Sacrificing Efficiency for Quality of Experience: YouTube's Redundant Traffic Behavior

Christian Sieber*, Poul Heegaard†, Tobias Hoßfeld‡, Wolfgang Kellerer*
*Chair of Communication Networks, Technical University of Munich, Germany
{c.sieber, wolfgang.kellerer}@tum.de
†Norwegian University of Science and Technology, Trondheim, Norway
{poul.heegaard}@item.ntnu.no
‡Universität Duisburg-Essen, Germany
{tobias.hossfeld}@uni-due.de

***Abstract*—Internet traffic reports show that YouTube is one of the major sources of data traffic world-wide. Furthermore, the data traffic shifts from mostly fixed landlines to cellular data connections where bandwidth is sparse and expensive. Previous studies revealed that YouTube uses a user-friendly HTTP Adaptive Streaming (HAS) strategy which sacrifices bandwidth efficiency to increase the average playback quality for the user. That way, it happens that the same video segment is transmitted in two or more quality levels, but only one can be shown to the user. We denote this as redundant traffic and this work is dedicated to understanding the influence factors on the amount of redundant traffic. This paper presents the results of a large-scale study with over 12,000 video views over a bottleneck link shaped to various bandwidths.We first evaluate the playback characteristics and show that YouTube's HAS algorithm linearly increases the average playback quality with the available bandwidth while at the same time video buffering is sub-linearly decreased. Furthermore, we identify video-dependent bandwidths which optimize the playback time on a quality level. Afterward, we show that this is achieved by discarding lower layer segments and therefore paid with redundant traffic of up to 40 %. We evaluate the overall efficiency of the system and show that YouTube is able to improve the average quality level by up to 0.7 quality levels by using this adaptation strategy. However, a penalty of 0.5 quality levels is paid for it due to the discarded data of the lower quality segments.**

I. Introduction

Video streaming services in the Internet gain more and more importance. The offered content and services for video on-demand (VoD) including live streaming exhibit a strong increase in popularity. By now, Internet traffic reports account VoD for the largest single type of consumer traffic in the Internet for landline and cellular access [1].

HTTP over TCP has become the de-facto standard for delivery of the VoD content. The HTTP protocol is firewall-friendly and easy to implement. Furthermore, large content delivery networks (CDN) are in place to distribute HTTP content globally and provide the content close to the users. At the beginning, VoD over HTTP was implemented by progressive download [2]. With progressive download, a HTTP server provides a single file with the content in a specific quality level. The player, e.g., the browser, starts to download the file and at the same time, or after an initial buffering phase, begins to play the video to the user. However, in case of insufficient network bandwidth, the video buffer depletes and the video stalls, which significantly decreases the Quality of Experience (QoE) of the end user [3].

ISBN 978-3-901882-83-8

Progressive download does not allow to adapt to current network conditions or to specific devices, e.g., phone or TV screen. By now, progressive download is being replaced by HTTP Adaptive Streaming (HAS). See [2] for a survey on HAS. HAS encodes the content into different quality levels, segments it into small chunks of a few seconds and makes it available over HTTP. A manifest file describing the content, e.g., video codec, number of quality levels and chunk location, is placed alongside the segments on the HTTP server. At first the client requests the manifest file and afterward downloads the content segments in a quality level chosen by the client. Dynamic Adaptive Streaming of HTTP (DASH) is an ISO standard describing the structure of the manifest file and is in use by some of the major content providers. However, the quality level adaptation logic, which dictates how the client should adapt the quality, is not part of the standard and is left to be decided by the implementation of the streaming client. Factors to consider for the adaptation are for example the viewing device, the available download bandwidth and the video bit-rates of the quality levels. As every client can implement their own adaption algorithm, the QoE of the user differs between content providers and streaming clients. The QoE of adaptive streaming is an active research topic [2]. QoE studies show that stalling must be avoided [3], [2] and that the quality switching rate should be minimized [4]. The average quality level is a major influence factor for the QoE in adaptive streaming [4].

In this paper we take a closer look at one of the major sources of video streaming traffic [5] in the Internet, YouTube. In a first pilot study [6] we have shown that YouTube's adaptation algorithm replaces previously downloaded lower quality segments. However, this introduces overhead, denoted as redundant traffic, as the lower quality segments are discarded. Hence, there is a trade-off between network efficiency and average playback quality that also affects the Quality of Experience of the user. Due to the limited scope of the study (four videos in total, 150 views), no reliable conclusions could

be drawn with respect to possible influence factors.

In this paper, we tackle two important questions. First, when does redundant traffic occur and how much redundant traffic is transferred over the network? Second, what is the overall efficiency of this approach, i.e., how much of the additional traffic can actually be used to improve the average quality level and how much has to be paid as penalty for replacing lower layer segments? To answer this, over 12,000 video views were recorded in our test-bed under different emulated network bottlenecks. We first describe the playback characteristics as observed for the different network conditions from the end user's point of view. Afterward we relate the playback characteristics to the recorded redundant traffic and show that the amount of redundant traffic can be estimated based on the playback characteristics. At the end, we evaluate the overall efficiency by comparing the quality gain due to the adaptation strategy with an estimation of the maximum achievable quality level based on the amount of total downloaded data in the session.

This paper is structured as follows. We first describe the background and related work in Section II. Here we also explain the behavior which leads to the redundant traffic in detail. In Section III we introduce the measurement methodology and explain the measurement set-up. In Section IV we discuss general playback characteristics observed in the study. In Section V we first evaluate the relationship between the observed playback characteristics and the overhead introduced by discarding lower quality segments. Afterward we evaluate the overall efficiency of the adaptation strategy considering the overhead. In Section VI we conclude the work and give future research directions.

II. Background & Related Work

In the following we first introduce HTTP adaptive streaming in general. Afterward we discuss the results of our previous study where we do a first characterization of the adaption behavior of YouTube. At the end we revisit the related research to this topic.

