The Cost of Aggressive HTTP Adaptive Streaming: Quantifying YouTube's Redundant Traffic

Christian Sieber, Andreas Blenk, Max Hinteregger, Wolfgang Kellerer
Chair of Communication Networks
Department of Electrical and Computer Engineering
Technische Universität München, Germany
Email: {c.sieber,andreas.blenk,wolfgang.kellerer}@tum.de

***Abstract*—Video content and, in particular, YouTube's content account for the largest amount of today's Internet traffic. However, little is known about the behavior of video streaming services for different kinds of network environments and under varying network conditions. Due to network operators' lack of knowledge about the transmitted content, network resources may not be optimally used in general. Thus, we propose a dyadic measurement system composed of application, i.e., client-based and network-based monitoring for YouTube's video traffic. Using our proposed monitoring methodology, we analyze the behavior of YouTube's HTTP-based adaptive video streaming mechanisms. In detail, we quantify via experimental measurements on real network traffic YouTube's behavior for different videos under static and varying network conditions. Our measurement results show that in case of varying network conditions, YouTube demands different video qualities in parallel in order to adapt to the network situation. However, this behavior can result in up to 33 % of redundant network traffic, i.e., downloaded video content of different quality levels for the same play time. Due to our findings, network operators should try to optimize the allocation of network resources for video content in a way that avoids varying network conditions, resulting in less waste of network resources.**

I. INTRODUCTION

Today's Internet traffic is dominated by video content. Among all transport protocols used for video content delivery, TCP/IP is nowadays more frequently used than RTP/UDP, which was originally designed for delivering multimedia content. Whereas RTP/UDP is designed to adapt to varying network conditions, TCP/IP may struggle due to its connection-oriented nature that provides a reliable data delivery. However, video content providers have to deliver the best optimal video quality to their users, i.e., they have to provide a high Quality of Experience (QoE). Thus, most video content providers use HTTP Adaptive Streaming (HAS) over TCP to deliver content to their users to cope with varying network conditions. With HAS, a video player can choose between different quality levels of a video dynamically, thus the video playback can be adapted to the current network conditions. Furthermore, in order to allow a continuous video playback without any interruptions, i.e., stalling events, a player may request and download different quality levels simultaneously.

The streaming configuration for videos is dependent on the video content providers. For instance, providers determine the amount of available quality layers of videos. Furthermore, the streaming strategy, e.g., in which quality the video should start playing, which quality levels should be requested, or when the video should switch between different quality levels, is dependent on the video provider and its provided player as well. However, as the video client is only roughly aware of its network access condition and even completely unaware of the overall network conditions, the player's streaming strategy may not always lead to an improvement of the perceived video quality of users. Thus, triggered quality requests may even result in a further degradation of the overall network performance, thus a decrease of video quality. Besides, today's networks are also not aware of details of transmitted video content due to lack of information exchange between video content providers and network operators. Consequently, operators are not able to provide network resources in a way that allows video clients adapting to the current network situation in an optimal way.

In the Internet, YouTube is one of the dominant video content providers. In North America, YouTube's video traffic counts with 17.61 % for the largest proportion of the total mobile network traffic, and with 13.19 % for the second largest amount in fixed access networks [1]. Whereas YouTube is responsible for a big share of Internet video traffic, less is known about its streaming set-up for video content, and in particular, how its adaptive streaming mechanisms behave under varying network conditions. Existing works [2], [3], [4], [5] either focus on YouTube's video streaming mechanism based on Flash Video (FLV) or did not investigate its dynamic behavior under varying network conditions. With this paper, we provide a new measurement set-up that allows us to quantify the behavior of video content delivery services. In particular, we demonstrate how to use a dyadic monitoring system to quantify YouTube's video streaming behavior under varying network conditions in detail. Knowledge about YouTube's streaming behavior may enable network operators to allocate their resources in such a way that both the network operator and the video content provider may benefit.

The experimental set-up used for measuring YouTube's performance is composed of virtual components, i.e., virtual machine images, and OpenFlow-based software switches, i.e., open vSwitch [6]. Such a set-up allows to distribute and to replicate all experiments. For measuring YouTube's performance, we combine two existing monitoring mechanics, an application (client/browser-based) monitoring and a network-based monitoring. Combining both monitoring approaches allows to analyze YouTube's video streaming performance. To quantify YouTube's performance in this paper, we focus on video content that was downloaded but not displayed, i.e., video content that was unnecessarily downloaded. Unnecessarily downloaded video content directly refers to wasted bandwidth, i.e., an in-efficient use of available network resources.

