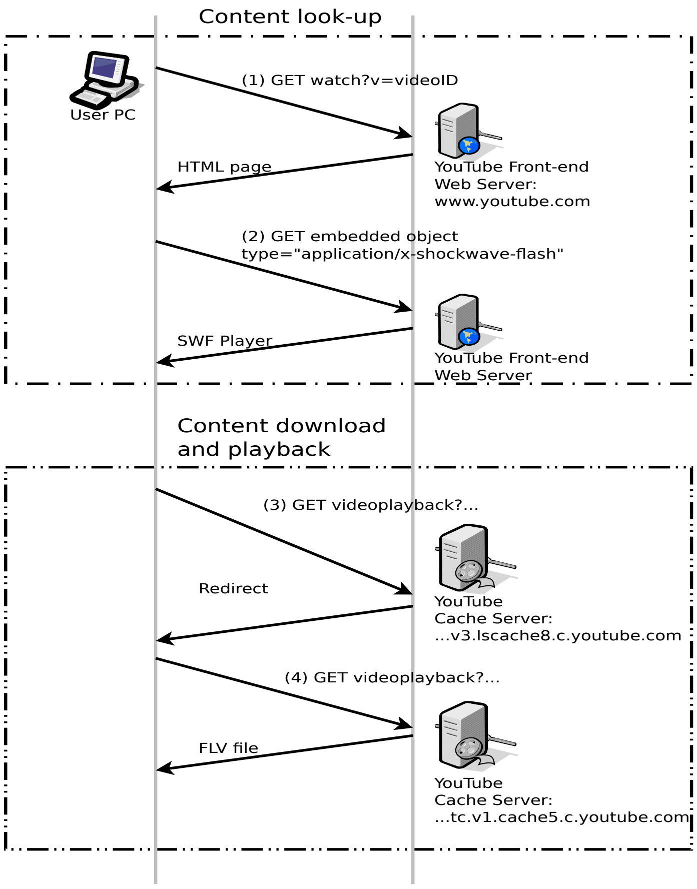

Content look-up
User PC
(1) GET watch?v=videoID
HTML page
YouTube Front-end Web Server: www.youtube.com
(2) GET embedded object type="application/x-shockwave-flash"
SWF Player
YouTube Front-end Web Server
Content download and playback
(3) GET videoplayback?...
Redirect
YouTube Cache Server: ...v3.lscache8.c.youtube.com
(4) GET videoplayback?...
FLV file
YouTube Cache Server: ...tc.v1.cache5.c.youtube.com

Fig. 1. Schema of YouTube page download

YouTube Cache Server Infrastructure. The users receive the video they request from a cache node. Individual **cache nodes** are organized in cache clusters, with all the machines of a **cache cluster** being co-located. The number of machines per cache cluster is highly variable and depends, among others, on the demand for service issued in the region where the cluster is located and also on the available physical space to host the cache nodes. Each cache node as of 2011 has a 10 Gb/sec network access and 78 TByte of disk storage [4].

The various cache clusters are organized in a three tier hierarchy. The global infrastructure of the YouTube caches has been revealed by Adhikari *et al.* [7] in 2011. They used the distributed infrastructure of the PlanetLab network to request thousands of videos from different vantage points in the world, which allowed to reverse engineer the cache infrastructure and the cache selection policies. We complement their findings with our own active measurements [15] undertaken in 2011 and 2012 from France. Our analysis focuses on residential Internet access and reveals the techniques applied by

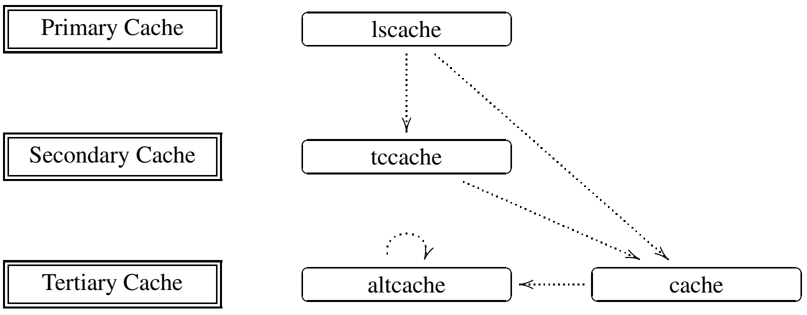


Fig. 2. Organization of the YouTube Caches; dashed lines indicate possible redirections

Google to deliver the videos to residential customers. Since our machines where connected to the Internet through *different ISPs*, we were able to observe differences in treatment of customers coming from different ISPs.

YouTube has a three tier caching infrastructure that comprises of four different *logical* namespaces as shown in Figure 2. The machines allocated to the different cache clusters are identified via particular naming conventions. We recall here the main findings of [7] on the YouTube cache clusters. As of 2011 there are:

- 38 primary cache clusters with about 5000 unique IP addresses corresponding to the `lscache` namespace;
- 8 secondary cache clusters corresponding to the `tccache` namespaces with about 650 IP addresses;
- 5 tertiary caches clusters corresponding to the `cache` and `altcache` namespaces with about 300 IP addresses.

All these cache clusters are located in a total of 47 different locations distributed over four continents; there is no cache cluster in Africa. About ten primary cache clusters are co-located inside ISPs and the IP addresses of these cache nodes in these clusters are not part of the address space managed by Google but part of the ISPs address space. Since we have $38 + 8 + 5 = 51$ cache clusters but only 47 different locations, some cache clusters belonging to different levels of the hierarchy must be at the same physical location (*i.e.* some primary and secondary caches are co-located).

For the caches in each cache cluster, a particular *logical* naming structure is applied.

- Each primary cache cluster has a total of 192 logical caches corresponding to the `lscache` namespace, which looks as follows: `city_code.v[1-24].lscache[1-8].c.youtube.com`. As `city_code` the three letter code for the airport closest to that cache cluster is used.
- There are also 192 logical caches in each secondary cache cluster, corresponding to the `tccache` namespace, which are named as follows `tc.v[1-24].cache[1-8].c.youtube.com`.
- Each tertiary cache cluster has 64 logical caches corresponding to `cache` and `altcache` namespaces.

Introducing these logical name spaces has the following advantages:

- Each video ID can be deterministically mapped via consistent hashing onto a unique logical name in the `lscache` namespace, which makes it easy to decide for each cache what portion of the videos it is responsible to serve.
- There is a one-to-one mapping between the `lscache` and `tccache` namespace.
- The logical naming is the same for each cache cluster and it is completely independent of the number of *real cache* nodes in a particular cache cluster.
- It is the responsibility of DNS to map logical cache names onto the IP addresses of real cache nodes. In [7], each of the logical names from the `lscache` namespace is mapped to more than 75 different IP addresses distributed over the 38 primary cache clusters.

YouTube Datacenter Sizes. We have carried out active measurements in France [15], using simultaneously nine different Internet accesses (7 ADSL and 2 Fiber) to request videos during sessions that lasted for 2 days each. All these accesses are in the same physical location and the access rates for all the ADSL accesses are the same.

The YouTube videos crawled during these measurements were served by two datacenters: one in Paris (`par`), the other in Amsterdam (`ams`). In Tab. 1 we show the number of IP addresses seen for each datacenter and match each IP address with its corresponding `lscache` namespace.

Tab. 1 gives an idea of the size of a cache cluster and also shows the evolution of these cache clusters over time:

- The Amsterdam cache cluster increased its size by 50% within a few months; for this cache cluster there are more IP addresses than distinct `lscache` names, which means that a single `lscache` name will be mapped onto several IP address.
- The Paris cache cluster re-organized the distribution of its IP addresses into *two distinct* logical `lscache` namespaces. For this cache cluster there are fewer IP addresses than distinct `lscache` names, which means that several `lscache` names will be mapped on the same IP address.

In Figure 2, we also show the dynamics of redirections inside the YouTube cache layers. Each cache layer can redirect to the next cache level and the tertiary cache layer can be accessed through redirection out of any layer (including itself). We will explain this in detail in the next section on cache selection.

2.3 YouTube Cache Selection

YouTube cache selection is quite sophisticated and tries to:

- Satisfy users by selecting a *nearby* cache cluster and
- Perform internal redirection to another cache cluster to perform load balancing among cache clusters.

The choice of a close-by cache cluster (in terms of RTT) is typically done through DNS resolution. DNS is used for coarse grained load balancing, with a TTL of five minutes. Before getting into the process of cache server selection through redirections, we first study the selection of first cache server.

Table 1. YouTube Datacenters sizes according to the Number of IP addresses seen for crawls of all ISPs on each URL Regexp

(a) September 2011

URL Regexp	# IPs
o-o.preferred.par08s01.v[1-24].lscache[1-8].c.youtube.com	160†
o-o.preferred.par08s05.v[1-24].lscache[1-8].c.youtube.com	160†
o-o.preferred.ams03g05.v[1-24].lscache[1-8].c.youtube.com	328
o-o.preferred.*ISP*-par1.v[1-24].lscache[1-8].c.youtube.com	98

† these two sets of 160 IP addresses are identical

(b) December 2011

URL Regexp	# IPs
o-o.preferred.par08s01.v[1-24].lscache[1-8].c.youtube.com	80‡
o-o.preferred.par08s05.v[1-24].lscache[1-8].c.youtube.com	80‡
o-o.preferred.ams03g05.v[1-24].lscache[1-8].c.youtube.com	494
o-o.preferred.*ISP*-par1.v[1-24].lscache[1-8].c.youtube.com	130

‡ these two sets of 80 IP addresses are *distinct*

Choice of First Cache Server. With our active measurement carried out across different ISPs in France [15] we also wanted to investigate if clients from different ISPs get directed to the same cache cluster or not. We only focus on the first cache server returned and do not take into account redirections. All the accesses are located in the same place with the same access rate for ADSL. The city codes are `par` and `ams` for Paris and Amsterdam respectively.

We see from Tab. 2, cache cluster used to serve clients clearly depends on the ISP. Here are the main findings (cf. Tab. 2):

- ISP B has all its `lscache` names pointing to one cache site (`par08s01`) in Paris;
- ISP N has all its `lscache` names pointing to the Paris cache site, but with two different logical name spaces (par08s01 and `par08s05`);
- ISP O has dedicated `lscache` names carrying the IPS name (*`ISP`*`-par1`). Also, these names get resolved to IP addresses that belong to a specific AS (36040), which is different from the Google or YouTube ASes.
- ISPs S and F are directed to both cache clusters in Paris or Amsterdam with different proportions: about $2/3$ to Amsterdam for ISP S and 10% for ISP F.

These results highlight that there is a customization done for each ISP for reasons only known to Google.

