From Click to Playback: A Dataset to Study the Response Time of Mobile YouTube

Frank Loh, Florian Wamser, Christian Moldovan, Bernd Zeidler, Tobias Hoßfeld
Institute of Computer Science, University of Würzburg, Germany
frank.loh@informatik.uni-wuerzburg.de

Dimitrios Tsilimantos
Paris Research Center, Huawei Technologies France
dimitrios.tsilimantos@huawei.com

Stefan Valentin
Department of Computer Science, Darmstadt University of Applied Sciences, Germany
stefan.valentin@h-da.de

ABSTRACT

Responding fluently to user requests is important to keep them immersed. In this paper, we are presenting an extensive dataset to study the response time of YouTube's mobile video streaming service on Android. We illustrate the application of our dataset by studying YouTube's *initial delay* for a subset of 9 videos in 75 network scenarios. We find that in 41 % of the cases, YouTube exceeds the attention span of a typical user, while deep immersion is only reached in 15 % of the cases. Our factor analysis implies that the allocation of the initial CDN node is the critical link in this delay chain. Since our dataset includes a large variety of factors, we are describing setup, methodology, and data structure in detail. Our dataset and measurement tools are publicly available at [8].

CCS CONCEPTS

• **General and reference → Measurement**; Evaluation; Performance; • **Networks** → *Network measurement*.

KEYWORDS

datasets, YouTube, video streaming, initial delay

ACM Reference Format:
Frank Loh, Florian Wamser, Christian Moldovan, Bernd Zeidler, Tobias Hoßfeld, Dimitrios Tsilimantos, and Stefan Valentin. 2019. From Click to Playback: A Dataset to Study the Response Time of Mobile YouTube. In *10th ACM Multimedia Systems Conference (MMSys '19), June 18–21, 2019, Amherst, MA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3304109.3325819

1 INTRODUCTION

The key recipe for user immersion has been the same for 50 years: Keep the response time low enough so that the user's flow of thought is not interrupted. From the seminal works of Miller [9] and Newell [11], we know that delays for the user interface below 0.1 seconds are not noticed, delays from 0.1 to 1 seconds are noticed but still feel "fluent" to most users, while users consider delay of more than 2 seconds as "closure" and will change task when delays exceed 10 seconds.

Keeping response time in the single digits is, thus, quite important for services that try to keep their users immersed. YouTube is a prominent example of such a service, since (i) its advertisement-driven business model relies on user attention and (ii) the fluency of video content has to be matched with a responsive user interface.

In this paper, we are describing an extensive dataset to study the response times of YouTube's mobile application for Android. The dataset consists of time-synchronized log files from the network, transport, and application layer. The measurements were performed from November 23, 2017 to February 05, 2019 for 75 network scenarios and include 4498 runs with 783.88 hours of video playback time. Our dataset and measurement software are available at [8]. We will provide a detailed description of the experimental setup, methodology, and data structure in the following sections to enable the reproducibility of our results and to motivate further studies in the community.

To illustrate the usage of this data, we are studying YouTube's *initial delay*, i.e., the time between requesting a certain video and the playback of its first video frame. Initial delay measures how quickly YouTube's user interface performs its main functionality, i.e., the selection of videos and their delivery, and how often it cannot stay within a user's attention span. For detailed analysis, we separate YouTube's response time into three sub-metrics. First, *page load* corresponds to the time between the user's selection of the video and the DNS request to the video server. This metric measures, how quickly the service can redirect the user request to the actual streaming content on a certain node in its Content Delivery Network (CDN). Our second metric, *request to play* measures the delivery speed of YouTube. That is, the time between the DNS request for the first video chunk and the playback of the first video frame. Finally, we aggregate these two metrics to the response time for the first video frame, called *initial delay*. For each of these metrics, we analyze the effect of chosen video, bandwidth, and video quality.

After studying these metrics and factors for our dataset, we find that:

(1) a user experiences an initial delay of less than 2 seconds in 59 % of the cases and less than 1 second in 15 % of the cases.
(2) the chosen video and initial streaming bandwidth have the strongest effect on the request to play time, which dominates the overall initial delay.