The remainder of this paper is structured as follows. In Section II, we explain HAS, refer to related work, and explain YouTube's mechanism that results in wasted traffic. In Section III, we outline in detail the dyadic monitoring system and how YouTube's redundant traffic metric is estimated, which is used in this paper to quantify YouTube's performance. Section IV presents the measurement results of the experiments. Finally, we conclude and propose future research prospects in Section V.

II. Background & Related Work

In the following, we first discuss HTTP Adaptive Streaming (HAS). Afterward, we present the related work in the relevant areas of research. At the end of the section, we examine the behavior of YouTube's adaptation algorithm that leads to the observed redundant traffic.

A. *HTTP Adaptive Streaming*

HTTP Adaptive Streaming (HAS) allows for client-driven adaptation of video content served by a HTTP server to device capabilities (e.g. screen size) and available bandwidth. For HAS, the content provider segments the video into fixed-length segments (e.g. 2 seconds) and encodes each segment into different representations (e.g. quality levels). The segments in different representations are served to the client through the HTTP protocol. In MPEG-DASH, a wide-spread HAS standard adopted by YouTube, the URLs and encoding information (e.g. average bit-rate) of the segments are made available to the client through an XML-encoded manifest file (i.e. the Media Presentation Description (MPD) file). The client first requests the manifest file to learn the location and properties of the different segments. Afterward the client selects the segments of a desired quality level based on an unspecified adaptation algorithm. The adaptation algorithm is not part of the MPEG-DASH standard. Yet 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].

B. *Related Work*

In [2] the authors evaluate the three streaming providers Hulu, Netflix and YouTube in terms of streaming behavior, traffic redundancy and bandwidth exploitation. For traffic redundancy, the results for a single experiment are given. In the experiment a tablet device (iPad) running the YouTube application plays back an unspecified 750 second YouTube clip. The device is connected over WiFi to a shaped bottleneck link. The bandwidth (in Mbps) of the bottleneck link is set to a specific static bandwidth from the list $[5, 0.5, 1, 2, 4, 5]$. With the start of the experiment, the bandwidth is set to the first value in the list (i.e. 5 Mbps) and every 2 minutes the subsequent value from the list is selected. For this scenario the authors observe that YouTube switches aggressively to the next higher bit-rate when the bandwidth of the link is increased, dropping all buffered data of the lower qualities in that process. For that experiment the authors calculate a percentage of redundant traffic of 16 % for YouTube (Hulu 21 %, Netflix 22.5 %). In our work, we confirm that behavior and evaluate the implications for the network in greater detail. In [4], the authors evaluate mobile video traffic and show that in some cases more than 35 % of the traffic is redundant. As cause for this behavior, the authors discuss frequent termination of TCP connections and discarded on-fly packets. In [5], [8], the authors model YouTube's traffic. Their findings suggest that the adaptation behavior of YouTube changes with choose of device and player technology (e.g. from Flash to HTML5). In [9], the authors show how users could benefit from an information exchange between YouTube and the network. Furthermore, they show how application control (e.g. triggering quality switches by a local gateway resource management) can increase the resource usage efficiency. An application control by the network can help to prevent the redundant traffic observed by our work. In [10], the authors evaluate YouTube's application flow control mechanism. The results show that YouTube uses block sending to transfer video data in bursts and that 32 s of block sending are used in the initial buffering phase. The observed block sending approach may increase the number of overlapping segments if the block sending period is fixed to one quality level.

C. *Wasted Bandwidth & YouTube Behavior*

In the following, we describe the behavior of YouTube's adaptation algorithm which causes the duplicate download of playback intervals. Figure 1 illustrates the observed behavior. The figure is taken from one of our experiment runs and shows the range requests issued by YouTube's adaptation algorithm on the axis to the right. The time since the start of the experiment run when a new request is started is shown on the axis to the left from bottom to top. The figure shows an experiment with three quality switches (at 27 s, 45 s and 54 s) and the corresponding wasted range requests of the lower qualities. At first, the algorithm issues four consecutive range requests for the lowest quality level (i.e. 144p). Afterward, the algorithm starts to request one range of the next higher quality level (i.e 240p). But the requested range overlaps with the previously downloaded range of the quality level 144p. Next, the algorithm revises his decision and selects quality level 144p again before it switches directly to 360p. Overlaps can also be observed for the quality switches to 360p and 480p.