A. HTTP Adaptive Streaming

HAS enables client-driven adaptation of the quality level to device capabilities such as screen size, resolution and stereoscopic capabilities. Furthermore, by switching to different quality levels, the client can adapt the video bit-rate to the available bandwidth and therefor adapt to dynamically changing network conditions such as observed for example in cellular environments. HAS is implemented by encoding the content into multiple representations, e.g. different quality levels, segmenting it in small chunks (for YouTube: 10 s to 20 s, depending on the video) and making the segments available to the client through the HTTP protocol. The ISO standard MPEG-DASH is a widely accepted HAS standard adopted by YouTube. DASH defines an XML-based Media Presentation Presentation (MPD) file which describes the representations, e.g., average bit-rate or codec used, and gives the URL to the individual segments. At the beginning of a streaming session, the client requests the MPD file. Afterward the implementation on the client-side decides which chunks to request from which representation. As every content provider can implement their own adaption strategy, the experience for the user depends not only on the offered representations, but also on the ability of the adaption strategy to request in chunks in a user-friendly way. Evaluations show that the adaptation algorithm has a strong influence on the resulting playback behavior and therefore also on the perceived QoE of the user [7]. It has to be noted that YouTube also allows the end user to manually select a fixed quality level which deactivates HAS even if it results in frequent buffering. However, the default setting is the automated quality adaptation.

B. YouTube's Quality Adaptation Strategy

In our previous study [6] we describe the behavior of YouTube where the streaming client replaces previously downloaded chunks with higher quality ones. Figure 1 illustrates a request schedule for one of the experiment runs. At first, the YouTube player requests four segments of quality level 144p of a playback time up one minute and all four requests are made in the first 20 seconds of the experiment. At about 21 seconds into the experiment, the player revises its decision and replaces the segment containing the playback time 30 s to 45 s with a quality level of 240p. Afterward the player switches back to 144p and downloads playback time 60 s to 90 s in quality level 144p. Later in the experiment, the player switches up to a quality level of 480p by replacing 360p and 144p segments. The figure illustrates that some chunks of content are downloaded even more than twice.

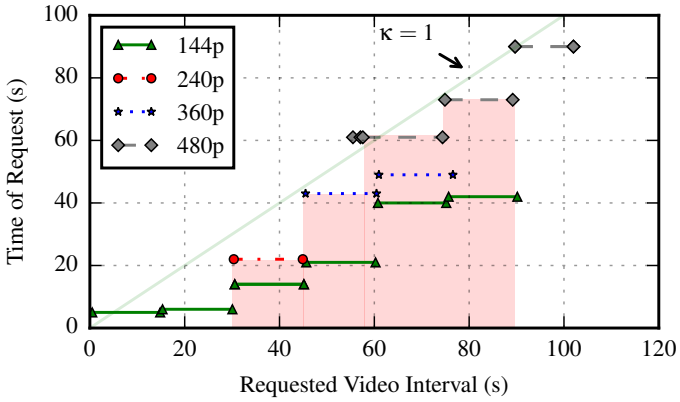


Figure 1. Example request schedule. The shaded areas indicate where lower quality segments are replaced by higher quality segments. For the first minute into the playback, 30 seconds of 144p are replaced with 15 seconds of 240p and 15 seconds of 360p to increase the average playback quality.

From the request schedule it becomes obvious that the adaption strategy tries to optimize the average playback quality shown to the user. Furthermore, by constantly downloading chunks, the algorithm prevents stalling of the underlying TCP connection. In a previous study [7] we show that by constantly utilizing the TCP connection a HAS strategy can keep the fair share of the network bandwidth provided by TCP adaptive behavior. Furthermore, adaptation strategies that underutilize

the TCP connection, e.g., if the playback buffer is full, have a distinct disadvantage compared to strategies which constantly utilize the TCP connection.

C. Related Work

In [8], Añorga et al. present a recent study about YouTube's HAS adaptation and review previous studies. Their findings show that YouTube uses a large playback buffer and therefore reacts only slowly to changing bandwidth conditions (13 s to 40 s). In [9], Yao et al. evaluate different sources of redundant traffic in video streaming services. They show how the iOS YouTube player requests overlapping segments to smoothen the playback. The amount of redundant traffic is not evaluated. In [10], Lui et al. evaluate the difference between Android and iOS-based YouTube media streaming. They conclude that buffering is not dependent on the playback time, but on the amount of data buffered. Furthermore, they quantify a redundant traffic of 15 % and account it for re-downloading the beginning of the video. Mansy et al. [11] analyze the streaming behavior of the three streaming providers including YouTube in terms of their playback characteristics, redundant traffic and bandwidth utilization. The authors observe that YouTube aggressively discards segments of lower quality levels to download higher bit-rate segments when the bandwidth increases. In the study, one video is evaluated in a wireless scenario with varying bandwidths (every 2 minutes a new link bandwidth is set). For this scenario, the authors conclude a percentage of redundant traffic of 16 %. The study also shows that other content providers deploy similar adaptation strategies. Nam et al. [12] show that in a mobile scenario more than 35 % of transferred data by YouTube is redundant. The authors account frequent termination of TCP connections and discarded on-fly packets as the case of the redundant traffic. Rao et al. [13] and Ito et al. [14] model the traffic patterns produced by a YouTube streaming session. Their findings suggest that the implementation of the adaptation strategy and therefor the resulting behavior varies with the type of the viewing device. Alcock et al. [15] describe the initial burst phase deployed by YouTube. In the initial burst phase, 32 s of playback time are sent to the client a fast as possible. We also observe this initial burst phase and account it for a source for redundant traffic, as a low quality initial burst phase is in some cases later replaced by higher quality segments.

This work extends the state-of-the art and presents the first large-scale study which statistically quantifies the amount of redundant traffic. Further, we relate redundant traffic to QoE influence factors and are able to quantify the QoE loss by downloading redundant traffic instead of higher quality levels.

III. Methodology

In the following we first discuss the experimental set-up used to collect the results. Afterward we introduce the notation and metrics evaluated in this work. At the end we give a description of how we selected the content and which characteristics the selected content exhibits.

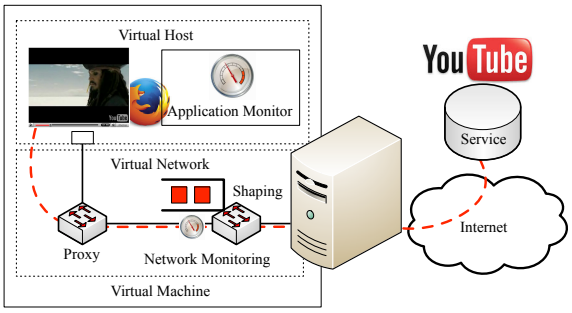


Figure 2. Experimental set-up based on a virtual machine (VM) and virtual network with browser-based and in-network-based monitoring.