MMSys '19, June 18–21, 2019, Amherst, MA, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6297-9/19/06…$15.00
https://doi.org/10.1145/3304109.3325819

(3) bandwidth is the second dominating factor for initial delay, while the effect of video quality is still significant but lower.

Comparing our measurements with the above-mentioned psychological constraints allows to state that YouTube's initial delay is far from ideal. In 41 % of the cases, YouTube does not start streaming within the attention span of a user. "Immersive" reaction time of 1 second is only reached for 15 % of the studied videos and scenarios. The high variance of *page load time* and its dependency on the chosen video, implies that the initial assignment of user requests to the corresponding CDN node is the most volatile step in YouTube's streaming process. This first step is certainly worth starting further investigations with the dataset.

Our paper is structured as follows. While the next section summarizes some related work, Section 3 details the experimental setup, metrics, and factors. Section 4 explains the studied scenarios and Section 5 the provided dataset. Section 6 presents our results and Section 8 draws the main conclusions.

2 RELATED WORK

The mobile use of YouTube's service has been only rarely covered by public datasets. Instead, most available data have been collected by using a personal computer as a streaming device [1]. This, however, cannot provide representative measurements for mobile use since YouTube's web player (running inside a web browser on a personal computer) operates differently than its native application for mobile operation systems [12, 14]. Although YouTube follows the Dynamic Adaptive Streaming over HTTP (DASH) standard [4] on both platforms, it employs different adaptive streaming algorithms [12]. The few public datasets that include measurement of mobile clients [12, 13], however, merely provide network traces. Since YouTube employs Transport Layer Security (TLS) to encrypt application-layer information, focusing on packet traces alone can only offer a limited view on the streaming process.

Our dataset aims to fill this gap by providing measurements that were simultaneously obtained at the network, transport, and application layer in a time-synchronous manner. Measurements at the application layer are performed with YouTube's native application, running on top of the Android mobile operation system. Android is not modified and the software runs on off-the-shelf Smartphones.

The measurement methodology was previously used in [5, 6]. Although these works use a similar setup, the current dataset provides delay as a new metric with derived measures, extended measurement duration and scenarios, data at the network level in *pcap* format, and TCP-based data while [5] offers QUIC traffic. Note that this dataset is based on log files from tcpdump [15] and observations of application output. Consequently, we neither record content nor manipulate applications.

3 EXPERIMENTAL DESIGN

In this section the setup and the data collection process for our experiments are described.

3.1 Setup

The experimental setup is an extension of the testbed in [5, 6, 14], which now captures a number of delay-defining events described below. The testbed provides reliable, automatic, and reproducible experiments without disturbing the streaming process to YouTube's native application on the Android operation system for mobile devices.

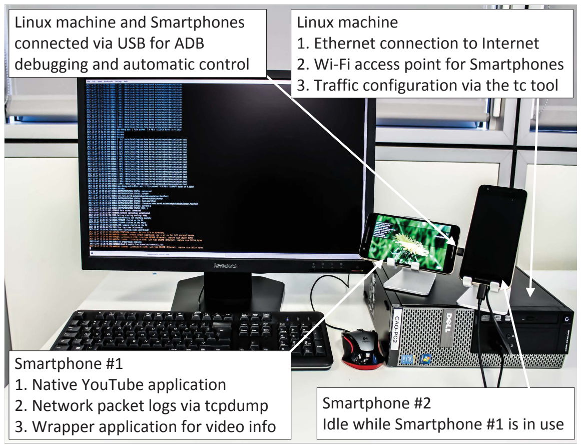
Linux machine and Smartphones connected via USB for ADB debugging and automatic control
Linux machine
1. Ethernet connection to Internet
2. Wi-Fi access point for Smartphones
3. Traffic configuration via the tc tool
Smartphone #1
1. Native YouTube application
2. Network packet logs via tcpdump
3. Wrapper application for video info
Smartphone #2
Idle while Smartphone #1 is in use

Figure 1: Hardware setup for our YouTube measurements at the network, transport, and application layer

The testbed is installed at the University of Würzburg and its hardware is shown in Figure 1. It consists of two Google Pixel XL Smartphones running Android 7.1.1 and a Linux PC running Ubuntu 14.04.4 LTS with (i) an Ethernet connection for Internet access and (ii) a Wireless Local Area Network (WLAN) access point for the Smartphones. Besides this WLAN link, each phone is connected to the PC via the Universal Serial Bus (USB) to control the phone via the Android Debug Bridge (ADB) [2] and to charge its battery. With this setup, typical home WiFi, LTE, or 3G bandwidth settings can be emulated in a controllable environment.