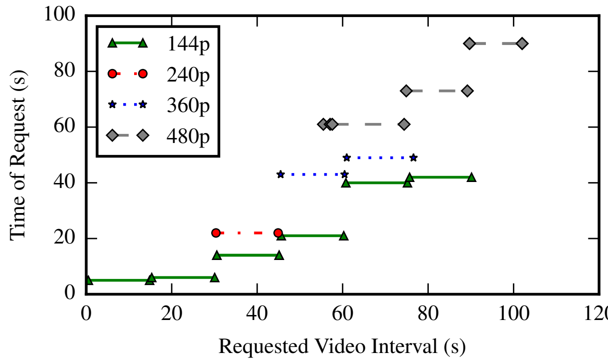

Fig. 1. Example for YouTube range requests taken from an experiment run. First ranges with 144p are selected. Later switch to 240p, 360p and 480p.

Presumably, this way the adaptation algorithm tries to maximize the user's perceived QoE in situations where it detects or assumes an increase in available bandwidth. By revising a previously made decision for a quality level for a specific range, the player can present the user with a higher quality earlier. This comes with the cost of the lost buffer contents of the previously selected lower quality level. The observed behavior is also discussed in [2].

III. Methodology

In the following, we discuss the methodology of our evaluation. We first give an overview on our experimental set-up consisting of a virtual environment with a browser and a virtual network. Next we introduce the two monitoring approaches used, namely in-network monitoring and browser-based monitoring. Afterward, we discuss how we estimate the amount of redundant data requested by the YouTube player. Next, we introduce the two reference bandwidths and three switching frequencies (referred to as traffic shaping patterns) used in the evaluation. Afterward, we give details about the three video sequences used in the evaluation. At the end of the section, we introduce the software utilized and implemented in our experimental set-up.

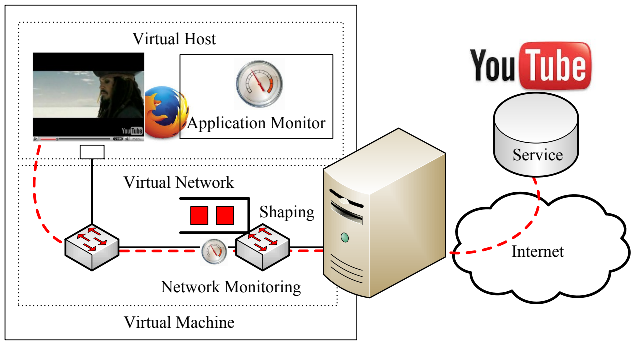

Fig. 2. Experimental set-up based on a virtual machine (VM) and virtual network with browser-based and in-network-based monitoring.

Figure 2 gives an overview on our experimental set-up. To allow for ease of distribution and reproducibility of the experimental environment, we implement the environment as a virtual machine (VM) image. Inside the VM we deploy a Linux-based operation system with graphical user interface, a web browser and a virtual network based on Mininet [11] connected to the Internet. In the current version of our experimental set-up the nodes emulate Layer-2 forwarding of commodity Layer-2 switches. Traffic shaping is done on the virtual network node connected to the Internet. The measurements are automatized and scripted based on configuration files. Each experiment run is limited to a maximum run-time of 800 seconds and after each experiment the virtual machine is reset to a default state. Per experiment run only one video playback is started and not repeated when the video finishes before the maximal run-time limit of the virtual environment.

A. Monitoring

Two monitoring approaches are used to record the state and behavior of the YouTube client application (i.e. the browser), in-network and browser-based monitoring. The in-network monitoring approach uses deep-packet inspection (DPI) to analyze and record the range requests made by the browser. For the browser-based monitoring we deploy a browser add-on. The two monitoring approaches complement each other.