The network impact of the cache cluster location on the ping time is found to be very low. For example, the minimum ping time from our lab in France to the Paris cache nodes is of 23.8 ms and of 28 ms to Amsterdam (because of relatively small distance between the two cities). However, the main point is that the cache cluster selected in

Table 2. Number of Videos for each ISP according to Regexp on `lscache` names for a controlled crawl in December 2011

	ISP								
URL Regexp	A¶	B¶	B§	F-1¶	F-2¶	N§	O¶	S-1¶	S-2¶
par08s01.v[1-24].lscache[1-8]	0	2676	2677	0	0	1890	0	1967	1528
par08s05.v[1-24].lscache[1-8]	1636	0	0	952	2425	799	0	0	0
ams03g05.v[1-24].lscache[1-8]	150	0	0	0	206	0	0	3033	2488
ISP-par1.v[1-24].lscache[1-8]	0	0	0	0	0	0	2591	0	0

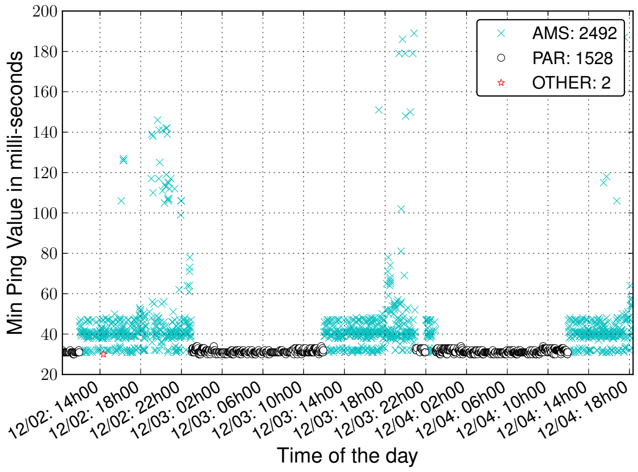
¶ ADSL access
§ Fiber access

not necessarily the geographically closest one and that the choice of the preferred cache cluster depends on the time of day as shown in Figure 3 and on the ISP the client is connected. Moreover, even if the minimum ping values for both cache clusters are about the same, the cross traffic on the path from France to Amsterdam can increase the ping value to values as high as 200 ms. Indeed, Figure 3 shows a large variance in ping times towards Amsterdam cache nodes. The most striking point is that the switch from one datacenter to another is done at a specific time every day, and this time is specific to each ISP.

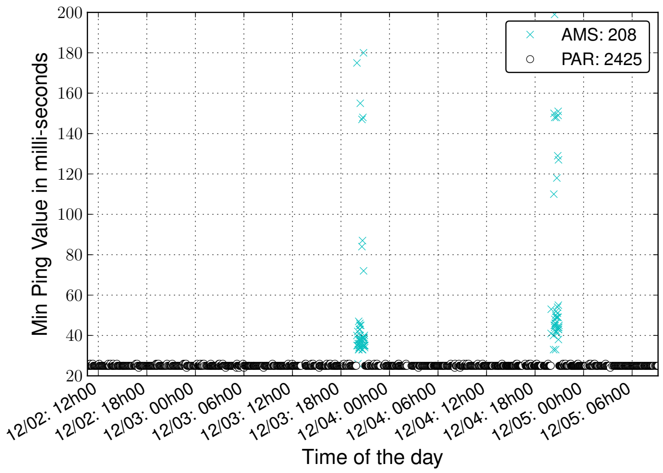
We have made the same observation in [15] for a different experiment with clients located in Kansas, USA, who were served from a variety of different cache clusters located anywhere in the US. In Figure 4, we present a partial map of USA with the location of the most prevalent cache clusters seen in this crawl. The symbols have the following meaning:

– The pin mark is the place from where the crawls are performed: Kansas City;
– Each circle corresponds to a YouTube cache site;
– The diameter of each circle represents the number of videos served by this cache site;
– The color of each circle represents the distance (ping time) towards the cache site: green for ping time lower than 60 ms, blue for ping time between 60 and 200 ms, and red for ping time larger than 200 ms.

Note that we do not show the San Jose and Los Angeles cache sites in the figure, which receive 11% and 3% respectively of the video requests. There are four more cache sites, which are located in Kansas-City, Lawrence, Chicago and New-York that receive a small fraction of all requests. The details can be found in [15]. We clearly see that the distance is not the primary criterion for the choice of the cache site: the most frequently used cache site is in Washington DC, even though it is much further away than the Dallas cache site.



(a) S-2



(b) F-2

Fig. 3. Ping time in milliseconds from our controlled lab towards two cache sites observed in a controlled crawl in December 2011

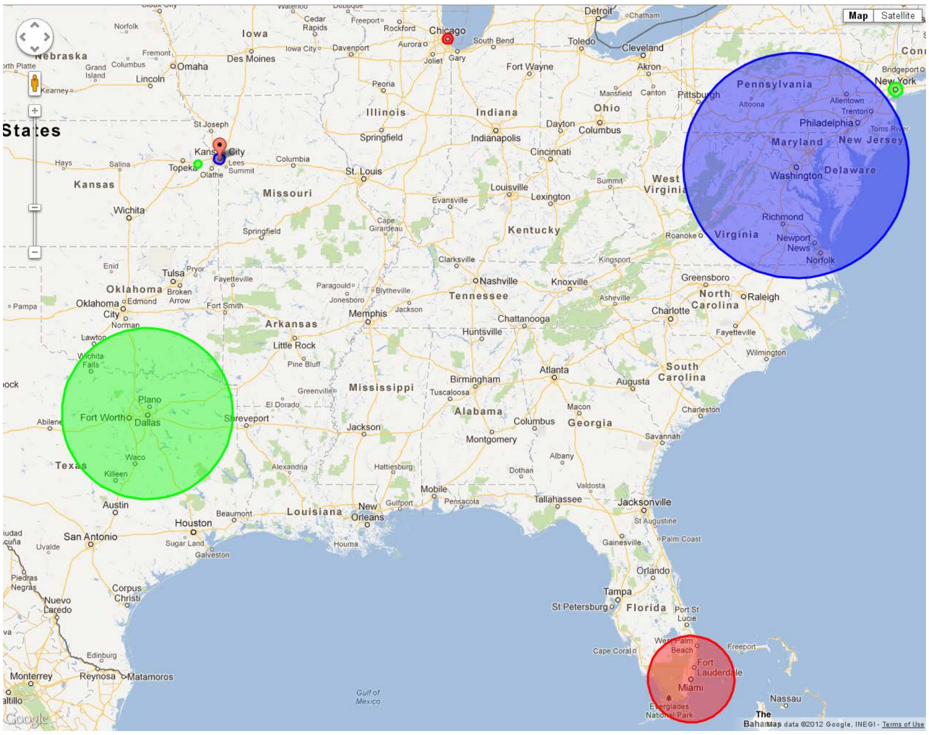


Fig. 4. Map indicates the cache site locations, number of requests served by each cache site (diameter of the circle), and distance (circle color: green for ping $\leq 60\,\mathrm{ms}$, blue for ping $\geq 60\,\mathrm{ms}$ and $\leq 200\,\mathrm{ms}$, and red for ping $\geq 200\,\mathrm{ms}$) of the YouTube cache sites in the crawls that originate from Kansas City (pin mark).

Cache Selection through Redirections. We have already seen that load balancing on YouTube cache servers can be done through DNS resolution: this process is *centralized* at the YouTube authoritative DNS servers. Load-balancing can also be done directly at cache server level. In this case, the cache server receiving the request can relay the request to another cache server via HTTP redirect message at application level. So YouTube can use both *centralized* and *decentralized* processes to balance the requests on its cache servers.

Cache hit. If the video is hot and there are copies at the primary caches, then a logical cache node (`lscache` namespace) in the *primary cache* is chosen. If there is no redirection, a machine from a cache cluster serves the video.
If the primary cache cluster selected is overloaded, a redirection to a *secondary cache cluster* (`tccache` namespace) occurs. The secondary cache can serve the video or redirect it to a *tertiary cache site* (`cache` namespace) for load-balancing purposes.

Again, in the tertiary cache cluster, the cache server can deliver the video, or perform another redirection. A redirection from a tertiary cache site will remain *inside* *the tertiary cache level* and ocur towards a cluster from the `altcache` namespace. A machine in this namespace now serves the video or redirects it inside the same namespace in case of overload. Very rarely, several redirections occur inside the `altcache` namespace, with the total number of redirections being limited to 9. For further details see [7].

Cache Miss. If the video is cold and there are *no* copies at the primary caches, then the request will be most likely redirected from the first level cache to a third level cache. The third level cache will fetch the video from the backend data server, cache it, and deliver the video to the client. It is quite common, as we will see in the next section, that users do not watch the entire video. Therefore, all videos are broken into chunks (of 2 MBytes) and the cache will continue to retrieve from the backend servers new chunks of a video as long as the user keeps viewing that video. Note that despite all the efforts of the engineering team of Google, the cache miss rate remains steadily at about 10% [4].

YouTube Redirection Process

DNS Level Redirections. YouTube uses HTTP redirections to balance the load among its caches. As shown in Figure 2, the redirections usually direct the video request from a cache layer to the next one. Using traces from a European ISP, Torres *et al.* [12] observed that as the total number of requests kept increasing, the percentage of requests handled by the closest cache cluster located inside that ISP decreased from 80% to about 30%. In this case, DNS request resolution will direct clients to more distant but less loaded cache clusters.

Impact of Redirections on Performance. Each redirection involves:

1. DNS query to resolve the hostname of the next cache node,
2. Opening of a new TCP connection,
3. Issuing a new video query.

In case of redirections, the final cache node serving the video will most likely not be the closest one in terms of RTT, which has been observed in [12] for *the most popular videos of the day*.

The impact of redirection on the time until the first MByte is downloaded (referred to as video initialization time) has also been studied in [7]. The video initialization time is on average 33% higher if the video has been fetched through redirections. The fraction of sessions that have been redirected is evaluated in [10]: between 10% and 30% of all sessions are redirected at least once. The impact of redirections on the startup delay can also be important [10]:

- Without redirections, delays are in the order of milliseconds;
- With redirections, delay can increase by orders of magnitude, up to 10 seconds.

2.4 Application-Layer Traffic Management

YouTube videos are requested using HTTP over TCP. TCP is a transport protocol that assures reliable transfer by retransmitting lost data packets and performs congestion control to avoid overloading the network. Both error control and congestion control of TCP may result in high delay jitter.

The delivery strategy of YouTube videos has been studied in great detail by Rao *et al.* [14]. The authors show that the delivery strategy depends on the video container (Flash, Flash High Definition, or HTLM5), the type of client device (PC or mobile devices such as smart phones or IPad), and the type of browser (Internet Explorer, Chrome, or Firefox). The delivery strategy needs to reconcile a number of potentially conflicting goals such as:

- Smooth playout during the entire duration of a viewing session;
- Efficient use of the server resources such as disk I/O and timer management;
- Avoid to transmit too much data in advance of consumption in order to (i) reduce the amount of buffering at the client, which is particularly relevant in the case of mobile devices and to (ii) reduce the waste of network and server resources by sending data that are never used.

Finamore *et al.* [10] observed that 60% of the videos requested were watched for less than 20% of their total duration, resulting in an un-necessary transfer of 25–39% of the data. As we shall see in the following section, the impact of playback degradation is a primary factor in the video transfer interruption.