This configuration allows to control the mobile streaming process with common tools for Linux and Android. First, Linux' traffic control [7] is used on the WLAN links to emulate certain network profiles by limiting data rate or by adding packet errors and delay. Second, the PC uses the ADB to trigger events, such as typing or touching a specific point of the screen. This way, it interacts with the YouTube application exactly as a user would normally do to select, for example, a specific video and its playback resolution. In our setup, all these control actions are pre-configured and can run in a scripted set of experiments for weeks.

3.2 Data Collection

To monitor the YouTube streaming session, (i) the *tcpdump* traffic capturing tool [15] is used to collect network and transport layer information related to the stream, and (ii) our own *Wrapper App* [14] is used to extract application layer parameters. This application records streaming-related information from YouTube's *stats for nerds* feature. Since this information is openly presented to the user, our application can read it automatically by executing touch and swipe events on the phone via ADB. Interested readers are referred to [5, 14] for a detailed documentation and the free download of the Wrapper App. Tcpdump records Transmission Control Protocol (TCP) and the Internet Protocol (IP) headers on the Smartphones. Note that, unlike most deep packet inspection (DPI) techniques,

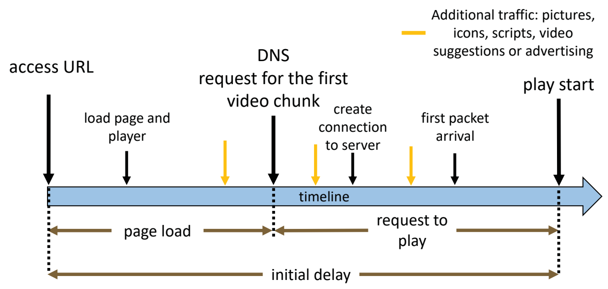
Additional traffic: pictures, icons, scripts, video suggestions or advertising
access URL
load page and player
DNS request for the first video chunk
create connection to server
first packet arrival
play start
timeline
page load
request to play
initial delay

Figure 2: Timeline of a YouTube streaming session with events and time-intervals considered in our study

we neither record nor analyze payload. The same applies to video content data.

Figure 2 illustrates all events and time intervals related to our initial video delay study. The *initial delay* time interval starts with the initial access of the video's URL and ends with the start of the playback. The times of these events are captured at application layer with the Wrapper App. While the first event is triggered by the measurement scripts and can, thus, be simply logged, the second event requires to capture and interpret the *df-flag* from stats for nerds. With this flag, played frames can be counted, indicating the start time of playback in a resolution of 1/frame rate.

Between these main events, an *initial DNS request for the first video chunk* is sent to the IP address of the YouTube server. In particular, we are using the time of this DNS request asking for the IP address of *googlevideo.com* from the tcpdump log files. In this way, it is possible to separate the video data flow on network layer from cross-traffic like pictures, icons, or advertising. Table 1 summarizes the recorded parameters. In the following, the duration from initial accessing the YouTube URL to the DNS request of the first video chunk is named *page load*. Afterwards, the duration to the actual play start is defined by *request to play*.