TABLE I. Monitoring placement & metrics

	Browser-based	In-network
Playback Quality	x	
Playback Position	x	
Playback State	x	
Buffer Level	(x) [1]	
Range Request		x
Downloaded Bytes		x

Table I gives an overview on the different capabilities of the two approaches in terms of metrics which are recorded by each of them. Through monitoring of the video player in the browser, we record the current playback quality (e.g. 180p), playback position (in seconds) and state (i.e. paused or playing). Furthermore, we record the buffer level of the currently playing quality level. By in-network monitoring we record all range requests and the amount of data transferred for each request. Next, we discuss both measurement approaches in detail.

1) In-Network Monitoring & Bytes-to-Second Index: Capturing and analyzing the packets transmitted ”on the wire” in the network allows to accurately quantify the amount of transferred data between two endpoints. Furthermore, by the means of DPI we gain an insight into the application protocol in use. In the case of YouTube the application protocol is HTTP (or HTTPS), which in turn uses the stream-based transport protocol TCP. With HTTPS, where the transport is encrypted by SSL, it is not possible to read the application data without invalidating the end-to-end property of the secure connection (e.g. through a men-in-the-middle attack by a proxy server). In our experimental set-up we disable the use of HTTPS for YouTube. For the remainder of this work we assume an unencrypted HTTP connection. This does not change the generality of our findings.

We deploy the DPI application on the virtual node connected to the Internet and in the first step capture all packets belonging to TCP connections with HTTP GET requests and responses. For the GET request, three URL-encoded parameters are of relevance for us. The *itag* parameter, which selects a specific quality of the video, the *range* parameter, which denotes a byte range of the video the client wants to download, and the *video id*. Note that this is a custom range selection mechanism, not the range parameter specified in the HTTP protocol. Also note that this way, the YouTube server may choose to not deliver the full byte range requested by the client, but a shorter one. To translate byte ranges to playback seconds, we download the video once in full, parse the encoded video data and create a bytes-to-second index.

2) Application Monitoring: In order to monitor the application metrics, we deploy a browser add-on. The add-on records four metrics. The player state (playing, paused), the playback position in video seconds, the playback quality and the the buffer level for the currently selected playback quality in seconds. We record the four metrics once per second.

3) Redundant Traffic Estimation: We define redundant traffic as the percentage of the total video size that was downloaded but not shown to the user. Be s a point in video time in seconds from the beginning of the video. Be $\beta(q, s)$ the number of bytes required to decode and display the video

[1]Only currently selected quality level.

TABLE II. TRAFFIC PATTERNS

Pattern	Interval (secs)	Bandwidth Patterns (Mbps)
1	No changes	0.5
	60	0.5 0.4 0.3 0.4 0.5 0.6 0.7 0.6 0.5
	120	0.5 0.25 0.5 0.75 0.5
2	No changes	1.0
	60	1.0 0.9 0.8 0.9 1.0 1.1 1.2 1.1 1.0
	120	1.0 0.75 1.0 1.25 1.0

TABLE III. VIDEOS USED IN THE EVALUATION

Video	Length	Description
Clip 1 [2]	101 seconds	Scene from a baseball match with commentary.
Patra [3]	843 seconds [5]	Point-of-view shot taken from a car driving down a steep mountain road.
Home [4]	5597 seconds [5]	Environmental documentary shot entirely from a helicopter.

interval $[s, s+1]$ on quality level q and be α the total number of video bytes downloaded during the playback of the video as recorded by the in-network monitor. Be l the length of the video in seconds. Furthermore, we denote $Q(s)$ as the quality level selected at video playback position s. The ratio of redundant traffic δ is calculated as $\delta = \frac{\alpha - \sum_{i=0}^{l} \beta(Q(i),i)}{\sum_{i=0}^{l} \beta(Q(i),i)}$. In a nutshell, the percentage describes the overhead relative to the bytes required for the playback of the video without overlapping playback time segments.