As the video transfer is done via HTTP over TCP, there is not guarantee that the data can be delivered to the client at the rate at least as high as the one at which they are consumed. The details of the transfer have been studied in [14], whose findings we summarize in the following: To increase the likelihood of a smooth playback, YouTube performs aggressive buffering when a video is requested. Initially, during a **startup** phase, the server sends as fast as possible to fill up the initial client playout buffer. This playout buffer contains about 40 seconds with Flash, and 10–15 MBytes with HTML5 with Internet Explorer as a browser, which is typically much more than 40 seconds worth of video. Once the initial buffer has been filled, two other strategies are used by the cache server:

- keeps sending as fast as possible, until entire video is transmitted;
- limits the rate of the transfer alternating between on-off cycles with a fixed period. During an on cycle, a fixed size block of video data is transmitted.

We limit our description to the case of streaming a video to a PC with Flash as container, and refer to the original paper [14] for more details.

Streaming the video to a PC has been the most extensively studied [6, 16]. In this case, the server streaming strategy is independent of the browser: When the startup phase is terminated, the cache server sends blocks of 64 KBytes at a frequency that allows to achieve an average transmission rate of 1.25 times the video encoding rate. As has been first observed by Alock and Nelson [16], injecting bursts of 64 KBytes means sending 45 maximum size TCP segments back-to-back into the network. Such large packet bursts will accumulate in the buffer of the bottleneck link and (i) cause delay spikes that may disrupt other latency sensitive application and (ii) inflict loss on the bursty YouTube flow itself. In response to these problems, Google engineers have recently proposed a modification to the server side sending policy that controls the amount of packets that can be injected back-to-back in order to limit the size of the packet bursts. For details of the new sender algorithm and its impact on packet loss and burst size see [6].

In this section we have provided the technical basis to understand YouTube content delivery over the Internet. Next, we investigate what influences the QoE experienced by the user. In particular, problems in the network may lead to stalling and QoE degradations. Therefore, we have to identify the key factors that influence YouTube QoE by means of subjective measurements and build an appropriate model, which can be used for QoE monitoring later on.

3 QoE of YouTube Video Streaming

User perceived quality of video streaming applications in the Internet is influenced by a variety of factors. As a common denominator, four different categories of influence factors [17, 18] are distinguished, which are influence factors on context, user, system, and content level.

- The **context level** considers aspects like the environment where the user is consuming the service, the social and cultural background, or the purpose of using the service like time killing or information retrieval.
- The **user level** includes psychological factors like expectations of the user, memory and recency effects, or the usage history of the application.
- The technical influence factors are abstracted on the **system level**. They cover influences of the transmission network, the devices and screens, but also of the implementation of the application itself like video buffering strategies.
- For video delivery, the **content level** addresses the video codec, format, resolution, but also duration, contents of the video, type of video and its motion patterns.

In this section, a simple QoE model for YouTube is presented whose primary focus is its application for QoE monitoring (within the network or at the edge of the network). Therefore, we take a closer look at objectively measurable influence factors, especially on the system and content level. For this purpose, subjective user studies are designed that take into account these influence factors; in particular, we utilize crowdsourcing to have a large pool of human subjects to conduct the tests (Section 3.1). The crowdsourcing environment also allows analyzing influence factors on user level and context level. After analyzing the user ratings that are the key influence factors on YouTube QoE (Section 3.2), simple QoE models and its corresponding mapping functions between those influence factors and the YouTube QoE can be derived (Section 3.3). For illustration, Figure 5 sketches the methodology from subjective user studies to QoE models for QoE monitoring.

3.1 Subjective User Studies for Quantifying YouTube QoE

Subjective user studies are the basis to quantify the YouTube QoE and to model the impact of influence factors on context, user, system, and content level. Therefore, realistic

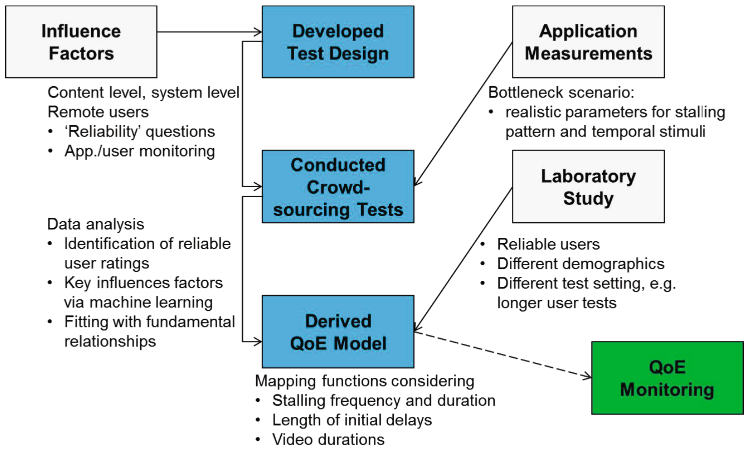

Influence Factors
Content level, system level
Remote users
• 'Reliability' questions
• App./user monitoring
Developed Test Design
Conducted Crowd-sourcing Tests
Data analysis
• Identification of reliable user ratings
• Key influences factors via machine learning
• Fitting with fundamental relationships
Derived QoE Model
Mapping functions considering
• Stalling frequency and duration
• Length of initial delays
• Video durations
Application Measurements
Bottleneck scenario:
• realistic parameters for stalling pattern and temporal stimuli
Laboratory Study
• Reliable users
• Different demographics
• Different test setting, e.g. longer user tests
QoE Monitoring

Fig. 5. Methodology applied from subjective user studies towards QoE models for QoE monitoring [19, 20]

test scenarios will be defined that consider typical video clips and stalling patterns. The general test methodology developed in [21] allows researchers to conduct subjective user tests about YouTube QoE by means of crowdsourcing. Further experiments were conducted in a laboratory environment to double-check the test results and to exclude the influence of the actual test setting and implementation.

Realistic Test Scenarios: Typical Stalling Patterns and Video Clips. The main goal of the experiments is to quantify the impact of system level influence factors, in particular network impairments on QoE. For YouTube video streaming, network impairments result into related *stalling patterns*. Stalling events during the video playout are caused by rebuffering of the video due to an insufficient content delivery rate. For example, if the video bit rate is larger than the available network or server data rate, the video buffer will emptied at some point in time and then the video freezes until the video buffer is filled again. As a consequence, the YouTube user has to wait until the video restarts playing. Furthermore, the user perceived quality suffers from *initial delays* before the YouTube video playout starts, since the player fills up the video buffer before the video playout. In general, the shift from unreliable media streaming to reliable HTTP over TCP streaming makes waiting times one of the key QoE influence factors in the domain of web-based video streaming. In the subjective user studies, these stalling patterns and also initial delays are simulated and then the user is asked about her user perceived quality – in presence of these waiting times.

To obtain realistic stalling patterns, the relationship between network QoS and stalling events must be derived, which is not trivial due to the application-layer traffic management by YouTube (see Section 2.4). In case of a bottleneck with a fixed network data rate, periodic stalling patterns occur [19], i.e., every Δt seconds a stalling event of almost fixed length L takes place. An illustration of the YouTube video buffer evolution in case of a bottleneck is depicted in Figure 6. As soon as the first threshold is exceeded, the video playout starts. However, if the video bit rate is larger than the network data rate (which is here the case due to the bottleneck), the video buffer is emptied faster than the network can deliver video data. As soon as the video buffer falls below a certain threshold [19], the video stalls.

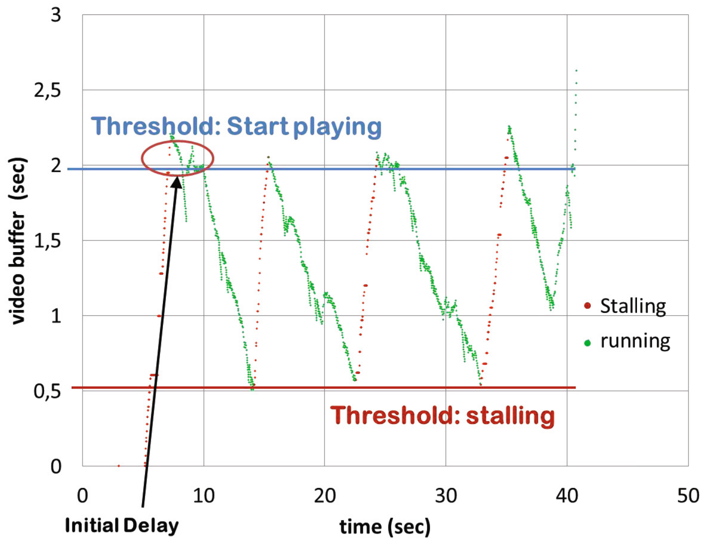


Fig. 6. Implementation of the YouTube video player [19]

On a content level, typical YouTube videos of various content classes like news, sports, music clips, cartoons, etc. were used in the tests. Thereby, the video clips had different resolutions, motion patterns and video codec settings. To reduce the state space of parameters to be tested, only 30 s and 60 s long videos are considered in the test results. On context level, realistic desktop settings are considered. Thus, the video experience in the test should be as similar as possible to a visit of the real YouTube website and the application should run on the users' default web browser.

Crowdsourcing QoE Tests. To conduct subjective user studies for YouTube QoE, crowdsourcing seems to be an appropriate approach. Crowdsourcing means to outsource a task (like video quality testing) to a large, anonymous crowd of users in the form of an open call. Crowdsourcing platforms in the Internet, like Amazon Mechanical Turk or Microworkers, offer access to a large number of geographically widespread users in the Internet and distribute the work submitted by an employer among the users. With crowdsourcing, subjective user studies can be *efficiently conducted at low cost* with an adequate number of users in order to obtain statistically significant QoE scores. In addition, the desktop-PC based setting of crowdsourcing provides a *highly realistic context* for usage scenarios like online video consumption.

However, the *reliability of results* cannot be taken for granted due to the anonymity and remoteness of participants: some subjects may submit incorrect results in order to maximize their income by completing as many tasks as possible; others just may not work correctly due to lack of supervision. To assure the quality of these QoE tests and identify unreliable user ratings, different task design methods are proposed in [21] like including content questions about the videos evaluated or consistency questions. For example, the user is asked about his origin country in the beginning and about his origin continent at the end of the test. The ratings of the participant are disregarded, if not all answers of the test questions are consistent. Furthermore, application-layer monitoring helps to identify reliably conducted tests. E.g. we monitored browser events in order to measure the focus time, which is the time interval during which the browser focus is on the website belonging to the user test. In addition to the crowdsourcing user studies, time-consuming laboratory studies were conducted to double-check the test results. The laboratory studies are described later in this section.