A. Experiment Set-up

Figure 2 illustrates the experiment set-up. The set-up consists of a virtual machine with *Xubuntu 14.04 64-bit* running a browser (*Firefox 21*) with the YouTube player, an HTTPS proxy inside the virtual machine and a virtual network which limits the available bandwidth. The set-up is connected to the Internet through a lightly utilized lab network and through the university's Internet connection. Internal traffic reports and monitoring of experiment download speed rule out external bottlenecks. Since the beginning of 2015, YouTube uses the encrypted HTTPS protocol for video delivery. In order to still being able to decrypt the traffic in the network, we inject a custom certificate into the browser and mark it as trustworthy for the domains used by YouTube. Furthermore, we use a customized version of *mitmproxy* [16] in order to intercept, capture and decrypt the YouTube traffic. The proxy was customized to quickly forward all incoming traffic without buffering and tested to rule out any performance issue as influence factor on the adaptation behavior.

In order to capture the player state and the HTTP requests in the network, we use browser-based and network-based monitoring. We implemented browser-based monitoring by embedding the YouTube video into a custom web-page and use an extension for the browser [17] to access the API of the player to monitor buffering events and quality switches. For network monitoring, we use the described proxy to capture and decrypt the HTTPS requests. In order to translate between the byte range requests and playtime seconds, we download the videos [18] in all evaluated quality levels and afterward decode the downloaded mp4 containers [19].

Before every experiment run, the virtual environment is reset to a default state. During the playback, the status of the experiment is constantly monitored and if the video was not played out until the end, the experiment run is discarded. A detailed description including the whole experimental set-up is available together with the raw and summarized traces online [20]. We encourage others to play with the data and do their own evaluations. At total, the traces and summarized results for 12,075 runs are available online (35 videos $\times$ 27 bandwidth values $\times$ 15 replications).

Table I
KEY VARIABLES AND NOTATIONS USED IN THE PAPER.

notation	meaning
B	downloaded and played bytes
B_T	total downloaded bytes
ρ	Redundant Traffic Ratio (RTR)
Q	number of quality levels
$\mathbb{Q}$	set of quality levels; $\mathbb{Q} = \{q_0; q_{Q-1}\}$
q	quality level, $q \in \mathbb{Q}$
V	number of videos
v	video index
R	number of replicated downloads (of each video v)
r	replication index
F	number of bandwidth levels
f	bandwidth level
f^*	rescaled bandwidth level
n_v	number of segments in video v
τ_v	(fixed) segment play time of video v
x_{ij}	quality index indicator; 1 if segment i is downloaded at quality level j, 0 otherwise
s_{ij}	size of segment i for video quality level j
J	average quality level
J_i^+	maximum quality level downloaded of segment i; $J_i^+ = \{\max j \| x_{ij} > 0\}$
J_i^-	minimum quality level downloaded of segment i; $J_i^- = \{\min j \| x_{ij} > 0\}$
b_i	number of buffer event in segment i
ψ	buffering rate for bandwidth f
t_{fv}	buffering time for video v and bandwidth f
T_{fq}	average relative time/probability on quality level q in a video sequence with bandwidth f
γ	average bit-rate
ϕ_i	1 if quality level has switched from segment $i-1$ to i, 0 otherwise;
w	average number of switches
ϵ	overall efficiency
κ	buffering ratio

B. Metrics

The notation used in the paper is summarized in Table I. In the following we define the key performance metrics as used in our analysis. Efficiency ϵ is defined in Chapter V.

The average bit-rate per quality level q of video v

$$\gamma_{qv} = \frac{1}{n_v \tau} \sum_{i=1}^{n_v} \{s_{iq}\}_v \tag{1}$$

Total downloaded bytes in video sequence v:

$$B_T = \sum_{i=1}^{n_v} \sum_{j=0}^{Q-1} x_{ij} s_{ij} \tag{2}$$

Total *played* bytes in a session under the assumption that always the maximum quality level downloaded J^+ is played:

$$B = \sum_{i=1}^{n_v} s_{iJ_i^+} \tag{3}$$

Each bandwidth level f, replication r and video v has a unique B_{frv} and a corresponding average quality level J_{frv}. The average quality level with played bytes B is denoted J_B and is the average over the video sequences with $B_{frv} = B$.

The average played quality level for bandwidth f:

$$J_f = \frac{1}{RV} \sum_{r=1}^{R} \sum_{v=1}^{V} \frac{1}{n_v} \sum_{i=1}^{n_v} \{J_i^+\}_{frv} \tag{4}$$

The average buffering event rate for bandwidth f:

$$\psi_f = \frac{1}{\tau RV} \sum_{r=1}^{R} \sum_{v=1}^{V} \frac{1}{n_v} \sum_{i=1}^{n_v} \{b_i\}_{frv} \tag{5}$$

The average quality switching rate for bandwidth f:

$$w_f = \frac{1}{\tau RV} \sum_{r=1}^{R} \sum_{v=1}^{V} \frac{1}{n_v} \sum_{i=1}^{n_v} \{\phi_i\}_{frv} \tag{6}$$

The estimated probability of playtime on quality level q in a video sequence for bandwidth f:

$$T_{fq} = \frac{1}{RV} \sum_{r=1}^{R} \sum_{v=1}^{V} \frac{1}{n_v} \sum_{i=1}^{n_v} \{x_{iq}\}_{frv} \tag{7}$$

The buffering ratio κ (8) is the ratio between the experiment run-time and the duration of the video:

$$\kappa_{frv} = \frac{n_v \tau_v + t_{frv}}{n_v \tau_v} \tag{8}$$

Buffering ratio $\kappa = 1$ means that no buffering happened during the experiment. κ for a specific bandwidth f is defined as $\kappa_f = \frac{1}{RV} \sum_{r=1}^{R} \sum_{v=1}^{V} \kappa_{frv}$.

C. Redundant Traffic Ratio (RTR)

The redundant traffic is defined as the ratio between the downloaded bytes that are discarded ($B_T - B$) and the bytes that are played (B). This gives the overhead of the playback with the data volume of only the played out segments as reference. The Redundant Traffic Ratio (RTR) for one single download is then

$$\rho = \frac{B_T - B}{B} \tag{9}$$

For download at a specific bandwidth factor f, the index is added to B_T in Eq. (2) and to B in Eq (3) and the redundant traffic ratio ρ is updated accordingly.