B. *Traffic Shaping & Shaping Patterns*

In order to evaluate the behavior of the YouTube player under varying available bandwidth conditions, we control the Internet downlink bandwidth on the virtual network node connected to the Internet. Table II presents the two traffic patterns used in this work. The first pattern oscillates around a reference bandwidth of 0.5 Mbps with a minimum bandwidth of 0.25 Mbps and maximum bandwidth of 0.75 Mbps. The second around a reference bandwidth of 1.0 Mbps with a maximum bandwidth of 1.25 Mbps and minimum bandwidth of 0.75 Mbps. The reference bandwidth values are chosen loosely based on the average bit-rates of the quality levels 360p and 480p of the three videos selected for this work. For each pattern we introduce two inter-arrival intervals for the bandwidth changes, 60 seconds and 120 seconds. E.g., for the 60 seconds interval, each traffic shaping setting stays active 60 seconds long before a new traffic shaping value is selected. The values listed in Table II for each pattern and interval are set active in the order as presented in the table.

C. *Selected Video Sequences*

Table III presents the three videos selected for the evaluation. One short clip selected from the list of popular videos, one 10 minute video and one full movie. Clip 1 presents a 101 seconds scene from a baseball match. The 10 minute video (*Patra*) shows a scenic first-person view of a car driving down a mountain. The selected movie (*Home*) is a nature documentary filmed entirely from a helicopter. Table IV gives the average bit-rates of the four quality levels 144p, 240p, 360p and 480p of each clip. The bit-rate range from 0.105 Mbps (Clip 1, 144p) to 1.022 Mbps (Patra, 480p).

D. *Used Software*

The experiments are conducted on a virtual Xubuntu 14.04 64-bit. For traffic shaping the Linux Traffic and Control framework is used. The YouTube web-page with the video content is fetched and displayed by Firefox. As video player the Flash player is selected. The Firefox playback status and buffer level is monitored using *YoMo plugin* [12]. The in-networking monitoring was accomplished by *YoMo* [13]. The Bytes-to-playback-seconds index of the downloaded video content was created utilizing MP4Box of the GPAC project [14].

TABLE IV. AVERAGE BIT-RATES OF USED VIDEO SEQUENCES

Quality	Clip 1	Home	Patra
144p	0.108 Mbps	0.105 Mbps	0.109 Mbps
240p	0.245 Mbps	0.242 Mbps	0.245 Mbps
360p	0.419 Mbps	0.477 Mbps	0.443 Mbps
480p	0.772 Mbps	0.933 Mbps	1.022 Mbps

IV. RESULTS

In the following, we present the results of our evaluation for the static and dynamic bandwidth case. At the end of the section, we discuss and summarize the main findings. Each experiment was repeated 50 times. Our experimental set-up limits the maximal run-time to 800 seconds. Longer videos are interrupted when reaching the time limit and a new experiment is started. If not otherwise noted, the data is presented as box plots with the median, first and third quartile and whiskers for the most extreme values which are not considered outliers. Outliers (i.e. data points outside of 1.5 times the inner quartile range) are omitted. The mean of the data is presented as red diamond. Next we discuss results for static bandwidth.

A. *Static Bandwidth*

Figure 3 gives the percentage of redundant traffic for Clip 1 for a set of static bandwidths. The evaluated bandwidth values range from 0.2 Mbps to 1.5 Mbps in steps of 0.1 Mbps. Furthermore, we evaluate a static bandwidth of 10 Mbps as a reference and verification. 10 Mbps is ten times the average bit-rate of the evaluated video and sufficient bandwidth to avoid quality switches during playback.

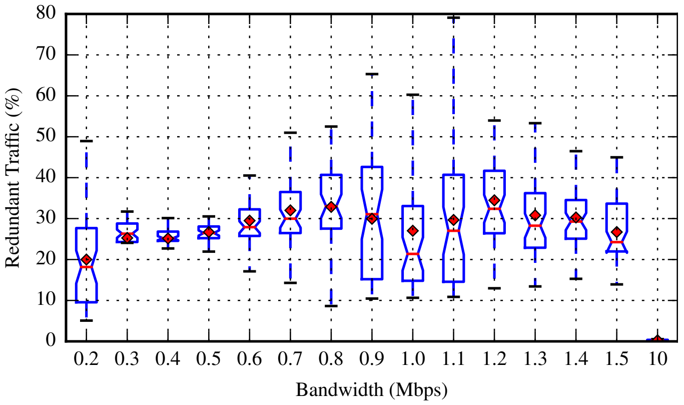

Fig. 3. Percentage of redundant traffic for Clip 1 for a set of static bandwidths.

The lowest percentage of redundant traffic is observed for a static bandwidth of 0.2 Mbps, with a median of 18 %.