To have a realistic test scenario, the video experience in the test should mimic a visit of the real YouTube website. To this end, an instance of the YouTube Chromeless Player was embedded into dynamically generated web pages. With JavaScript commands the video stream can be paused, a feature we used to simulate stalling and initial delays. In addition, the JavaScript API allows monitoring the player and the buffer status, i.e. to monitor stalling on application layer. In order to avoid additional stalling caused by the test users' Internet connection, the videos had to be downloaded completely to the browser cache before playing. This enables us to specify fixed unique stalling patterns and initial delays which are evaluated by several users. In particular, we varied the number of stalling events as well as the length of a single stalling event, the length of initial delays, but also the video characteristics in the tests.

During the initial download of the videos, a personal data questionnaire was completed by the participant which also includes consistency questions from above. This personal data allows analyzing the impact on user level. Data was collected concerning the background of the user by integrating demographic questions, e.g. about age or profession. Further, the users were asked to additionally rate whether they liked the content. To get insights into the user's expectations and habits in the context of YouTube, we additionally estimated the user's access speed by measuring the time for downloading the video contents and the users were asked about the frequency of Internet and YouTube usage.

The user then sequentially viewed three different YouTube video clips with a predefined stalling pattern. After the streaming of the video, the user was asked to give his current personal satisfaction rating during the video streaming. In addition, we included gold standard, consistency, content and mixed questions to identify reliable subjective ratings. The workers were not aware of these checks and were not informed about the results of their reliability evaluation. Users had to rate the impact of stalling during video playback on a 5-point **absolute category rating (ACR) scale** with the following values: (1) bad; (2) poor; (3) fair; (4) good; (5) excellent. To be more precise, we asked the users the following question "Did you experience these stops as annoying?" with following answer choices: (5) "imperceptible", (4) "perceptible", (3) "slightly annoying", (2) "annoying", (1) "very annoying".

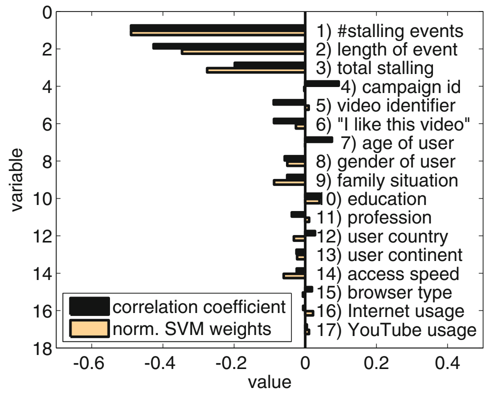
The Microworkers.com platform was used for the crowdsourcing-based QoE tests at the University of Würzburg, since Microworkers allows conducting online user surveys like our YouTube QoE tests. Microworkers supports workers internationally in a controlled fashion, resulting in realistic user diversity well-suited for QoE assessment. The Microworkers platform had about 350 thousand registered users world-wide in the mid of 2012 (see [22] for a detailed analysis of the platform and its users). In total, we conducted seven crowdsourcing campaigns focusing on stalling only and three campaigns focusing on initial delays, respectively. The payment for the crowdsourcing test users was less than 200 €. As described in detail in [21], unreliable test user ratings were identified and the users were filtered out. Throughout the stalling campaigns, 1,349 users from 61 countries participated in the YouTube stalling test and rated the quality of 4,047 video transmissions suffering from stalling. For the initial delay tests, 686 users rated 4,116 videos.

User Studies in Laboratory Environment. In order to validate the crowdsourcing test results and the filtering of unreliable user ratings, similar experiments were carried out in the 'i:lab' laboratory at FTW in Vienna. The user ratings from the crowdsourcing and the lab experiments lead to similar quantitative results and conclusions, see [23, 24]. For the sake of completeness, we shortly describe the lab experiments focusing on initial delays which results are depicted in Figure 8. All other numerical results concerning user ratings stem from crowdsourcing tests as described above.

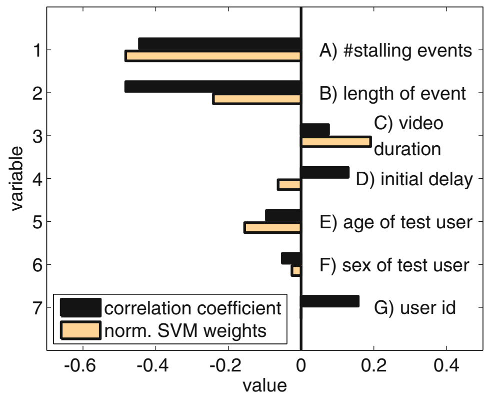
The lab experiment on initial delays contains 41 conditions. The experiment had a total duration of 1.5 h, with an active QoE testing part of about 1 h. The test duration also included a 5 min break in-between the tests and a comprehensive briefing phase at the beginning of the test. Additionally, subjects had to fill out questionnaires about their background, technical experience as well as the current condition, fatigue and cognitive load. After the active testing part, each test was finalized with a debriefing interview and a demographic questionnaire. The QoE testing part consisted of short video clips with a duration of 30 s and 60 s. We used clips out of five different content classes: action trailer, music, animation, documentation and news. After each clip participants were asked to rate the perceived overall quality, including video quality and loading performance, using a 5-point ACR scale on an electronic questionnaire. In total, we collected data from 36 Austrian adults (19 male, 17 female) aged between 20 and 72 years (mean 39.16, median 36.5), recruited by public announcements.

3.2 Key Influence Factors

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. Therefore, we analyze correlation coefficients as well as support vector machine (SVM) weights [21]. The Spearman rank-order correlation coefficient between the subjective user rating and the above mentioned variables is computed. In addition, SVMs are utilized to obtain a model for classification: Every variable gets a weight from the model indicating the importance of the variable. However, SVMs are acting on two-class problems only. Therefore, the categories 1 to 3 of the ACR scale to the "bad quality" class and the categories 4 to 5 to the "good quality" class.



(a) Stalling parameters, video characteristics, demographics [21]



(b) Stalling parameters, video duration, initial delays [24]

Fig. 7. Identification of key influence factors on YouTube QoE

Figure 7a shows the results from the key influence analysis. On the x-axis, the different influence factors are considered, while the y-axis depicts the correlation coefficient as well as the SVM weights which are normalized to the largest correlation coefficient for the sake of readability. We can clearly observe from both measures that *the stalling parameters dominate and are the key influence factors*. It is interesting to note that the user ratings are statistically independent from the video parameters (such as resolution, video motion, type of content, etc.), the usage pattern of the user, as well as its access speed. In particular, we used different video contents, but got no differences on the user ratings. A possible explanation may be that a video user does not care about the video quality (i.e. which encoding scheme is used, what is the video bit rate, etc.). If the video stalls, the video experience of the user is disturbed – independent of the actual video characteristics. Thus, for a YouTube QoE model, those video characteristics can be neglected – in contrast to the actual stalling pattern. However from a networking point of view, higher video bitrates lead to more stalling events if there is a bottleneck in the network. Hence, the video bit rate may be considered for QoE monitoring (see Section 4.2) to estimate the number of stalling events (which is the relevant for the QoE model).

However, the videos considered in the experiments [21] had a fixed length of 30 s and no initial delays for buffering the video contents were considered. Therefore, a second row of experiments was conducted [24] in which the following parameters were varied: number of stalling events, duration of a single stalling event, video duration, initial delay. Hence in addition to the first row of subjective studies, the video duration and initial delays were considered as influence factors. To check again the influence of user demographics on QoE, the age, sex, and user id were also considered in the analysis.

The results in Figure 7b reveal again that *the number of stalling events together with the stalling length are clearly dominating the user perceived quality*, while the user demographics have no influence. Furthermore, the impact of initial delays is statistically not significant. We take a closer look at initial delays that may be accepted by the user for filling up the video buffers to avoid stalling. In case of bad network conditions, providers need to select a trade-off between these two impairment types, i.e. stalling or initial delays, which allows QoE management for YouTube video streaming clouds [25]. Therefore, we ask the question whether initial delays are less harmful to QoE than stalling events for YouTube. However, the results in Figure 8 show that no statistical differences are observed for video clips of 30 s and 60 s regarding the impact of initial delays on the QoE. QoE is thereby quantified in terms of **Mean Opinion Scores (MOS)** which is the average value of user ratings on the ACR scale for a given test condition, i.e. a certain initial delay in that case.

3.3 A Model for QoE Monitoring

The identification of key influence factors has shown that YouTube QoE is mainly determined by stalling frequency and stalling length. To quantify YouTube QoE and derive an appropriate model for QoE monitoring, we first provide mapping functions from stalling parameters to MOS values. Then, we provide a simple model for YouTube QoE monitoring under certain assumptions. Finally, we highlight the limitations of the model.

QoE Mapping Functions. As fundamental relationship between the stalling parameters and QoE, we utilize the IQX hypothesis [26] which relates QoE and QoS impairments x with an exponential function $f(x) = \alpha e^{-\beta x} + \gamma$. In [21], concrete mapping functions for the MOS values depending on these two stalling parameters, i.e. number N of stalling events and length L of a single stalling event, were derived. To be more precise, YouTube videos of 30 s length were considered in the bottleneck scenario leading to period stalling events. In order to determine the parameters α, β, γ of the exponential function, nonlinear regression was applied by minimizing the least-squared errors between the exponential function and the MOS of the user ratings. This way we obtain the best parameters for the mapping functions with respect to goodness-of-fit.

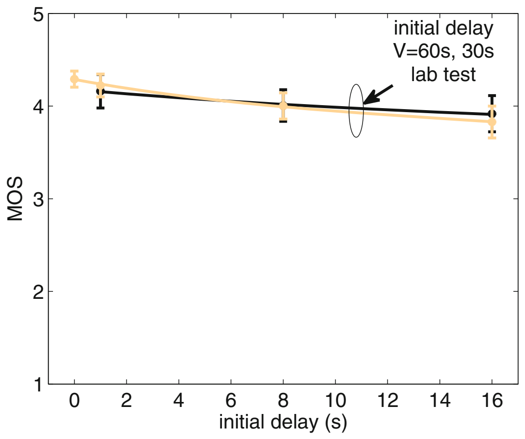


Fig. 8. Initial delays have almost no influence on MOS for videos of duration 60 s and 30 s – compared to influence of stalling length [24]

In this work, however, the aim is to derive a model for monitoring YouTube QoE. Therefore, we reduce the degree of freedom of the mapping function and fix the parameters α and γ. If we consider as QoS impairment x either the number of stalling events or the stalling duration, we observe the following upper and lower limits for the QoE $f(x)$, i.e. $\lim_{x\to 0} f(x) = \alpha + \gamma$ and $\lim_{x\to\infty} f(x) = \gamma$, respectively. In case of no stalling, i.e. $x = 0$, the video perception is not disturbed and the user perceives no stalling. As we asked the user "Did you experience these stops as annoying?", the maximum MOS value is obtained, i.e. $\alpha + \gamma = 5$. In case of strong impairments, however, i.e. $x \to \infty$, a well-known rating scale effect in subjective studies occurs. Some users tend to not completely utilize the entire scale, i.e. avoiding ratings at the edges leading to minimum MOS values around 1.5 [27]. Hence, we assume $\alpha = 3.5, \gamma = 1.5$ and derive the unknown parameter β from the subjective user tests. The obtained mapping functions as well as the coefficient of determination R^2 as goodness-of-fit measure are given in Table 3. In particular, the mapping function $f_L(N)$ returns the MOS value for a number N of stalling events which have a fixed length L. It can be seen that R^2 is close to one which means a very good match between the mapping function and the MOS values from the subjective studies.