Table 1: Selection of recorded parameters

Layer	Tool	Parameter
Network	tcpdump	packet arrival time [s]
	tcpdump	packet source IP address
	tcpdump	packet source port number
	tcpdump	packet destination IP address
	tcpdump	packet destination port number
Transport	tcpdump	packet payload size [Bytes]
Application	Wrapper App	time of initial URL access [s]
	Wrapper App	time of initial video play start [s]
	Wrapper App	initial video quality

4 SCENARIOS

In this section the scenarios and configurations used to obtain the dataset are described. In particular, we show how the data can be grouped into different classes. The varied parameters during the measurement include the initial bandwidth, bandwidth changes during the video playback, and changes in the video resolution. In total, 75 different scenarios and 28 videos have been measured. For each scenario, a different subset of videos is used. Each measurement for a scenario is repeated at least ten times. In every setting, only one video is played out during a video session to avoid side effects due to protocol interactions.

In many scenarios the bandwidth is limited after a given period. In total, 24 different initial bandwidth settings are measured. They range from 10 kbit/s to 100 Mbit/s. Our scenarios can be categorized in terms of the bandwidth characteristics into the following classes:

(1) A constant bandwidth is used as baseline.
(2) Sudden drops in bandwidth as happening when connectivity is lost (e.g., tunnel scenarios) that may trigger resolution changes or stalling events. The bandwidth can drop once or repeatedly in a circular way.
(3) Furthermore, a step-wise bandwidth decay is done in some measurements, which leads with DASH to quality changes and decreasing resolutions. Constant and random step durations have been studied.

The resolution is set to *auto* in most of the scenarios. With this option, YouTube's DASH implementation automatically selects a resolution based on the provided adaptation logic, i.e., based on buffer state and/or available bandwidth. In some scenarios the resolution is controlled manually. A given resolution is selected for the complete video duration or it is changed manually at certain points in time. This disables DASH and stalling may occur.

The bandwidth settings are combined with resolution settings. For each tested combination, the times at which changes in the bandwidth occur are varied. The exact parameters that are used in each scenario are described in configuration files that are provided with the dataset at [8]. This set includes data for 28 YouTube videos, played with the YouTube mobile application. We chose videos of different content types and different duration to cover a broad range of video bitrates.

To study the influence of video resolution and bandwidth on initial delay, two subsets of scenarios are measured and compared: First, an automatic video resolution is chosen while the bandwidth is varied between 0.5 Mbit/s and 100 Mbit/s with different steps according to the provided configuration file. Second, the video resolutions are fixed to 240p, 360p, 480p, or 720p and a constant bandwidth in the range of 0.5 Mbit/s to 100 Mbit/s is set.

5 DATASET

This section provides an overview of the content and structure of the dataset. With the measurement setup described in Section 3, 4498 runs containing 783.88 hours of playback time have been measured between November 23, 2017 and February 05, 2019. Each measurement is described by its scenario and measured YouTube video ID. The captured information is collected in a file named as follows: *<type of measurement>* *<timestamp>* _*<scenario>* _*<videoID>* _*<information>.csv*. The initial bandwidth setting for all scenarios is presented in the *initial_bandwidth.csv* located in the dataset folder. The scenario overview is listed in the *scenarios.txt*.

For all measurements, the complete traffic on network layer is recorded in a file called *network_traffic.csv*. It contains the following captured information: packet arrival timestamp, packet length in

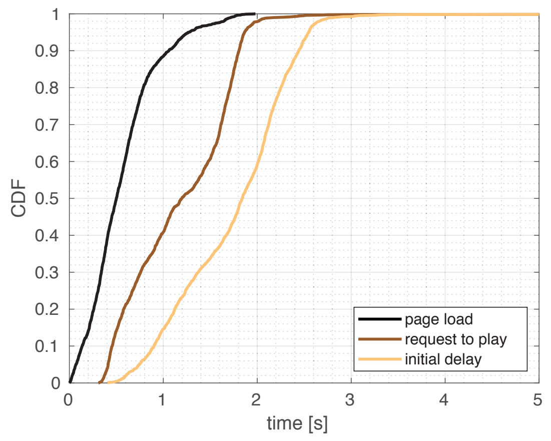

Figure 3: CDF over the complete dataset for three metrics of initial response time

bytes, source and destination IP address, source and destination TCP port if applicable, and the payload protocol number.

Furthermore, all active transport layer traffic flows are summarized in a file called *network _flows.csv*. Therein, the source and destination IP address, source and destination TCP port, flow start timestamp, flow end timestamp, flow duration, flow size in bytes, and number of packets the flow contains are listed.