[2]https://www.youtube.com/watch?v=u42d_upf-5g, accessed 15-02-2015
[3]https://www.youtube.com/watch?v=Ad1RGjg2ips, accessed 15-02-2015
[4]https://www.youtube.com/watch?v=jqxENMKaeCU, accessed 15-02-2015
[5]Our experimental setup limits the time of playback to 800 seconds.

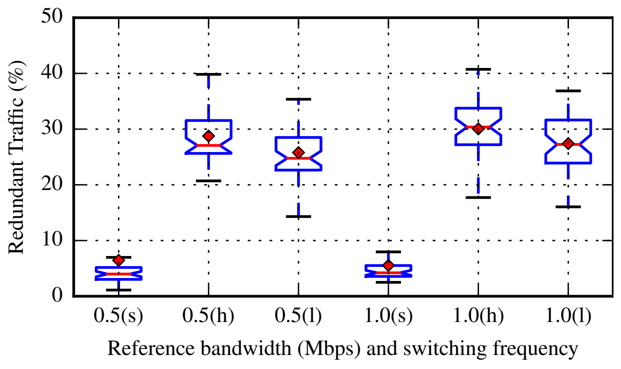

Fig. 4. Patra: Redundant traffic for two **s**tatic reference bandwidths and two switching frequencies (every 60 seconds as ***h****(igh)* and 120 seconds as ***l****(ow)*).

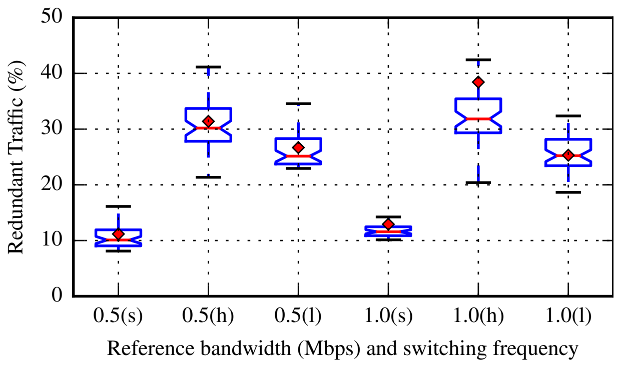

Fig. 5. Home: Redundant traffic for two **s**tatic reference bandwidths and two switching frequencies (every 60 seconds as ***h****(igh)* and 120 seconds as ***l****(ow)*).

For 0.8 Mbps, 1.2 Mbps and 0.9 Mbps, we observe a median redundant traffic percentage of over 30 % (33 %, 32 % and 31 %, respectively). For the reference bandwidth of 10 Mbps, we observe a median of 0.2 %. Furthermore, we notice that the 1st to 3rd quantile intervals are the largest encompassing a bandwidth of 1 Mbps. Also we observe the second lowest median of about 21 % for 1 Mbps.

For Patra and Home, two static bandwidth values are evaluated, 0.5 Mbps and 1.0 Mbps. For Patra, a median redundant traffic percentage of 4 % for both 0.5 Mbps and 1.0 Mbps is observed. For the Home video, we measure a median redundant traffic percentage of 10 % and 12 % for 0.5 Mbps and 1.0 Mbps, respectively. More details are given in the subsequent section IV-B, where the static bandwidth results of Patra and Home are used as a reference.

B. Dynamic Bandwidth

In the dynamic bandwidth scenario we evaluate the two traffic shaping patterns presented in Table II for the video sequences Patra and Home. Table V summarizes the results as the medians for all bandwidth/frequency combinations.

Figure 4 depicts the percentage of redundant traffic for the Patra video sequence for the two reference bandwidths (0.5 Mbps and 1.0 Mbps) and the three switching frequencies (static, every 60 s and every 120 s). For the static case, the data shows a low redundant traffic of 4 % for both reference bandwidths. Additionally, the experimental results for the two static cases show a low variation (a difference from first to third quartile of about 2 %). For the two switching frequencies, the percentage of redundant traffic is considerable higher. For the high (i.e. every 60 s) switching frequency, we observe a percentage of redundant traffic of 27 % (0.5 Mbps) and 30 % (1 Mbps), with a first to third quantile range of about 6 %. For the lower switching frequency (i.e. every 120 s), we observe a lower median percentage of 25 % and 27 %, respectively.