Figure 9 depicts the MOS values for one, two, three and four seconds stalling length for varying number of stalling events together with exponential fitting curves (as discussed in [26]). The x-axis denotes the number of stalling events, whereas the y-axis denotes the MOS rating. The results show that users tend to be highly dissatisfied with two or more stalling events per clip. However, for the case of a stalling length of one second, the user ratings are substantially better for same number of stalling events. Nonetheless, users are likely to be dissatisfied in case of four or more stalling events, independent of stalling duration. As outlined in the previous chapter of this book "From Packets to People: Quality of Experience as a New Measurement Challenge", most of the users accept a quality above 3 on the ACR scale, i.e. a fair quality.

Table 3. Parameters of mapping functions (see Figure 9) of stalling parameters to MOS together with coefficient of determination R^2 as goodness-of-fit measure

event length L	mapping function depending on number N of stalling events	R^2
1 s	$f_1(N) = 1.50 \cdot e^{-0.35 \cdot N} + 3.50$	0.941
2 s	$f_2(N) = 1.50 \cdot e^{-0.49 \cdot N} + 3.50$	0.931
3 s	$f_3(N) = 1.50 \cdot e^{-0.58 \cdot N} + 3.50$	0.965
4 s	$f_4(N) = 1.50 \cdot e^{-0.81 \cdot N} + 3.50$	0.979

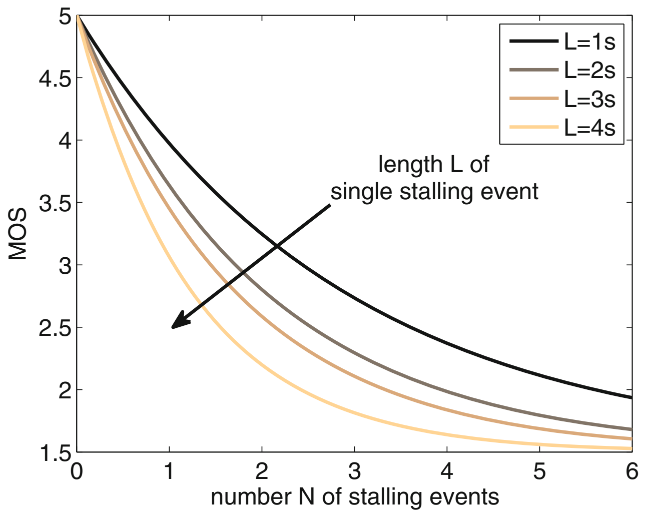


Fig. 9. Mapping functions of stalling parameters to MOS [21]. Video duration is fixed at 30 s. No initial delay is introduced. Parameters are given in Table 3.

It has to be noted that it is not possible to characterize the stalling pattern by a simple total stalling duration $T = L \cdot N$ only, as the curves for $f_L(N)$ depending on the total stalling duration $T = L \cdot N$ differ significantly [20]. Therefore, *stalling frequency and stalling length have to be considered individually in the QoE model.*

Simple Model for QoE Monitoring. Going beyond the pure mapping functions, we develop next an appropriate QoE model for monitoring. The intention of the monitoring is to provide means for QoE management [20] for ISPs or the video streaming service provider. Hence, the model has to consider an arbitrary number N of stalling events and stalling event length L, while the subjective user studies and the provided mapping functions $f_L(N)$ in the previous section only consider a finite number of settings, i.e. $L \in \{1, 2, 3, 4\}$ s. As a result of the regression analysis in the previous section, the parameter β_L of the exponential mapping function $f_L(N) = 3.5^{\beta_L N} + 1.5$ is obtained as given in Table 3.

Figure 10 shows the obtained parameter β_L depending on the length L of a single stalling event. As four settings for L were tested in the subjective user studies, the figure contains four different dots (L, β_L) together with the actual 95 % confidence interval for the nonlinear least squares parameter estimates β_L. Except for the obtained parameter value β_3 for $L = 3$ s, all other values β_L lie on a straight line. Furthermore, the confidence interval of the parameter β_3 overlaps the line. Therefore, we assume in the following a linear relationship between β_L and the event length L which can be easily found as $\beta(L) = 0.15L + 0.19$.

As simple QoE model $f(L, N)$, we therefore combine our findings, i.e. $f_L(N)$ and $\beta(L)$, into a single equation taking the number of stalling events N and the stalling length L as input

$$f(L, N) = 3.50e^{-(0.15L+0.19)\cdot N} + 1.50 \quad \text{for} \quad L \in \mathbb{R}^+, N \in \mathbb{N}\,. \tag{QoE}$$

Figure 11 illustrates the obtained model for YouTube QoE monitoring as surface plot. On the x-axis the number N of stalling events is depicted, on the y-axis the stalling event length L, while the actual MOS value $f(L, N)$ according to Eq.(QoE) is plotted on the z-axis. The figure clearly reveals that the number of stalling events determines mainly the QoE. Only for very short stalling events in the order of 1 s, two stalling events are still accepted by the user with a MOS value around 3. For longer stalling durations, only single stalling events are accepted.

Other Influence Factors and Limitations of the Model. The scope of this section is to clearly summarize the assumptions and limitations of the provided QoE model as long as the implications for YouTube QoE monitoring. First, we analyze the impact of stalling on YouTube videos of two different durations of $V = 30$ s and $V = 60$ s, respectively. We consider now a single stalling event only ($N = 1$) and vary the stalling duration L from 0 s to 8 s. Figure 12 shows the exponential fitting functions $g_V(L) = 3.5^{\beta_L \cdot L} + 1.5$ of the user ratings in the subjective tests. We see that the curves g_{30} and g_{60} are deviating significantly from each other. Thus, the MOS for the same stalling duration shows significant differences for different video durations: a video with a stalling event of length 8 s is rated 3.30 and 2.51 for a video duration of 60 s and 30 s respectively. Therefore, the video duration must also be taken into account in a proper QoE model. However, it has to be noted that the video duration only plays a role if there are only a very few stalling events (compared to the video duration). Otherwise, the actual number of stalling events dominates the user perceived quality. Nevertheless, Figure 12 depicts the curve for the QoE model in Eq.(QoE) and reveals some limitations of the provided QoE model for monitoring. For longer video durations, the MOS is higher compared to shorter clips with same stalling patterns. Hence, the provided QoE model (obtained for short video clips of 30 s) underestimates the actual QoE. Therefore, QoE monitoring based on this model will give some lower bounds which is desired for QoE management and the avoidance of any QoE degradation of the video streaming service.

Additional assumptions and limitations of the provided QoE model are as follows. In the subjective tests only typical YouTube video formats were considered, however, more

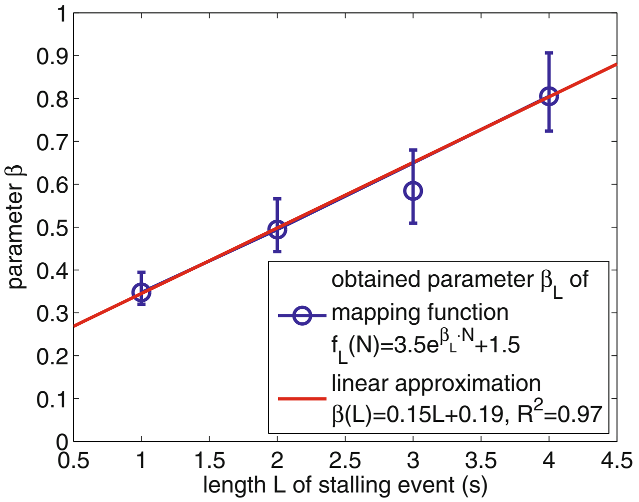


Fig. 10. Parameter β_L of obtained mapping function for given length L of single stalling event. A linear approximation yields a high goodness-of-fit R^2 close to 1

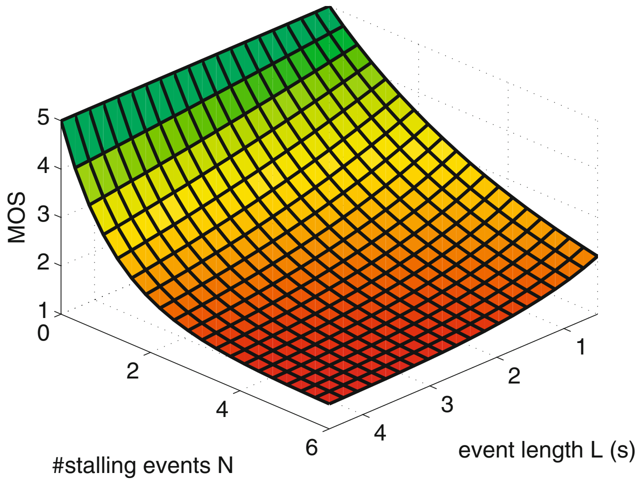


Fig. 11. Simple QoE model maps a number N of stalling events of average length L to a MOS value, $f(L, N) = 3.50 \cdot e^{-(0.15 \cdot L + 0.19) \cdot N} + 1.50$

"extreme" scenarios e.g. very small resolution vs. HD resolution are a subject of future work. Furthermore, the applied stalling pattern considers a bottleneck in the network or at the server side with a constant data rate. This leads to a periodic stalling pattern in which the duration of a single stall event has a fixed duration [21]. Arbitrary stalling patterns due to networks with varying bottleneck capacities are not investigated so far due to the lack

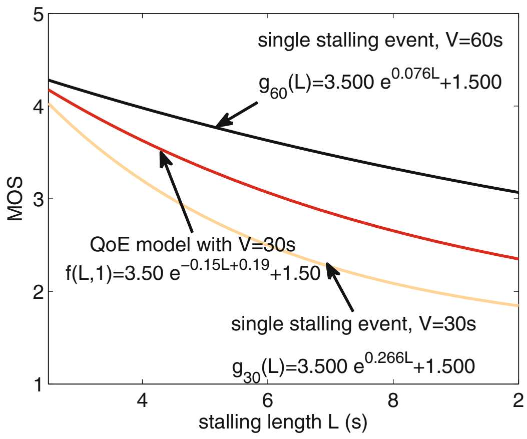


Fig. 12. Influence of stalling length on MOS for video duration of 60 s and 30 s, respectively [24]. A single stalling event is considered without any initial delays.

of related arbitrary stalling models and corresponding subjective user studies. Hence, the impact of different traffic patterns on YouTube QoE is also a matter of future research. As mentioned above, a general relationship between the ratio of the stalling duration and the video duration onto YouTube QoE still is also of interest. Further assumptions of the model are the exponential relationship according to the IQX hypothesis with fixed parameters α and γ due to limits of the MOS values. In addition, we assume a linear relationship between the parameter $\beta(L)$ and the length of a stalling event L.