D. Content

In order to represent the variety of videos uploaded to YouTube, we use the YouTube API provided by Google to automatically select suitable videos. We define 5 categories "*minecraft*", "*music*", "*funny cats*", "*gopro*", "*game*" by using the category name as search query string. Furthermore, we filter the query results by the following criteria. The selection of videos considers the following aspects: 1) embeddable, i.e., use in HTML iframe allowed, 2) syndication is allowed, 3) available in high resolutions and 4) videos which were published between 1 and 9 month ago. The query result is sorted by popularity, i.e., view count, and from the result we select videos with a duration of 1, 2, ... , 10 minutes with an allowed deviation of 5 seconds. In total, 35 different videos were accessible during the whole experiment time and are

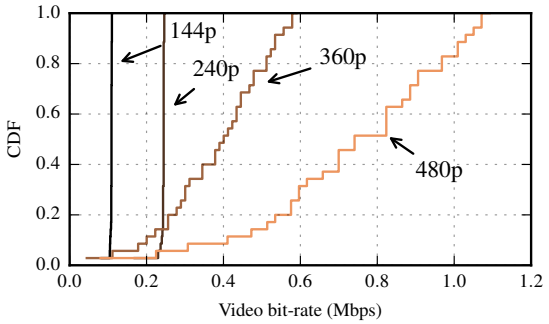


Figure 3. CDF of the quality levels of the selected video sequences. Means: 144p: 0.11 Mbit/s, 240p: 0.24 Mbit/s, 360p: 0.37 Mbit/s, 480p: 0.72 Mbit/s.

included in the evaluation. Average duration of selected videos is 5.3 minutes.

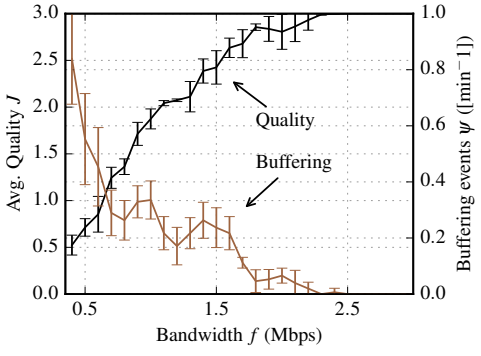
Figure 3 shows the cumulative distribution function (CDF) of the bit-rates of the four quality levels, $\mathbb{Q} = \{144p, 240p, 360p, 480p\}$ for the selected videos. Note that the client player uses some decision logic to select a subset of the available resolutions on YouTube's servers based on the type of device, e.g. based on the screen size. In our environment this was the subset $\mathbb{Q}$. The average bit-rate for each quality level is calculated by Eq. 1, then γ_{144p}=0.11 Mbit/s, γ_{240p}=0.24 Mbit/s, γ_{360p}=0.37 Mbit/s, and γ_{480p}=0.72 Mbit/s. Note that although 360p has a higher number of pixels than 240p, approximately 18 % of the videos in quality level 360p are encoded with a lower average bit-rate then the average bit-rate for quality level 240p. YouTubes encoding of (cover art) music videos leads to higher data volume for lower resolutions than for higher resolutions. The reason for this is not clear as it depends on the YouTube internal encoding of those videos. Please note that we nevertheless include those videos in the first part of the evaluation as they are part of the content mix observed on the platform. For the evaluation of the efficiency, we exclude those videos.

IV. Video Playout and Adaptation Characteristics

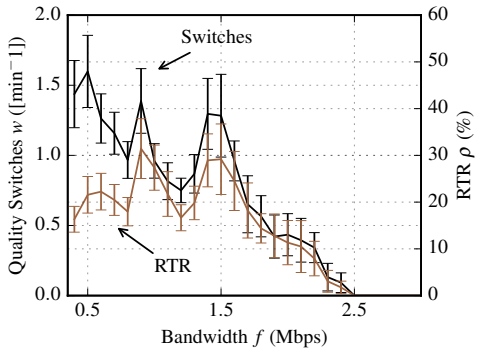
This section presents key characteristics of the adaptation observed in our measurements for different bandwidths f. This includes the average quality level J_f, quality switching rate w_f, buffering rate ψ_f, and the probability of a playback T_{fq} at quality q in a session. If not otherwise stated, error bars in the figures depict the 95 % confidence interval.

Figures 4 (a) and (b) illustrate how the average quality J_f, the buffering rate ψ_f, the quality switching rate w_f and the redundant traffic ratio ρ_f develops for increasing bandwidth f for video v = CbhnuRhbC, which shows scenes from a video game with picture-in-picture commentary. The average bit-rate for the four quality levels are γ ={0.11 Mbps, 0.25 Mbps, 0.43 Mbps, 0.84 Mbps}.

Figure 4(a) shows the average quality J and the buffering rate ψ. The buffering rate decrease from 0.8 [min^{-1}] at 0.8 Mbps to 0 at 2.5 Mbps. The buffering rate has two peaks for ρ. The average quality level increases approximately linear from 0.5 at 0.8 Mbps to the maximum of 3 at 2.5 Mbps.



(a) Buffering rate and average quality



(b) Quality switching rate and RTR

Figure 4. Average playback quality J_f, buffering rate ψ_f [min^{-1}], quality switch rate w_f [min^{-1}] and RTR ρ_f for video v = CbhnuRhbX5I. 95 % confidence intervals over 15 runs are indicated.

In Figure 4(b) we observe that for bandwidth $f \leq 1.6$ Mbps, the RTR is approximately $\rho = 20$ %, except for two peaks of 30 % at both 0.9 Mbps and 1.5 Mbps. With $f >$1.6 Mbps, the RTR decreases linearly until it reaches zero at about 2.5 Mbps. The quality switching rate has a linear decreasing trend from approximately 1.5 [min^{-1}] down to 0 [min^{-1}] at 2.5 Mbps. The same two peaks at 0.9 Mbps and 1.5 Mbps as for the RTR ρ_f are also observed for w_r. Next we summarize the results over all videos per bandwidth f.