Last, the *initial _delay.csv* contains information about the initial video playback delay. Therefore, the *<type of measurement>* in the path distinguishes between *initialDelayData* and *requestToPlaystartData* measurements. In both data subsets, the initial DNS request, the playback start timestamp, and the duration from DNS request to playback start is logged. The *initialDelayData* folder additionally stores the timestamp for accessing the URL of the YouTube video. Thus, the initial web page access, the time from initial access to DNS request and the duration from initial URL access to playback start, are listed there. These additional information is available for 2995 of 4498 runs. In this context, we would like to refer again to Figure 2, where the different time intervals are illustrated in chronological order for one measurement run.

6 EVALUATION

The following evaluation is based on the investigation of the *initialDelayData* dataset. The measured videos have a minimum duration of 1.6 min, a maximum duration of 71.8 min, a mean duration of 19.2 min, and a standard deviation of 23.7 min. In this dataset, the URL access, the first video chunk DNS request, and the playback timestamp is logged. Thus, the page load, the request to play duration, and the initial delay can be investigated. The analysis is done with all sessions using TCP as network layer protocol, which corresponds to 1909 of the recorded sessions of the dataset. The remaining have been using QUIC and will be analyzed in comparison to TCP in future work.

Evaluating the access URL timestamp, the first DNS request, and the playback timestamp over the complete dataset, Figure 3 provides the Cumulative Distribution Functions (CDFs) for initial delay, for request to play, and for page load respectively. The CDFs show that all page loads are faster than 2 s. In more than 98 % of all runs, the duration for request to play is less than 2 s. The complete initial delay from URL access to playback is below 2 s in 59 % of the cases.

The median for the page load is 0.65 s, for request to play it is 1.03 s, and for the complete initial delay it is 2.02 s. By having a look at the overall initial delay, the request to play duration dominates the page load for all sessions. In more than 40 % of the sessions, the request to play duration is twice as high as the page load time while in close to 30 % it is threefold. In total, the correlation between page load and request to play duration is slightly negative with −0.12.

We will now study the effect of the selected video, initial bandwidth limitation, and initial video resolution on the initial delay. Although a variety of other factors (e.g., such as device type and CPU load) affect delay as well, we are focusing on these three factors as they are easy to generalize. Since, initial delay only depends on bandwidth and video bitrate at the beginning, we are only analyzing the beginning of each session in our dataset.

The impact of the selected video is illustrated in Figure 4 for page load time, in Figure 5 for the request to play duration, and in Figure 6 for the initial delay. For all videos, most page load times are below 1 s. Only the second, fourth, and fifth video of Figure 4 show many outliers. Nevertheless, the maximal median in page load duration is visible for the second video with 0.54 s while the minimal median is shown for the third video with 0.27 s. We conclude that the initial delay is significantly affected by the played video but this effect varies strongly. The variance in the request to play duration leads to a large variance in the total initial delay. This is a result of different bandwidth settings and initial qualities. Additionally, differences might result from different video suggestions or page load times.

We now study the impact of bandwidth on the three metrics for initial delay. Figure 7, shows the page load duration for a constant bandwidth of 1 Mbit/s, 2.5 Mbit/s, 3 Mbit/s, and 5 Mbit/s. Figure 8 shows the request to play delay and Figure 9 shows the initial delay for the same bandwidth settings.

Except for 1 Mbit/s, the page load shows outliers. Like with different videos, the difference in the median is low. Compared to that, a clear tendency to smaller request to play duration is visible with larger bandwidth in Figure 8. The median for 1 Mbit/s is 1.75 s while it is 0.98 s for 5 Mbit/s. This is also visible for the overall initial delay.

The effect of video resolution is illustrated in Figure 10, Figure 11, and Figure 12. For the page load duration, a little amount of outliers is visible except for 720p quality. Overall, the page load is the fastest for 1080p. The request to play duration is highly variant symbolized by the large boxes or the large amount of outliers visible for 480p. Less variance is shown for 1080p. Additionally, there is no tendency visible in the median over all qualities. Since the request to play duration dominates the total initial delay, this is also visible for the initial delay.