In summary, *the concrete MOS values depend on the video duration and the actual stalling pattern*. From a QoE management perspective, stalling must be avoided to keep YouTube users satisfied. Even very short stalling events of a few seconds already decrease user perceived quality significantly. However, initial delays are tolerated up to a reasonable level. Hence, YouTube QoE monitoring should pro-actively detect imminent stalling, so that QoE control mechanisms are triggered timely in advance – which is possible with the provided QoE model in the previous section.

4 Monitoring YouTube QoS and QoE

This section provides an overview of different QoE monitoring approaches that have been investigated for YouTube video streaming. As already discussed in the previous section, YouTube QoE is primarily determined by stalling effects on the application layer, as opposed to UDP-based video streaming common for traditional IPTV services. Consequently, the central challenge for QoE monitoring is the *robust detection of stalling events* as they occur during playback. With the number of stalling events and the average length of a single stalling event as input parameters, the QoE model in Eq.(QoE) can be applied to quantify the user perceived quality.

In this context, we distinguish between two categories of monitoring approaches, which differ in terms of measurement point and layer at which the information is captured. **Client**-level monitoring approaches require modifications of the player application or the installation of additional monitoring clients, but can provide exact results since performance (i.e. stalling patterns) is directly measured at the end user's terminal at the application layer. **Network**-level approaches aim to detect impairments solely from measurements within the network, typically performed at ISP level. We discuss several network-layer monitoring approaches suitable for estimating YouTube QoE, as well as their accuracy (compared to application-level monitoring) and implementation aspects.

4.1 Client-Level Monitoring

As far as client-level monitoring is concerned, most of the related work has been conducted in the context of QoE optimization and traffic management. Staehle et al. [28] present 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 [29] 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 has been used in [30] to develop a novel dynamic network stack based on functional blocks for optimizing the QoE of YouTube playback. The underlying monitoring approach termed **YoMo** is discussed in the following.

C1. YoMo– An Application-level Comfort Monitoring Tool. The YouTube monitoring tool YoMo [28] measures the video player buffer status directly at the end user site on application layer. This allows predicting an imminent stalling taking into account user interactions as pausing the video or jumping within the video, but requires an instance of YoMo running at the user device. Additional challenges for monitoring at the end user level address privacy concerns of users as well as trust and integrity issues due to cheating users to obtain better performance by fraud. However, monitoring at the end-user gives the best view on user perceived quality and YoMo itself is a lightweight Java tool cooperating with a Firefox plugin. The YoMo tool performs several tasks [28]:

- Detect a YouTube flow and forward this information to the mesh advisor which is able to trigger adequate resource management tools in order to avoid a QoE degradation.
- Analyze the packets of the YouTube flow to calculate the video buffer status β.
- Constantly monitor β and raise an alarm if this falls below a critical threshold.

YoMo monitors the client's incoming traffic and identifies a YouTube video flow by detecting the FLV signature. The data of each YouTube video flow is continuously parsed in order to retrieve the information embedded in the FLV tags. In particular, each FLV tag includes the time when the frame is to be played out, allowing to derive the currently available playtime T preloaded in the video buffer.

YoMo estimates the **buffer level** β by computing the difference between playtime T and current **video position** t. However, while the **playtime** T can be derived as the time stamp of the last completely downloaded FLV tag, the current video position t cannot be obtained from the FLV tags, but from the YouTube player only which can be accessed by the YouTube API with scripting languages. Therefore, YoMo uses a simple

Firefox extension which retrieves t from the YouTube player and sends it to the YoMo software. In [28] further technical details are described extensively. Moreover, the paper shows that in the case of a sudden connection interruption, YoMo is able to predict the time of the video stalling event with an accuracy of about 0.1 s. YoMo and the Firefox plugin may be downloaded from the G-Lab website[1].

C2. Pytomo – Analyzing Playback Quality of YouTube Videos. In the context of traffic characterization and measurement of playback performance, another client-level monitoring has been developed by Juluri *et al.* [31]: Pytomo.

The purpose of this Python-based tool is to measure the download and analyze the playback performance of YouTube videos. To this end, it emulates user by downloading a given YouTube video and then selects a number of random related links for further downloads. In this respect Pytomo is not a monitoring tool running in the background like YoMo but rather a *crawler* that actively downloads content in order to perform its measurements. Pytomo's crawling process can be summarized as follows:

– an initial set of YouTube videos is chosen (by default the most popular videos of the week);
– for each selected video, the cache-URL of the video server hosting the clip file is obtained;
– the (possibly different) IP addresses of the video server are obtained by querying three different DNS servers;
– the ping statistics are collected for each resolved IP address of the video servers;
– from each resolved IP address the first 30 seconds of the video is downloaded at the default video format (640x390)[2];
– the crawl continues with the next video.

For each video download, Pytomo collects the following statistics: ping statistics, video information, download statistics, playback statistics such as initial buffer, number of stalling events, total buffering duration, buffer duration at the end of the download (see [31] for a detailed description). Similar to the findings of section Section 3, the authors recommend the number of stalling events and total buffering duration as main indicators for playback QoE.

Like YoMo, Pytomo estimates the playout buffer level by comparing video playtime with the video position. To derive the stalling patterns, this information feed into a model of the YouTube player. While Pytomo relies on the same application-level quality predication approach, it can be considered complementary to YoMo: as a crawler, it actively measures YouTube QoE with high repeatability, albeit at its current state does not predict end user QoE. Furthermore, Pytomo allows to study the impact of the DNS resolver used on playback quality, which as shown in [31], can have a considerable impact. Pytomo is a GPL software and can be freely downloaded from `http://code.google.com/p/pytomo`.

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

[2] In order to perform a video download, Pytomo first resolves the IP address of the content server and then uses this IP address to perform the analysis. This ensures that the analysis and video download are being performed on the same server.

4.2 Network-Level Monitoring

While being highly accurate, client-level approaches are not applicable in the case of an ISP interested in monitoring YouTube QoE inside its network. The main reason is that 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.

In the case of YouTube, the main challenge is the accurate approximation of the stalling events that take place at the application layer using network packet traces only, which is non-trivial. One example of work in this field is [32], where the authors present an approach for measuring YouTube stalling events from packet traces. The paper presents some first interesting results, but is too limited in terms of number of analyzed videos to draw conclusions on the accuracy of the approach. Moreover, no QoE models or estimations are derived from this analysis. In contrast, other works focus on monitoring YouTube stalling events and estimating the corresponding QoE levels, relying exclusively on passive monitoring of TCP flows. Previously in [33], we have presented and compared three different passive monitoring approaches, referred to as 'M1', 'M2', and 'M3'. The first approach M1 is based on the download time of the whole video; M2 relies on measuring the end-to-end throughput of the connection; finally, M3 approximates the actual video buffer status. All three approaches allow to estimate the stalling pattern without relying on application-level or client side measurements, however with different levels of accuracy and complexity. In general, 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 [19]. Assuming a known distribution for L, we can formulate the first and most simple monitoring approach:

M1. Download Time vs. Video Duration. In this approach, the 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 $N = {}^{T}/_{L}$ (cf. [33]). Note that 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 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 and the video id. An alternative 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, frame rate and key frames are specified. In that case, the monitoring system has to parse the network packets and needs to understand the container format. Hence, both options require some extra effort for the system to get the video duration. However, the major drawback of M1 is that it uses the *total* duration of the video as input. Indeed, if the user does not watch the entire video and thus aborts downloading before the end, which is very frequent in practice, this monitoring approach cannot be applied. For this reason, a more complex approach is required for passive probing scenarios.

M2. Network Throughput vs. Video Encoding Rate. The second approach is based on the stalling frequency approximation in [21]. Based on a considerable body of measurement data, the authors show that the frequency F of stalling events can be well approximated with an exponential function

$$F(x) = -1.09e^{-1.18x} + 0.36 \tag{1}$$

with x being the normalized video demand $x = {}^{V}\!/_{B}$ defined as ratio of video encoding rate V and download throughput B. In that case, the bottleneck capacity B has to be estimated, which can be easily done from passive monitoring packet traces and throughput measurements [34]. 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 current 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 for M1 and M2 is comparable, but M2 can also be applied to cases where the user does not download and watch 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, L can be approximated by a t location-scale distribution [19]. 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.

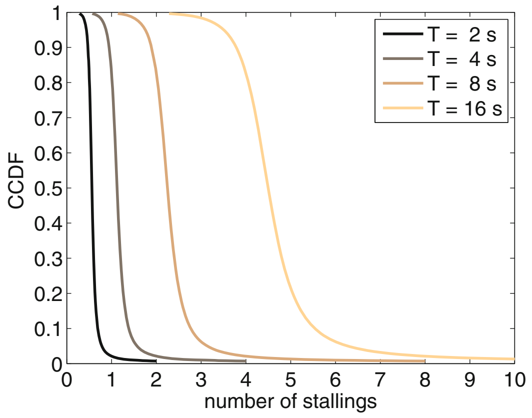


Fig. 13. Monitoring approaches M1 and M2 can only estimate the number of stalling events with a certain probability [33]

Figure 13 shows the complementary cumulative distribution function of the number of stalling events $N = F^{-1}$ 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, 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 of 8 s, the number of stalling events 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 [21], 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 practice an ISP can use both approaches only for upper or lower bound estimations of QoE.

For some ISP, it may be sufficient to determine whether stalling occurs or not and how the user reacts in response. Therefore, we define the QoS metric **reception ratio** ρ as ratio between download throughput B and video encoding rate V, i.e.

$$\text{reception ratio } \rho = \frac{B}{V}\,.$$

Although the reception ratio cannot be directly related to QoE, it is a good indicator if there are problems in the network. We demonstrate this by relating the reception ratio to the user behavior based on our work in [11], which is one of the first attempts to quantify the impact of playback quality on the viewing behavior.

A download throughput lower than encoding rate should result in interrupted playback: thus we define the **reception quality** as QoE indicator in the following way,

- if reception ratio > 1, we consider the video has **good reception quality**;
- otherwise we consider the video has **poor reception quality**.

This metric may first seem quite crude, so we have used active measurements from [15] to evaluate its accuracy. The dataset used consists of eight packet traces of one hour collected between 2008 and 2011 (see [11] for details). We use two standard metrics usually used in pattern recognition and information retrieval, namely **precision** and **recall**. They are based on the concepts of

- *True Positive **TP**:* reception ratio > 1 and the video had no stall;
- *False Positive **FP**:* reception ratio > 1 but the video had at least one stall;
- *True Negative **TN**:* reception ratio < 1 and the video had at least one stall;
- *False Negative **FN**:* reception ratio < 1 but the video had no stall.

Out of these notions, we build these evaluation metrics:

- **recall** = TP / (TP + FN): this corresponds to the fraction of uninterrupted videos correctly evaluated;
- **precision** = TP / (TP + FP): this corresponds to the ratio of uninterrupted videos in the videos with reception ratio > 1.