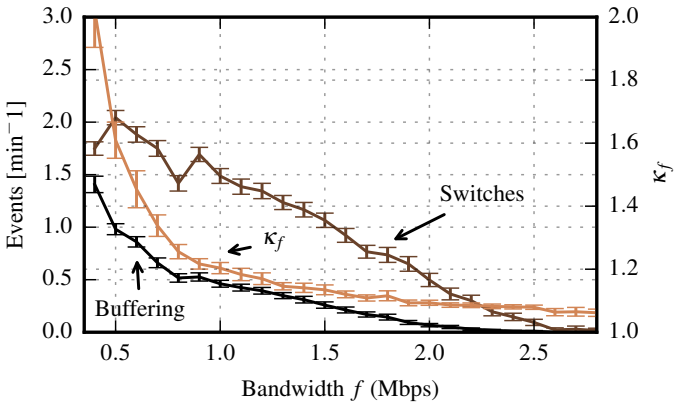


Figure 5. Buffering events [min^{-1}], switching events [min^{-1}] and buffering ratio κ_f. κ_f and buffering rate decreases non-linear. Switching rate decreases linear.

Figure 5 shows the buffering rate, the switching rate and the

average *buffering ratio* κ for the different bandwidth values f and averaged over all videos. The buffering ratio κ (8) is the ratio between the experiment run-time and the duration of the video. $\kappa = 1$ means that no buffering happened during the experiment. In our set-up, the browser start-up time is included in the experiment time. The minimum value κ can reach is about 1.05, depending on the video. The figures show that the buffering rate and average buffering ratio decreases non-linear with increasing f, while the corresponding decrease in quality switching rate is approximately linear. At the lowest evaluated bandwidth 0.4 Mbps, we observe an average buffering rate of 1.4 [min^{-1}] and an average of buffering time close to two times the duration of the video. The quality switching rate is on average between 1.75 [min^{-1}] and 2.0 [min^{-1}] for 0.4 Mbps to 0.5 Mbps. At about 2.6 Mbps, the three metrics reach their minimum of zero for the buffering and switching events and about 1.07 for the ratio between experiment and video duration. From the figure we conclude that for a bandwidth of 2.6 Mbps all videos in our result set are, on average, played back without buffering events and quality switches. Furthermore we see that switching events are more frequent than buffering events and decrease slower for increasing f.

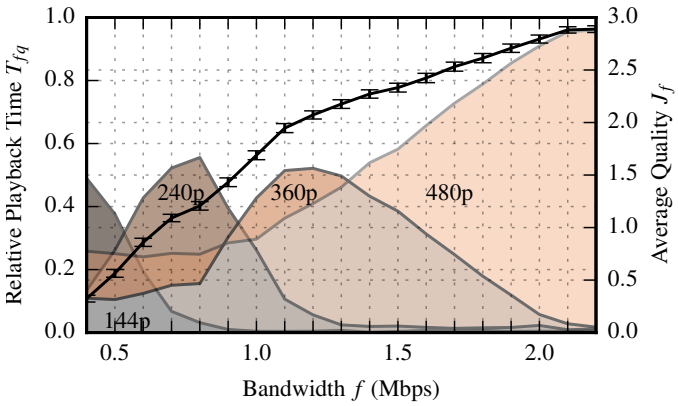


Figure 6. Average quality level and probability of playout time on quality level T_{fq} ($\sum_{q \in \Omega_I} T_{fq} = 1$).

In Figure 6, the average quality level is depicted on the axis to the right. The axis on the bottom gives the bottleneck bandwidth in the range of $f = (0.4, 2.2)$ Mbps. The shaded areas indicate the average fraction of time spent on each of the quality levels for a specific bandwidth, i.e., the relative playback time (scale on the left axis). Two key observations can be made from the figure. First, for each quality level there is a distinct bandwidth range where one of the four different quality levels dominates. This is illustrated by T_{fq} for $q = 0, 1, 2, 3$ which have their maximum at different bandwidths f. Second, even at a low data rate as $f = 0.5$, the $T_{0.5,4} = 0.2$, which means that in 20 % of the time the highest quality level is viewed to the user. This can be explained by the fact that some of the videos, especially music videos with static cover, have a low average bit-rate for the higher quality levels and therefor can be selected in the highest quality level even if the available download-rate is low. The figures also illustrate for which range of f the different qualities dominate and when we switch from one level to the next higher or lower quality level. Based on Figure 6 we cannot identify a clear relationship between the bandwidth and the probability of selecting a specific quality level as the regions are heavily overlapping. This is due to the high variance in video bit-rates as shown in Figure 3.

To align the probabilities of the four quality levels, T_{vq}, we scale the bandwidth f with a quality and video dependent bandwidth factor $f^*_{vq} = f/\gamma_{vq}$. In Figure 7 we plot the relative playback time T_{vq} for each experiment run and for all four quality levels q using the rescaled bandwidth f^*_{vq}. The error bars indicate the confidence interval of 95 % for each of the bins. From this we observe that $q_1 = 240$p, $q_2 = 360$p and $q_3 = 480$p reach their maximum T_{fvq} at approximately $f^*_q = 3$. This is indicated by the vertical (red) line. For 240p and 360p the maximums are close to 60 %, while for 480p the maximum is 100 %. For 144p, the maximum of about 55 % is reached at $f^*_1 = 3.8$. There are no samples for $f^*_1 < 3.8$ available, which corresponds approximately to a bandwidth of 0.4 Mbps. Hence, the maximum is reached for all quality levels at $f^* = 3$, independent of the quality level q. It can be read as; when three times the average bit-rate of a certain video v of a certain quality level q is available, this quality level dominates. If we have more, we switch to a higher level, if it exists. Note that we compare here the link bandwidth with the raw video bit-rate, without any overheads (e.g. IP, HTTP, TCP or video container overhead) and without the audio stream. Therefor, the absolute value where T_{vq} reaches its maximum may vary depending on the scenario. But, the results show that there is a (narrow) range of bandwidths for each video where each quality level reaches a maximum probability of being played out to the user. This suggests, that the adaptation algorithm of YouTube uses the average bit-rate of a quality level of a video and compares it with the available network bandwidth to determine which quality level to select next. The results also reveal that although the network bandwidth remains constant quality switches occur and the video is watched in different quality levels. In the next chapter we discuss how playback characteristics affect on the redundant traffic.

V. Evaluation of Adaptation Efficiency

In the previous section we characterized the playback behavior from the perspective of the user (average quality level, buffering and switching events) and discussed how the available bandwidth influences the playback behavior. Next, we put the evaluated playback characteristics in perspective to the amount of redundant traffic downloaded in the background, unnoticed by the viewer, and subsequently discuss the effectiveness of this adaptation approach. We do this by introducing a metric which summarizes the quality gain due to the adaptation on the one side and the penalty introduced by the redundant traffic on the other side.