Figure 5 presents the results for the Home sequence. In the following, we compare the findings for the Home sequence to the Patra sequence. For the Home video sequence, we notice a considerable higher percentage of redundant traffic for the static bandwidth case. Whereas the experimental results for Patra show a percentage of 4 % for both reference bandwidths, the evaluation of the Home sequence show a percentage of 10 % and 12 %, respectively. For the two switching frequencies, the median percentage of redundant traffic is similar for both sequences. For switches every 60 s the median percentage is higher for Home by 3 % and 2 %, respectively. For switches every 120 s, the median percentage is equal for both sequences for a reference bandwidth of 0.5 Mbps. For a bandwidth of 1 Mbps the median percentage for Patra is lower by 2 %.

Table V summarizes the results for the two traffic shaping patterns. For the static reference bandwidth we observe the lowest median percentage of redundant traffic (4 % to 12 %). For the higher switching frequency (i.e. every 60 seconds) we see 27 % and 30 % for Patra and Home with the first pattern, respectively, and 30 % and 32 % for the second pattern.

TABLE V. MEDIAN OF REDUNDANT TRAFFIC FOR HOME AND PATRA FOR DYNAMIC BANDWIDTH SCENARIO

Ref. Bandwidth	Switching Frequency	Patra	Home
0.5 Mbps (Pattern 1)	static	4 %	10 %
	every 60 s	27 %	30 %
	every 120 s	25 %	25 %
1.0 Mbps (Pattern 2)	static	4 %	12 %
	every 60 s	30 %	32 %
	every 120 s	27 %	25 %

C. Discussion

In the following, we discuss the main findings of this work (i.e. the influence factors) and the implications. Table VI summarizes the identified influence factors. Dynamic bandwidth as a factor shows an increase of redundant traffic from 4 % in the static case to 30 % (bandwidth switch every 60 s, 1 Mbps) for Patra. The evaluation of the video length shows, that the short video clip has a higher redundant traffic percentage as the longer video sequences for static bandwidths. Furthermore, we see a dependency on the content. Patra and Home show a difference of up to 8 % for the static bandwidth case.

A possible explanation for the difference in redundant traffic for the two video sequences Patra and Home (4 % to 10 % and 4 % to 12 % for static 0.5 Mbps and static 1.0 Mbps, respectively) can be inferred from the bit-rate histograms of the two sequences. Figure 6a presents the bit-rates of the quality levels 360p and 480p for the first 800 seconds of the Home video sequence. From the figure, we conclude that the bit-rate of the video sequence for the two quality levels is not constant, but can fluctuate during the playback. Whereas the first 200 seconds of the sequence the bit-rate oscillates with

TABLE VI. IDENTIFIED INFLUENCE FACTORS FOR YOUTUBE'S REDUNDANT TRAFFIC

Influence Factor	Description
Dynamic Bandwidth	The percentage of redundant traffic increases for dynamic bandwidths compared to static traffic shaping.
Video Length	The evaluation shows, that the two longer videos have a lower percentage of redundant traffic for static bandwidths.
Content	The amount of redundant traffic depends on the content (i.e. the video). For Patra and Home a difference of 8 % was observed (4 % to 12 % for a static bandwidth of 1.0 Mbps).

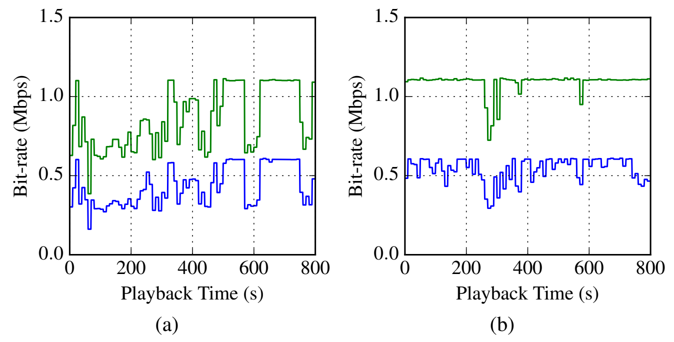

Fig. 6. Bit-rate histogram of the first 800 seconds of the quality levels 360p and 480p for 10 second segments for a) Home and b) Patra.