Let us now summarize the observed effects of the three studied factors. Changing the video shows a large but highly variant effect on the request to play delay. The effect of bandwidth is more clear. The results show that the bandwidth is negatively correlated to the request to play and thus to the initial delay. It affects initial

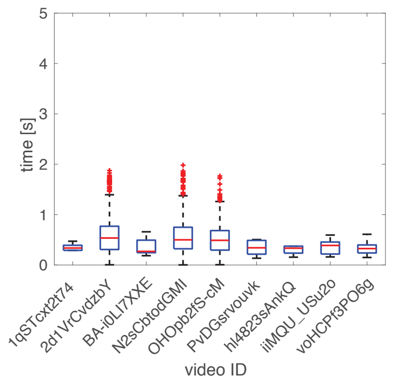

Figure 4: Page load time vs. video

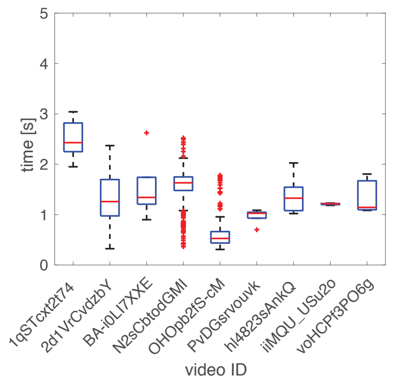

Figure 5: Request to play time vs. video

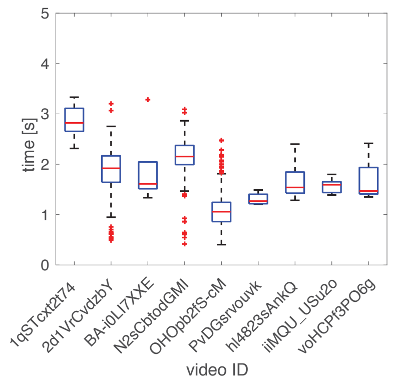

Figure 6: Initial delay vs. video

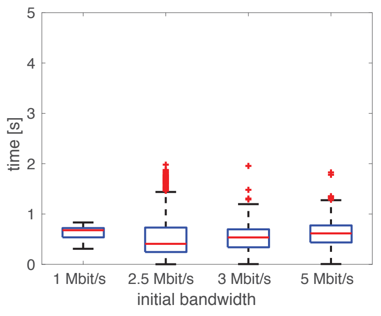

Figure 7: Page load time vs. bandwidth

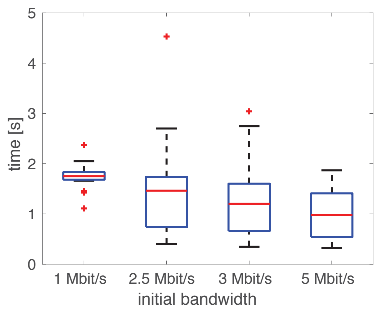

Figure 8: Request to play t. vs. bw

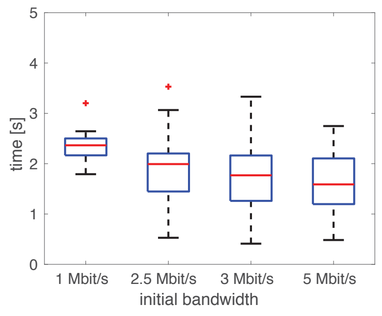

Figure 9: Initial delay vs. bandwidth

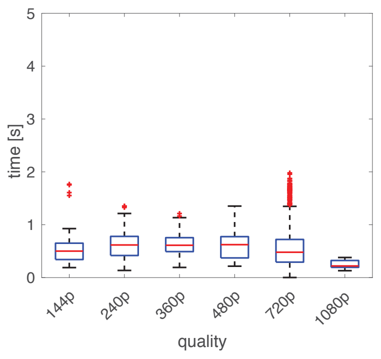

Figure 10: Page load time vs. quality

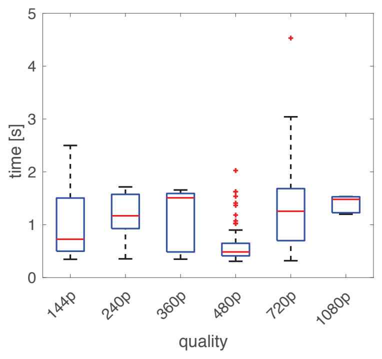

Figure 11: Request to play t. vs. quality

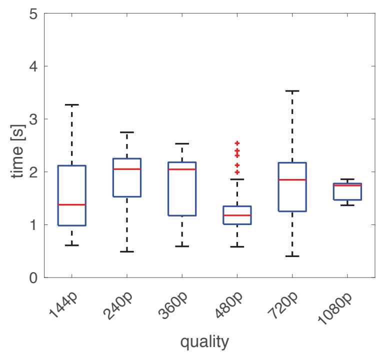

Figure 12: Initial delay vs. quality

delay at a similar average level as the chosen video but with a larger variance for the same bandwidth setting. The average effect of video bitrate is different for the investigated quality settings. While 144p and 720p show large variance in received request to play delays and 480p showed many outliers, the variance with 1080p is much smaller. In particular resolutions 144p, 480p, and 720p show a strong variance due to their choice as default qualities. In contrast, 1080p is only used for a very high available bandwidth.

Comparing the three delay metrics reveals that the request to play duration dominates the page load time, thus, affecting the initial delay most. Table 2 summarizes the main statistical parameters for our delay metrics. We also show the percentage of cases that fell in the user attention spans from [9, 11]. Consequently, a "fluent" initial delay of below 1 s is only reached for 14.67 % of the studied cases. In 44.74 % of the cases, YouTube started streaming between 1 s and 2 s which represents a typical attention span. Nevertheless, 40.76 % of the studied sessions exceed this acceptable time span.

Table 2: Overview of initial delay statistics and their percentage per user attention span [9, 11]

	Parameter of initial delay statistic				Percentage per attention span		
	min [s]	max [s]	median [s]	std [s]	[0s - 1s[	[1s - 2s[	[2s - 10s[
Page load	0.0019	1.9808	0.5573	0.4999	88.48	11.52	–