For the data set of December 2011 we obtain 91.8% of recall and 88.5% of precision. While the overall performance of this indicator is surprisingly good, it suffers from the

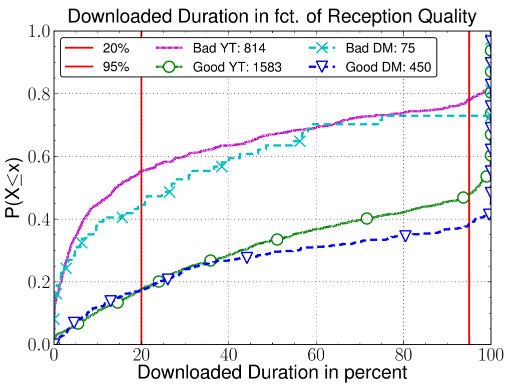


Fig. 14. Fraction of video downloaded as function of video reception quality for YouTube (YT) and Dailymotion (DM) (from [11])

following two limitations [19]. First, if the video duration is very short, sufficient data is downloaded before the video playout starts and no stalling occurs. Second, stalling is caused by the variability of the video bit rate, which may happen even when the network capacity is larger than the video bit rate.

Furthermore, an ISP is interested in the relation between the user perception and the user behavior. Therefore, we analyze in the following the fraction of the video downloaded (as user behavior indicator) depending on the reception quality (as QoE indicator).

In Figure 14, we plot the CDF of the *fraction of the video downloaded*. We distinguish between videos with good reception quality and those with poor reception quality. We see that under good playback quality, less than 50% YouTube transfers are aborted before the end, whereas with poor playback quality, as much as 80% of the transfers are aborted before the end. We have also included another popular video streaming site, namely DailyMotion. The results are quite similar for both video streaming sites, which justify the claim that reception quality influences the viewing behavior.

In Figure 15, we distinguish each video according to its duration and the fact that it has been completely downloaded (at least 90% of video downloaded):

– short videos ($\leq$ 3 minutes);
– long videos ($\geq$ 3 minutes) and completely downloaded;
– long videos ($\geq$ 3 minutes) and not completely downloaded;

We plot the downloaded duration *vs.* the content duration for YouTube videos: in Figure 15a for videos with good reception quality, and in Figure 15b for videos with poor reception quality. We observe that in case of good reception quality, 34% of the videos have a downloaded duration of 3 minutes or more, while in case of poor reception quality their share drops to only 15%. This confirms that the reception quality influences the behavior of the user, and this influence is more pronounced for long videos.

M3. YiN- YouTube in Networks Based on Playout Buffer Approximation. The third passive monitoring approach M3, also called **"YiN"** in [33], detects YouTube video flows like for client-level monitoring (see Section 4.1). It extracts video information from network packet data. In particular, the size and time stamps of (audio and video) frames are retrieved by means of deep packet inspection. Together with the YouTube video player parameters in particular the playing threshold Θ_1 and the stalling threshold Θ_0 (see Figure 6), the playout video buffer status is reconstructed on behalf of network data only at high accuracy. As soon as the YouTube video buffer exceeds Θ_1, the player starts the video playback. If the buffer underruns Θ_0, the video stalls. The player parameters are determined in [33] based on the application-layer measurements in [19]. However, it has to be noted that there may small deviations of these values from video to video in practice, since the player takes into account the actual structure of the video codec for optimized video playout. Consequently, such small errors may propagate and lead to inaccuracies in practice.

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

On behalf of these measures the stalling pattern over time, i.e. over the TCP acknowledgments, can be computed as follows [33].

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

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

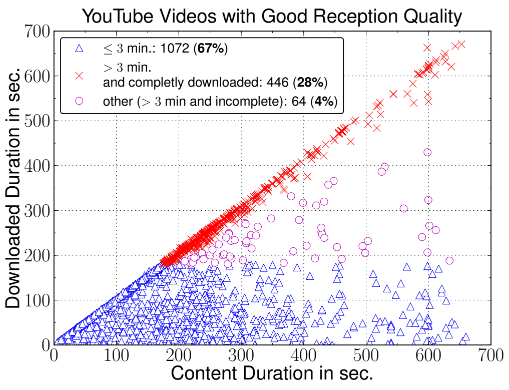
$$\rho_i = \rho_{i-1} + \begin{cases} 0, & \text{if } \psi_i \\ t_i - t_{i-1}, & \text{if } \neg\psi_i \end{cases} \tag{4}$$

$$\beta_i = \tau_i - \rho_i \tag{5}$$

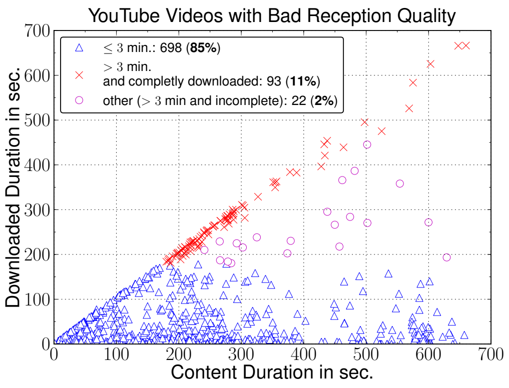
The actual video buffer can then be approximated by the difference between the frame time τ_i and the actual play time ρ_i. The iterative computation of the different variables is initialized in the following way, since YouTube first starts playing until the threshold Θ_1 is exceeded to fill the video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

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

QoS and QoE-Level Evaluation of Network Monitoring Approaches. As already outlined above (cf. Figure 13), the monitoring approaches M1 and M2 can only estimate the number of stalling events with a certain probability. Hence, their accuracy is not sufficient for proper QoE monitoring. In this context, the concept of reception ratio as



(a) Good reception quality videos



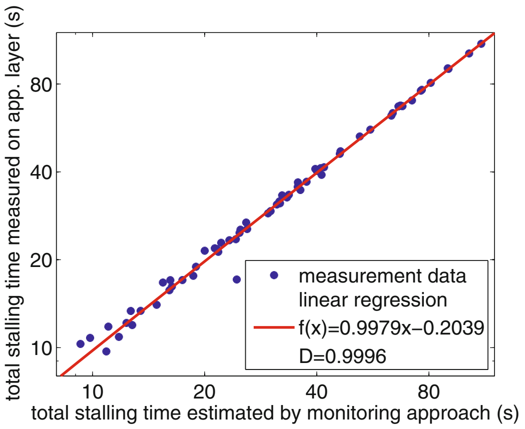
(b) Bad reception quality videos

Fig. 15. Fraction of video downloaded as function of video length (from [11])

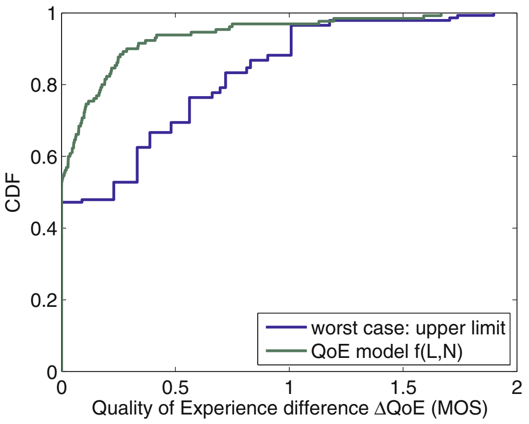
ratio between download throughput and video encoding rate was introduced to indicate whether stalling occurs or not.

To evaluate the accuracy of the YiN monitoring approach, which aims to retrieve the stalling pattern, the estimated video buffer was compared with the actual video buffer measured at application layer, which serves as ground truth. The measurements took place from June 2011 to August 2011 in a laboratory at FTW in Vienna (see [33] for further details). Furthermore, in a second step the stalling patterns were mapped to QoE according to the YouTube QoE model (Section 3.3) and the difference between 'measured' and 'estimated' QoE based on the reconstructed stalling patterns were compared.

In this respect, the results in [33] show that YiN is the most accurate approach that predicts that stalling pattern almost exactly with a coefficient of correlation between measured and estimated values of about 0.9998. Figure 16a illustrates this very strong correlation between the total stalling time estimated from packet traces and the total stalling time as measured on the application-layer.



(a) QoS: Measured vs. estimated total stalling times



(b) QoE: Difference $\varDelta QoE$ between measurements on application layer and estimation

Fig. 16. Network-levl monitoring YiN (from [33]): Comparison of application-layer measurements and approximations based on network-level information

Nevertheless, the remaining measurement error can lead to strong QoE differences due to the end user's non-linear perception of stalling that is also reflected in the QoE model discussed in Section 3.3. We compare our results considering two different evaluation scenarios: (i) the QoE model evaluation using the real stalling patterns, and (ii) a worst case evaluation, which provides an upper bound to the QoE estimation difference. For the worst case evaluation, short stalling events of 1 s length are considered, which sum up to the total stalling time. This is a worst case scenario because it leads to a higher number of stalling events than actually observed. Again, the difference $\varDelta QoE$ between 'measured' and 'estimated' QoE based on the reconstructed stalling patterns are compared.

The cumulative distribution functions of the ΔQoE values obtained in both scenarios are depicted in Figure 16b. The QoE difference w.r.t. the QoE model is almost zero for about 60 % of the analyzed videos. As worst case upper bound, two thirds of the analyzed videos show a QoE difference below 0.5. However, differences can be as large as one step on the MOS scale, as observed for 10 % of the videos. Thus, the YiN monitoring approach may estimate good quality (MOS 4), while the users actually only experience a fair quality (MOS 3). The main reason for these inaccuracies is – as described above – error propagation; since according to end-user quality perception and the underlying mapping from stalling QoS to YouTube QoE are highly non-linear, a relatively small measurement error can result in aforementioned MOS differences. For example, when the number of stalling events is very low, one stalling more or less already makes a huge difference in QoE. As a consequence, one has to take these error margins into account and set alarm thresholds accordingly [20].

Nonetheless, the above results demonstrate that an accurate reconstruction of stalling events from network-level measurement data is possible, and that YouTube QoE monitoring at ISP-level is feasible. However, only the most accurate and unfortunately the most complex YiN approach can be actually used for QoE monitoring purposes, since (i) stalling frequency and stalling duration both need to be measured, and (i) the non-linearity of human perception demands for high QoS measurement accuracy, particularly in those cases where stalling frequency is low.

5 Conclusion

In this chapter, we have presented the YouTube delivery infrastructure and we have investigated YouTube video streaming in terms of QoE impact of Internet delivery as well as resulting QoE monitoring aspects. To this end, we discussed the key mechanisms used by YouTube: Cache selection plays an important role in YouTube and we showed that surprisingly, cache server selection is highly ISP-specific and that geographical proximity is not the primary criterion, for reasons that need further investigation. In addition, DNS level redirections for load-balancing purposes occur quite frequently and can considerably increase the initial startup delay of the playback (in the order of seconds). However, the results from subjective user studies showed that initial delays up to about ten seconds have no severe impact on QoE. Hence, QoE models and QoE monitoring approaches may neglect those initial delays.