a mean of 0.3 Mbps and 0.65 Mbps, respectively, the bit-rate of the interval starting from 500 seconds to 750 seconds stays constant at 0.6 Mbps and 1.1 Mbps a high percentage of the time. For the two lower quality levels 144p and 240p not shown in the figure, we observe a different behavior. For them, the bit-rate stays stable at the average bit-rate given in Table IV. Figure 6b depicts the bit-rates for the quality levels 360p of 480p of the Patra video sequence. In contrast to the Home sequence, we do not observe a general trend of increasing bit-rate through the course of the 800 seconds. The bit-rate fluctuations do not exceed a maximum of 0.6 Mbps and 1.1 Mbps and the bit-rate returns quickly to the maximum value after a negative-spike. From the given bit-rate histogram, we can surmise, that the less stable bit-rate of the Home video causes an increase in the frequency of adaptation events. A higher number of adaptation events presumable increases the number of adaptation events where segments are discarded.

The evaluation shows that the characteristic of the bandwidth and the content influence the amount of redundancy caused by YouTube's adaptation. The identified influence factors can help to make better use of network resources. On the side of the content provider (i.e. YouTube), a closer look into the encoding of the videos can help to reduce redundant traffic. By choosing an encoding with less fluctuations, the adaptation algorithm might be more stable and decide less often to revise a previously made decisions. From the perspective of the network provider, an information exchange between the application (i.e. YouTube client or server) and the network can help to reduce the redundant traffic. By knowing the demands of the application (i.e. the bit-rate histogram), the network can assign a suitable share of resources to the application. This way, the network can aim to reduce the redundant traffic to the amount measured for static traffic shaping.

V. CONCLUSION

Video traffic counts for the largest portion of today's Internet traffic. Still, less is known about the behavior of large video streaming applications, e.g., of YouTube, in particular under strongly varying network conditions. However, such knowledge is beneficial for network operators to manage network resources in an application-aware manner, leading to a high resource efficiency. Accordingly, we provide an experimental measurement study of YouTube's HTTP-based adaptive video streaming mechanisms for varying network conditions. Using a dyadic monitoring system that combines client-based and in-network-based monitoring of YouTube's video transmission, we are able to quantify YouTube's behavior in detail. Results show that in case of varying network conditions, YouTube wastes up to 33 % of already downloaded video content. As the unnecessarily video content may not be used, the adaptation mechanism may not always improve the user's perceive video quality in general. Further, it may even burden the network with additional data, thus, decreases the overall network resource efficiency, i.e., network performance. Consequently, network operators should try to provide YouTube's video streaming application with a resource allocation that has a low fluctuation in order to avoid a waste of network resources.

YouTube forces data transmission via HTTPS since January 2015, i.e., secure connections that prevent a simple flow monitoring. Current work focuses on concepts that still allow analyzing YouTube's video performance based on in-network monitoring. We are currently developing an interface that enables the exchange of information between user applications and network operators. Having such an interface, we want to provide resource management mechanisms that allocate network resources in a way that incorporates application knowledge. We assume that such an application agnostic resource management decreases the amount of video content that is unnecessarily downloaded.

REFERENCES

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

[2] 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.

[3] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What Happens when HTTP Adaptive Streaming Players Compete for Bandwidth?" *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video, NOSSDAV '12*, 2012.

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

[5] 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.

[6] "Open vswitch." [Online]. Available: openvswitch.org/

[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] 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.

[9] F. Wamser, D. Hock, M. Seufert, B. Staehle, R. Pries, and P. Tran-Gia, “Using buffered playtime for qoe-oriented resource management of youtube video streaming,” *Transactions on Emerging Telecommunications Technologies*, vol. 24, no. 3, pp. 288–302, 2013.

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

[11] B. Lantz, B. Heller, and N. McKeown, “A network in a laptop: Rapid prototyping for software-defined networks,” in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Hotnets-IX*. ACM, 2010, pp. 19:1–19:6.

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

[13] ——, “YoMo: A YouTube Application Comfort Monitoring Tool,” in *New Dimensions in the Assessment and Support of Quality of Experience for Multimedia Applications*, 2010.

[14] “GPAC — Multimedia Open Source Project:,” http://gpac.wp.mines-telecom.fr/, [Online; accessed January 12th 2015].