Request to play	0.3084	20.9584	1.1980	0.7773	40.86	56.99	2.05
Initial delay	0.4043	21.6805	1.8409	0.8191	14.67	44.47	40.76

7 FURTHER USE OF THE DATASET

Initial delay is one of the most important quality factors affecting the YouTube video streaming user experience [3, 10]. By providing both, application and network data, models can be built using this dataset to help application developers and network performance analysts to characterize YouTube's behavior. Hardware manufacturers and mobile service providers rely on application profile properties from models to test and quantify application-side effects of new network configurations and innovations.

For researchers, the dataset is important to provide, to complete, and to update models for YouTube's performance in mobile networks with initial delay. The current literature concentrates so far only on the personal computer as streaming device [1]. This dataset helps to explore effects of initial delay, for example through machine learning. It helps to predict how quickly YouTube responds under specific network conditions for performance evaluation of mobile networks with respect to YouTube.

8 CONCLUSION

This paper had two objectives. First, it presented a dataset to study delays of YouTube's mobile application for Android. The dataset provides time-synchronized data from network, transport, and application layer for 783.88 hours of video playback time, measured over 14 months from 2017 to 2019. We are now publishing the dataset and tools at [8] and hope that our detailed description of the experimental setup, methodology, and structure will motivate the wide use of this dataset.

Our second objective was to illustrated the usage of this data by a relevant example. We chose to study the initial response time of YouTube's mobile application for Android. We have studied three measures of responsiveness, each corresponding to a different setup phase of a streaming session, and analyzed the effect of three factors. Grouping the resulting initial delay according to typical attention spans [9, 11] shows that YouTube exceeds a user's attention span in 41 % of the cases and that full immersion is only reached for 15 % of the studied cases. Our factor analysis suggests that the page load time, which includes the mapping of a user request to a CDN node, is the main aspect of this problem. This critical time of the session setup deserves further investigation, e.g., in terms of delays due to routing, database look-ups, and load balancing.