From a QoE management perspective, the smooth playback of the video rather than visual image quality is the key challenge, since YouTube uses HTTP via TCP to deliver the video. We saw that, more than any other impairment, stalling events (i.e. playback interruptions) have a dramatic impact on the QoE and should be avoided at any price. The monitoring of the stalling frequency and duration is the prerequisite for proper QoE monitoring. In this context, the throughput-based reception ratio plays a key role as QoE-relevant metric for predicting buffer under-runs. Concerning QoE monitoring, we compared several network-level and client-level approaches with regard to their accuracy to detect stalling pattern. As expected, this is more difficult for network-level approaches, which have to reconstruct client-level stalling patterns from network traffic information only. However, our evaluation of the YiN algorithm shows that this is feasible, albeit at the cost of increased demand for computational performance.

As far as future work is concerned, the QoE management for video streaming to smartphones remains an open issue. The mobile environment will lead to different traffic and stalling patterns that need to be evaluated from a QoS and QoE monitoring perspective accordingly.

Acknowledgements. The results presented in this chapter summarize a successful research period starting 2010. The different facets discussed in this chapter (subjective test design, crowdsourcing, QoE model, monitoring approaches, CDN structure) were supported by different research projects. In particular, the research has been supported by COST TMA Action IC0703 through two short term scientific missions by T. Hoßfeld ("Time Dynamic Modeling of Quality of Experience (QoE) for Web Traffic" and "QoE-based Bandwidth Dimensioning for ISPs to Support Online Video Streaming"); COST QUALINET Action IC1003 through the STSM "Modeling YouTube QoE based on Crowdsourcing and Laboratory User Studies" by T. Hoßfeld; by the European FP7 Network of Excellence "Euro-NF" through the Specific Joint Research Project "PRUNO"; and the project G-Lab, funded by the German Ministry of Educations and Research (Förderkennzeichen 01 BK 0800, G-Lab). 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. This work was funded by Deutsche Forschungsgemeinschaft (DFG) under grants HO 4770/1-1 and TR257/31-1. The authors alone are responsible for the content of the paper.

The authors would like to give special thanks to Alexander Platzer and Pedro Casas for the fruitful discussions on YouTube monitoring approaches; Michael Seufert and Matthias Hirth for implementing, running and improving the crowdsourcing experiments for QoE tests; and Sebastian Egger for all discussions on YouTube QoE and carrying out the laboratory tests. Further, we would like to thank the responsible editor Maja Matijasevic and the anonymous reviewers for their valuable comments and suggestions to improve this chapter.

References

1. Cisco: Cisco visual networking index: Forecast and methodology, 2011–2016 (May 2012)
2. Plissonneau, L., Vu-Brugier, G.: Mobile data traffic analysis: How do you prefer watching videos? In: Proc. of 22th International Teletraffic Congress (ITC'22), Amsterdam, Netherlands (September 2010)
3. Maier, G., Feldmann, A., Paxson, V., Allman, M.: On dominant characteristics of residential broadband internet traffic. In: Proc. of 9th ACM SIGCOMM Internet Measurement Conference (IMC 2009), Chicago, Illinois, USA (November 2009)
4. Kontothannsis, L.: Content Delivery Consideration for Web Video. In: Keynote at ACM Multimedia Systems 2012 (MMSys 2012), Chapel Hill, North Carolina, USA (2012)
5. Brodersen, A., Scellato, S., Wattenhoefer, M.: YouTube Around the World: Geographic Popularity of Videos. In: Proc. of 21th International World Wide Web Conference (WWW 2012), Lyon, France (April 2012)
6. Ghobadi, M., Cheng, Y., Jain, A., Matthis, M.: Trickle: Rate Limiting Youtube Video Streaming. In: Proc. of 2012 USENIX Annual Technical Conference, Boston, MA, USA (June 2012)

7. Adhikari, V.K., Jain, S., Chen, Y., Zhang, Z.L.: Vivisecting YouTube: An Active Measurement Study. In: Proc. of IEEE INFOCOM 2012 Mini-Conference, Orlando, Florida, USA (March 2012)
8. Adhikari, V.K., Jain, S., Zhang, Z.L.: YouTube Traffic Dynamics and Its Interplay with a Tier-1 ISP: An ISP Perspective. In: Proc. of 10th ACM SIGCOMM Internet Measurement Conference (IMC 2010), Melbourne, Australia (November 2010)
9. Adhikari, V.K., Jain, S., Zhang, Z.L.: Where Do You "Tube"? Uncovering YouTube Server Selection Strategy. In: Proc. of International Conference on Computer Communications and Networks (ICCCN 2011), Maui, Hawaii, USA (August 2011)
10. Finamore, A., Mellia, M., Munafo, M., Torres, R., Rao, S.: YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience. In: Proc. of 2011 ACM SIGCOMM Internet Measurement Conference (IMC 2011), Berlin, Germany (November 2011)
11. Plissonneau, L., Biersack, E.: A Longitudinal View of HTTP Video Streaming Performance. In: Proc. of ACM Multimedia Systems 2012 (MMSys 2012), Chapel Hill, North Carolina, USA (February 2012)
12. Torres, R., Finamore, A., Kim, J.R., Mellia, M., Munafo, M.M., Rao, S.: Dissecting video server selection strategies in the youtube cdn. In: Proc. of 31st International Conference on Distributed Computing Systems (ICDCS 2011), Minneapolis, Minnesota, USA (June 2011)
13. Zhou, J., Li, Y., Adhikari, V.K., Zhang, Z.L.: Counting YouTube videos via random prefix sampling. In: Proc. of 2011 ACM SIGCOMM Internet Measurement Conference (IMC 2011), Berlin, Germany (November 2011)
14. Rao, A., Legout, A., Lim, Y., Towsley, D., Barakat, C., Dabbous, W.: Network characteristics of video streaming traffic. In: Proc. of 7th Internation COnference on emerging Networking EXperiments and Technologies (CoNEXT 2011), Tokyo, Japan (December 2011)
15. Plissonneau, L., Biersack, E., Juluri, P.: Analyzing the Impact of YouTube Delivery Policies on the User Experience. In: Proc. of 24th International Teletraffic Congress (ITC'24), Krakow, Poland (September 2012)
16. Alcock, S., Nelson, R.: Application flow control in YouTube video streams. ACM SIGCOMM Computer Communication Review 41(2) (April 2011)
17. Hoßfeld, T., Schatz, R., Biedermann, S., Platzer, A., Egger, S., Fiedler, M.: The Memory Effect and Its Implications on Web QoE Modeling. In: Proc. of 23rd International Teletraffic Congress (ITC'23), San Francisco, USA (September 2011)
18. Schatz, R., Egger, S., Hoßfeld, T.: Understanding Ungeduld - Quality of Experience Assessment and Modeling for Internet Applications. In: Proc. of 11th Würzburg Workshop on IP: Joint ITG and Euro-NF Workshop Visions of Future Generation Networks (EuroView 2011), Würzburg, Germany (August 2011)
19. Hoßfeld, T., Zinner, T., Schatz, R., Seufert, M., Tran-Gia, P.: Transport Protocol Influences on YouTube QoE. Technical Report 482, University of Würzburg (July 2011)
20. Hoßfeld, T., Liers, F., Schatz, R., Staehle, B., Staehle, D., Volkert, T., Wamser, F.: Quality of Experience Management for YouTube: Clouds, FoG and the AquareYoum. PIK - Praxis der Informationverarbeitung und -Kommunikation (PIK) 35(3) (August 2012)
21. Hoßfeld, T., Schatz, R., Seufert, M., Hirth, M., Zinner, T., Tran-Gia, P.: Quantification of YouTube QoE via Crowdsourcing. In: Proc. of IEEE International Workshop on Multimedia Quality of Experience - Modeling, Evaluation, and Directions (MQoE 2011), Dana Point, CA, USA (December 2011)
22. Hirth, M., Hoßfeld, T., Tran-Gia, P.: Anatomy of a Crowdsourcing Platform - Using the Example of Microworkers.com. In: Proc. of Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2011), Seoul, Korea (June 2011)

23. Hoßfeld, T.: Towards YouTube QoE via Crowdsourcing. Online Lecture within COST Qualinet (November 2011), `http://www3.informatik.uni-wuerzburg.de/papers/tutorials_16.zip`
24. Hoßfeld, T., Schatz, R., Egger, S., Fiedler, M., Masuch, K., Lorentzen, C.: Initial Delay vs. Interruptions: Between the Devil and the Deep Blue Sea. In: Proc. of 4th International Workshop on Quality of Mulitmedia Experience (QoMEX 2012), Yarra Valley, Australia (July 2012)
25. Hoßfeld, T., Schatz, R., Varela, M., Timmerer, C.: Challenges of QoE Management for Cloud Applications. IEEE Communications Magazine 50(4) (April 2012)
26. Fiedler, M., Hoßfeld, T., Tran-Gia, P.: A generic quantitative relationship between quality of experience and quality of service. IEEE Network - Special issue on Improving Quality of Experience for Network Services 24(2) (March 2010)
27. Tominaga, T., Hayashi, T., Okamoto, J., Takahashi, A.: Performance comparisons of subjective quality assessment methods for mobile video. In: Proc. of 2nd International Workshop on Quality of Mulitmedia Experience (QoMEX 2010), Trondheim, Norway (July 2010)
28. Staehle, B., Hirth, M., Pries, R., Wamser, F., Staehle, D.: YoMo: A YouTube Application Comfort Monitoring Tool. In: Proc. of EuroITV Workshop QoE for Multimedia Content Sharing (QoEMCS 2010), Tampere, Finland (June 2010)
29. Staehle, B., Hirth, M., Pries, R., Wamser, F., Staehle, D.: Aquarema in Action: Improving the YouTube QoE in Wireless Mesh Networks. In: Proc. of Baltic Congress on Future Internet Communications (BCFIC 2011), Riga, Latvia (February 2011)
30. Hoßfeld, T., Liers, F., Volkert, T., Schatz, R.: FoG and Clouds: Optimizing QoE for YouTube. In: Proc. of KuVS 5thGI/ITG KuVS Fachgespräch NG Service Delivery Platforms, Munich, Germany (October 2011)
31. Juluri, P., Plissonneau, L., Medhi, D.: Pytomo: a tool for analyzing playback quality of YouTube videos. In: Proc. of 23rd International Teletraffic Congress (ITC'23), San Francisco, USA (September 2011)
32. Rugel, S., Knoll, T., Eckert, M., Bauschert, T.: A network-based method for measurement of internet video streaming quality. In: Proc. of 1st European Teletraffic Seminar, Poznan, Poland (February 2011)
33. Schatz, R., Hoßfeld, T., Casas, P.: Passive YouTube QoE Monitoring for ISPs. In: Proc. of Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2012), Palermo, Italy (July 2012)
34. Lai, K., Baker, M.: Nettimer: A Tool for Measuring Bottleneck Link Bandwidth. In: Proc. of USENIX Symposium on Internet Technologies and Systems (USITS 2001), San Francisco, California, USA (March 2001)