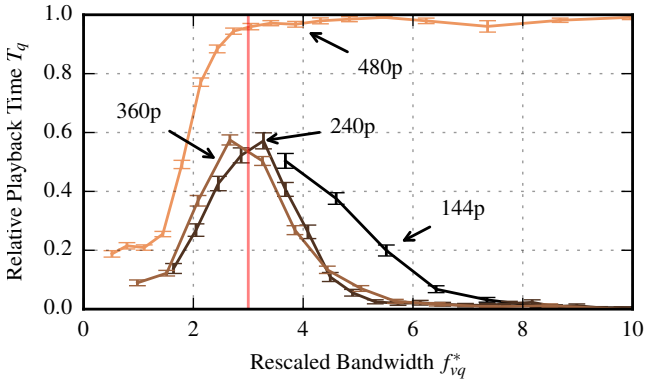
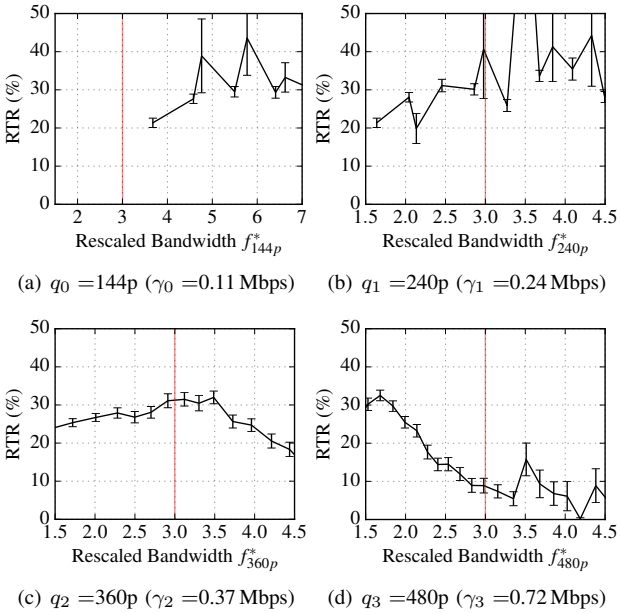


Figure 7. The probability of each playback quality level $T_{f^*_{vq}}$ as a function of the rescaled bandwidth f^*_{vq}. The vertical line at $f^*_{vq} = 3$ indicates where we find the peak for all quality levels.

A. Relationship of Playback Characteristics to Redundancy

In Figure 8 we take a closer look at the peak of relative playback time as shown in Figure 7 from the perspective of the RTR based on the rescaled bandwidth f^*_{vq}. The average bit-rate of each of the quality levels q is given as a reference and can be used to get an estimate of the bandwidth f based on f^*_q ($f = \gamma_q \times f^*_q$). However, this neglects the standard deviation of the bit-rates of the videos. The figure shows that for 240p and 360p, high T_q translates to about 30 % of RTR. For 144p, a RTR of 20 % for 3.6 of rescaled bandwidth is observed. The highest quality level 480p exhibits a RTR of 10 % at the point where it reaches close to 100 % of the playback time. We conclude from the figure that the peak of the estimated probability of playtime on a quality level q, T_{fq}, does not translate to a stable, i.e., sequential, quality selection behavior. On the contrary, the RTR for 240p and 360p show that the player downloaded up to 40 % of data more than it showed to the user.

Figure 9 depicts the switching and buffering rate for increasing values of RTR. K-means clustering is used to generate bins of RTR in the figure. The error bars indicate the 95 % confidence interval for a cluster. The shaded area in the background shows the probability density function, $g(\rho)$ for the observed values of RTR. The $g(\rho)$ shows that approximately 25 % of the experiment runs in the result set do not exhibit any or a minor percentage of redundant traffic and the second highest density of samples can be observed between a RTR of 15 % and 50 %. The figure illustrates that there is a close relationship between the switching/buffering rate and the amount of redundant traffic. While the buffering rate increases only slowly for increasing RTR, the switching rate increase is steeper. For 20 % of RTR the buffering rate is approximately one buffering event per two minutes, while switching rate is up to 1.2 switches per one minute. By implication, this shows that buffering events quickly translate to a high amount of redundant traffic. Buffering events are a sign of sudden drop in bandwidth (or equivalent: a sudden increase in the video bit-rate) or an effect of poor decision



(a) q_0 =144p (γ_0 =0.11 Mbps) (b) q_1 =240p (γ_1 =0.24 Mbps)

(c) q_2 =360p (γ_2 =0.37 Mbps) (d) q_3 =480p (γ_3 =0.72 Mbps)

Figure 8. RTR for the rescaled bandwidth f^* relative to the four quality levels. Note that the shown range of f^* is wider for 144p compared to the other quality levels.

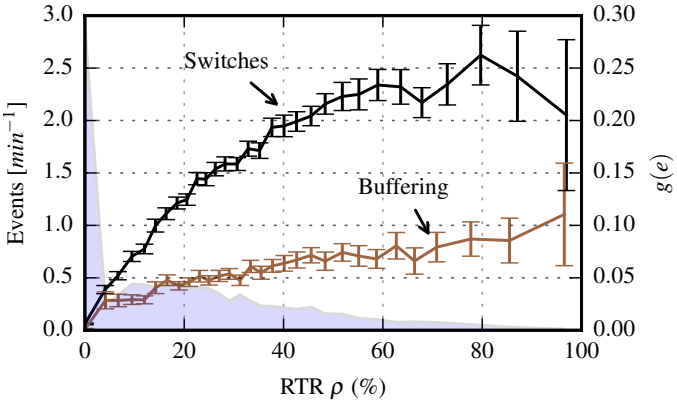
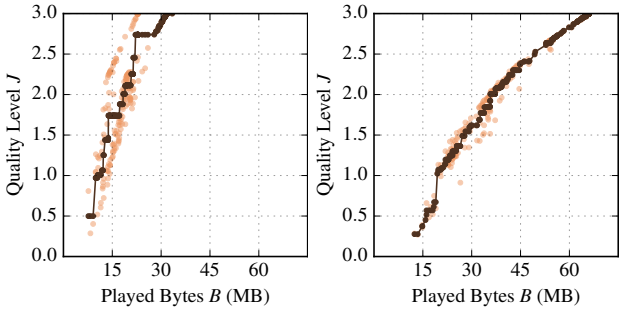


Figure 9. Switching and buffering events for increasing RTR. The shaded area illustrates the probability density function of the RTR values in the collected result set.

by the adaptation logic. Hence, it can be concluded that higher values of redundant traffic can be observed when the playout buffer is often depleted. Furthermore, from the (approximately) strict monotonic increase follows that it is possible to estimate the RTR of a viewing session based on the playback characteristics.