ACKNOWLEDGMENTS

This work was supported by Huawei Technologies under the research project "Quality of Experience Estimation at the Edge (QoE^3)".

REFERENCES

[1] S. Alcock and R. Nelson. 2015. Analysis of YouTube Application Flow Control. Dataset. Retrieved Feb. 7, 2019 from https://wand.net.nz/~salcock/youtube/

[2] Android Developers. 2017. Android debugging tool (adb)-Android SDK Platform. Manual page. Retrieved Feb. 7, 2019 from https://developer.android.com/studio/command-line/adb.html

[3] Tobias Hoßfeld, Sebastian Egger, Raimund Schatz, Markus Fiedler, Kathrin Masuch, and Charlott Lorentzen. 2012. Initial delay vs. interruptions: Between the devil and the deep blue sea. In *2012 Fourth International Workshop on Quality of Multimedia Experience*. IEEE, 1–6.

[4] ISO/IEC. 2014. Dynamic adaptive streaming over HTTP (DASH). *International Standard 23009-1:2014* (May 2014).

[5] Theodoros Karagkioules, Dimitrios Tsilimantos, Stefan Valentin, Florian Wamser, Bernd Zeidler, Michael Seufert, Frank Loh, and Phuoc Tran-Gia. 2018. Dataset: A Public Dataset for YouTube's Mobile Streaming Client. Open Dataset. Retrieved Feb. 7, 2019 from http://qoecube.informatik.uni-wuerzburg.de/

[6] Theodoros Karagkioules, Dimitrios Tsilimantos, Stefan Valentin, Florian Wamser, Bernd Zeidler, Michael Seufert, Frank Loh, and Phuoc Tran-Gia. 2018. A Public Dataset for YouTube's Mobile Streaming Client. In *Network Traffic Measurement and Analysis Conference (TMA)*. 1–6.

[7] A. N. Kuznetsov. 2001. Linux traffic configuration tool (tc). Manual page. Retrieved Feb. 7, 2019 from https://linux.die.net/man/8/tc

[8] Frank Loh, Florian Wamser, Christian Moldovan, Bernd Zeidler, Tobias Hoßfeld, Dimitrios Tsilimantos, and Stefan Valentin. 2019. A Public Dataset to Analyze the Response Times of Mobile YouTube. Open dataset. Retrieved Feb. 7, 2019 from https://go.uniwue.de/initialdelaydataset

[9] R. B. Miller. 1968. Response time in man-computer conversational transactions. In *Proc. of AFIPS Fall Joint Computer Conference*, Vol. 33. 267–277.

[10] Hyunwoo Nam, Kyung-Hwa Kim, and Henning Schulzrinne. 2016. QoE matters more than QoS: Why people stop watching cat videos. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE.

[11] A. Newell. 1990. *Unified Theories of Cognition*. Harvard University Press.

[12] A. Rao, Y. Lim, C. Barakat, A. Legout, D. Towsley, and W. Dabbous. 2011. Network Characteristics of Video Streaming Traffic. In *Proc. ACM CoNEXT*.

[13] S. Sengupta, H. Gupta, Niloy Ganguly, B. Mitra, P. De, and S. Chakraborty. 2015. CRAWDAD iitkgp/apptraffic dataset (v. 2015-11-26). Dataset. Retrieved Feb. 7, 2019 from https://crawdad.org/iitkgp/apptraffic/20151126/

[14] Michael Seufert, Bernd Zeidler, Florian Wamser, Theodoros Karagkioules, Dimitrios Tsilimantos, Frank Loh, Phuoc Tran-Gia, and Stefan Valentin. 2018. A Wrapper for Automatic Measurements with YouTube's Native Android App. In *Network Traffic Measurement and Analysis Conference (TMA)*. 1–8.

[15] The tcpdump group. 2015. tcpdump packet analyzer v. 4.7.4, (libpcap v. 1.7.4). Manual page. Retrieved Feb. 7, 2019 from http://www.tcpdump.org/