B. Efficiency of Adaptation Strategy

Next, we discuss the efficiency of the observed adaptation strategy. In particular we want to answer the following questions: How much of the additionally downloaded data was actually used to improve the average playback quality? And how much average quality was lost, compared to an optimum where the segments are downloaded without overlaps. For calculation of the efficiency we exclude videos where higher quality levels have a lower bit-rate than lower quality levels.



(a) v_1 = _pjBVUyq530 (b) v_2 = vbLLqaa9ksw

Figure 10. Approximated function θ for two of the videos.

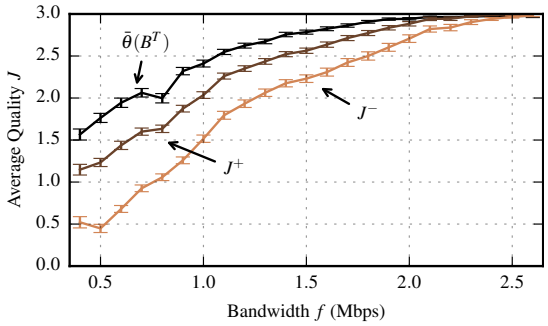


Figure 11. The worst case average quality J_f^-, the played average quality J_f^+ and the estimated optimum quality $\theta_f(B^T)$ for increasing bandwidth f.

The translation between average playback quality of a certain downloaded volume of bytes z is estimated by a function $\theta(z)$, which is determined by regression of the observation of the quality levels as a function of the bandwidth f. We apply isotonic regression [21] to fit the played bytes B to the average quality level. Isotonic regression approximates monotonic functions and does not assume a specific shape of the underlying function. The translation between the bytes B of the *viewed* quality level J^+ and the quality level J^+ itself is (mostly) monotonic, therefor isotonic regression is applicable to the problem at hand.

Figure 10 illustrates the isotonic regression result for two of the videos v_1 and v_2 used in the result set. The red dots depict the samples collected for the video, (B, J^+). The connected green dots show the approximated function $\theta(y)$, $y \in (0, B^+)$ (B^+ is the maximum observed B). Two observations can be made from the figure. First, θ_{v_2} is more flat than θ_{v_1}. Second, v_2 shows a larger deviation for the bytes required for a specific average quality level. The difference in slope is a result of the different bit-rates for quality level $q_3 = 480\text{p}$ for the two videos. $q_3 = 480\text{p}$ of video v_1 has an average bit-rate of $\beta_{v_1} = 0.49\,\text{Mbps}$, while for v_2 the average bit-rate is $\beta_{v_2} = 0.78\,\text{Mbps}$. The deviation for the same quality level indicates that the standard deviation for the video bit-rate for video v_1 is higher than for v_2. Next, we use θ to estimate the optimum average quality level for a given B^T and compare it to the observed average playback quality and to the worst case quality level J^- where no lower quality segments were replaced by segments with a higher quality.

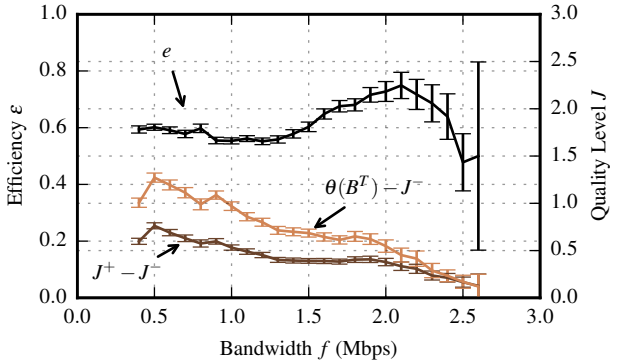


Figure 12. Efficiency ϵ_f, difference between played and worst case quality level $J_f^+ - J_f^-$ and difference between estimated optimum and worst case quality $\theta_f(B^T) - J_f^-$ over bandwidth f.

Figure 11 depicts the worst case quality level J_f^-, the played quality level J_f^+ and the estimated optimum quality level based on the total downloaded amount of data $\theta(B^T)$ for a bandwidth factor up to $f = 2.6\,\text{Mbps}$. It can be seen in the figure, that for low bandwidths, the absolute difference between three quality levels is larger than for higher bandwidth values. For example for 0.5 Mbps, the quality level increases due to the adaption strategy by 0.7 quality levels. However, based on the amount of downloaded data, the redundant traffic introduced a penalty of 0.5 quality levels. For higher bandwidth values, the absolute difference between the three quality levels J_f^-, J_f^+ and $\theta(B^T)$ becomes less, but the ratio between them stays roughly the same. For a bandwidth of 2.6 Mbps, no difference can be observed.

Next, we define a metric for the efficiency of the adaptation strategy in respect to the amount of downloaded and played data. In the best case, an adaption strategy allows the player to utilize all the bytes downloaded. However, this is only possible if $\theta(B^T) = J_f^+ = J_f^-$, as each difference between J_f^+ and J_f^- introduces redundant traffic and therefor increases also the difference between estimated optimum $\theta(B^T)$ and J_f^+.

We define the *efficiency* ϵ of the algorithm as the ratio between the measured quality gain $(J^+ - J^-)$ and the estimated maximal quality gain $\theta(B^T) - J^-$ (see Eq (10)).

$$\epsilon = \frac{J^+ - J^-}{\theta(B^T) - J^-} \tag{10}$$

If $\epsilon = 1$, the algorithm was able to utilize all the additionally downloaded data to improve the average quality level. If $\epsilon = 0$, the algorithm requested additional segments, but was not able to improve the average quality level by doing so.

Figure 12 depicts the efficiency ϵ_f, the difference between played and worst case quality level $J_f^+ - J_f^-$ and difference between estimated optimum and worst case quality $\theta_f(B^T) - J_f^-$ over bandwidth f. The figure shows that for low bandwidths ($f < 0.8\,\text{Mbps}$), the adaptation strategy is able to utilize 60 % of the additionally downloaded data to increase the average quality level J. For bandwidth from 0.8 Mbps to 1.5 Mbps, the efficiency drops to 55 %. For bandwidths from 1.5 Mbps up to

2 Mbps, the efficiency increases up to 65 %. For bandwidths larger than 2 Mbps, the width of the confidence interval do not allow a reliable conclusion. For overcapacity bandwidth f, the ϵ_f looks "noisy" because there are no adaptation, and hence not redundancy, and therefor the $\theta_f(B^T)$, J_f^-, and J_f^+ are (almost) equal. To summarize the figure, the algorithm is on average able to utilize 60 % of the additionally downloaded data to improve the average quality level compared to J^-, where no replacement of lower quality segments takes place. The left over 40 % are the penalty the algorithm has to pay for discarding previously downloaded segments.

VI. Conclusion & Outlook

The traffic share of video streaming in the Internet continues to increase. Major content providers such as YouTube provide a global infrastructure to serve commercial and user-generated contents to every end-user's device. YouTube alone accounts for about 15 % of the total downstream Internet traffic in the US. Furthermore, reports show that the video traffic also increases for cellular access where available data-rate is scarce. Therefor it is important to understand how YouTube adapts to the available bandwidth and how efficiently it uses the available network resources. Previous studies reveal that the adaption strategy used by the YouTube client tries to optimize the average playback quality by replacing previously downloaded lower quality segments with higher quality segments. This increases the average quality shown to user. However, it decreases the efficiency in respect to the used network resources. In this paper we present the results of a large-scale study with more than 12,000 video views of different contents while the downstream traffic was shaped to emulate a bottleneck link with a certain bandwidth. First we show how YouTube adapts to the available bandwidth from the perspective of the user in terms of quality switching, buffering events and playback quality. Higher available bandwidth increases almost linearly the average playback quality, while buffering ratio and buffering event rate are decreasing sub-linearly. The switching frequency is decreasing almost linearly. The results show that for every video and quality level there is a specific network bandwidth which maximizes the time spent on the specific quality level. Thus, this network bandwidth is related to the video bit-rate of that quality level. However, that specific bandwidth does not force the player to select a specific quality level with a probability larger than 0.6 for any quality level below the maximum, except for the trivial case of bandwidth overprovisioning of more than 300 % of the video bit-rate. In any other case, quality switching always occurs.

In the second part of the study we discuss the efficiency of the adaption strategy from a networking view point. The results show that by replacing lower quality segments, the YouTube player is able to increase significantly the average playback quality by up to 0.7 quality levels. However, 30 % redundant data has to be downloaded for this. Based on the amount of redundant traffic, we estimate the optimal, i.e. highest, average quality level which could be achieved by downloading the same total amount of data but avoiding redundant traffic. The results show that without redundancy the amount of downloaded data could have been used to increase the average quality level by up to 1.3 quality levels. Therefor, about a half quality level is unnecessarily downloaded and discarded due to the redundancy.

In the future, we plan to include highly varying bandwidth conditions, e.g. as found in cellular access. Furthermore, the maximum achievable playback quality for a certain data volume can be formulated as an optimization problem instead of regression based on historic data. This can give insight into an upper bound for potential improvements to YouTube's adaptation algorithm.

References

[1] "Cisco visual networking index: Foreecast 2014 - 2019." [Online]. Available: http://goo.gl/0iqjo9

[2] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-Gia, "A survey on quality of experience of http adaptive streaming," *Communications Surveys Tutorials, IEEE*, 2014.

[3] T. Hoßfeld, R. Schatz, E. Biersack, and L. Plissonneau, "Internet video delivery in youtube: from traffic measurements to quality of experience," in *Data Traffic Monitoring and Analysis*. Springer, 2013, pp. 264–301.

[4] T. Hoßfeld, M. Seufert, C. Sieber, T. Zinner, and P. Tran-Gia, "Identifying qoe optimal adaptation of http adaptive streaming based on subjective studies," *Computer Networks*, vol. 81, pp. 320–332, 2015.

[5] Sandvine, "Global Internet Phenomen Report 1H 2014," 2014.

[6] C. Sieber, A. Blenk, M. Hinteregger, and W. Kellerer, "The cost of aggressive HTTP adaptive streaming: Quantifying YouTube's redundant traffic," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, may 2015, pp. 1261–1267.

[7] C. Sieber, T. Hoßfeld, T. Zinner, P. Tran-Gia, and C. Timmerer, "Implementation and User-centric Comparison of a Novel Adaptation Logic for DASH with SVC," in *IFIP/IEEE International Workshop on Quality of Experience Centric Management (QCMan)*, Ghent, Belgium, May 2013.

[8] J. Añorga, S. Arrizabalaga, B. Sedano, M. Alonso-arce, and J. Mendizabal, "YouTube's DASH implementation analysis," in *19th International Conference on Circuits, Systems, Communications and Computers (CSCC)*, 2015, pp. 61–66.

[9] L. Yao, W. Qi, G. Lei, S. Bo, C. Songqing, and L. Yingjie, "Investigating Redundant Internet Video Streaming Traffic on iOS Devices: Causes and Solutions," *Multimedia, IEEE Transactions on*, vol. 16, no. 2, 2014.

[10] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen, "A comparative study of android and ios for accessing internet streaming services," in *Passive and Active Measurement*. Springer, 2013, pp. 104–114.

[11] A. Mansy, M. Ammar, J. Chandrashekar, and A. Sheth, "Characterizing Client Behavior of Commercial Mobile Video Streaming Services," in *Proceedings of Workshop on Mobile Video Delivery, MoViD'14*, 2014.

[12] H. Nam, B. H. Kim, D. Calin, and H. G. Schulzrinne, "Mobile video is inefficient: A traffic analysis," 2013.

[13] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*. ACM, 2011.

[14] M. Ito, R. Antonello, D. Sadok, and S. Fernandes, "Network level characterization of adaptive streaming over http applications," in *IEEE Symposium on Computers and Communication (ISCC)*, June 2014.

[15] S. Alcock and R. Nelson, "Application flow control in youtube video streams," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, Apr. 2011.

[16] "mitmproxy." [Online]. Available: https://mitmproxy.org/

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

[18] "youtube-dl." [Online]. Available: https://rg3.github.io/youtube-dl/

[19] "ffprobe." [Online]. Available: https://ffmpeg.org/ffprobe.html

[20] "Traces and programs used in this paper." [Online]. Available: http://git.io/vRSSW/

[21] R. E. Barlow, D. J. Bartholomew, J. Bremner, and H. D. Brunk, *Statistical inference under order restrictions: the theory and application of isotonic regression*. Wiley New York, 1972.