Is the Uplink Enough? Estimating Video Stalls from Encrypted Network Traffic

Frank Loh*, Florian Wamser*, Christian Moldovan*, Bernd Zeidler*, Dimitrios Tsilimantos†, Stefan Valentin‡, Tobias Hoßfeld*
*University of Würzburg, Germany, {firstname.lastname}@informatik.uni-wuerzburg.de
†Paris Research Center, Huawei Technologies France SASU, {dimitrios.tsilimantos}@huawei.com
‡Darmstadt University of Applied Sciences, Department of Computer Science, Germany, {stefan.valentin}@h-da.de

***Abstract*—Today's traffic projections speak of almost 58 % video traffic across the Internet. Nearly all video traffic is encrypted, accounting for more than 50 % encrypted traffic worldwide. To analyze video traffic today, or even estimate its quality in the network, a deep look into the traffic characteristics has to be done. But then, important quality metrics from the traffic behavior can be derived. Based on extensive measurements we show in this work how to measure and estimate video stalls for mobile adaptive streaming. The underlying dataset includes more than 900 hours of video footage from the native YouTube app, measured over 18 different videos in 56 network scenarios in two cities in Europe. We outline a possible approach to estimate the video playback buffer size based on uplink video chunk requests in real-time to break down the video stalls. This work is intended as a tool for network operators to receive further knowledge of the characteristics of video streaming traffic to quantify the most important QoE degradation factors of one of the most important applications today.**

I. Introduction

The amount of traffic transfered all around the world in today's Internet is tremendous. Besides the rapidly changing environment, from a network operators point of view, dealing with the vast amount of data is one of the largest challenges. According to recent estimates, almost 58 % of Internet traffic is video [1], [2]. In mobile networks it is more than 70 % [3]. YouTube and Netflix are the biggest contributors [1], both using adaptive video streaming due to the fluctuating mobile conditions [4]. In addition, more than 50 % of network traffic is encrypted in the Internet [1], with YouTube and Netflix both completely encrypting their streaming. Thus, as a network operator, the challenge is to recognize the application quality of video streaming from the encrypted traffic and draw correct conclusions in order to ensure the best possible and cost-efficient transport [5].

Studies show that video experience depends on stalling, initial delay, playback quality, and the frequency of quality changes [6], [7], [8]. This does not only apply to the users' Quality of Experience (QoE) but also to User Engagement [9] or user churn [10]. For that reason, it is important for the network operator and also for the streaming service provider to be able to estimate the traffic characteristics for video streaming. With this knowledge the streaming quality at the users side can be quantified and improved. First of all, this requires a profound understanding of video streaming and a mapping of application situation to traffic characteristics. Second, measurements of different scenarios and network conditions allow an assessment of the situation in order to develop a general approach. Since most streaming traffic is encrypted, the measurements have to be made, both at network layer and on the application side to correlate them and evaluate the network traffic.

978-1-7281-4973-8/20/$31.00 © 2020 IEEE

This paper examines one of the key factors influencing the user's video QoE degradation: the video stalls. Initially, traffic patterns are presented, causally related to video streaming application properties. Statements about the patterns are developed that the buffer filling status can be successively tracked during the whole playback. This knowledge is used to estimate video stalls, i.e. an empty buffer level is estimated out of the traffic patterns. By drawing conclusions from the uplink requests to received video seconds without taking the downlink into consideration, the approach is lightweight from a computational point of view and simple for implementation and utilization.

The contribution of this work is a simple characterization of video streaming in connection with the causal traffic characteristics. The work is intended to give network operators an approach for greater understanding of video streaming traffic within their network without the needs of keeping track of every single packet in the downlink. The work can be seen as a step towards a comprehensive but also simple and efficient video stalling analysis from encrypted network traffic. It allows statements about a well-functioning video streaming or a video streaming session with problems.

The remainder of this paper is structured as follows. Section II outlines basics that help understanding the paper's approach and terminology. It sets the fundamentals for adaptive video streaming that are required for the rest of the work. Section III presents the dataset for validation of the estimation algorithm, after which, in Section IV, chunk requests are analyzed in detail. Based on this, traffic patterns are presented and linked to application situations. Section V uses the knowledge from Section IV to derive an estimator for video stalls. Afterwards, the evaluation takes place in Section VI. We conclude the work with a comparative related work in Section VII and a final conclusion in Section VIII.

II. Background

This section contains substantial definitions and essential streaming background information. The technical terminology of streaming-related terms is established followed by a differentiation of streaming at application layer and network layer.

Adaptive Video Streaming: In the past, when a user requested a video stream, it was downloaded in a single quality independent of the available bandwidth in a best effort manner. However, if the average download rate is insufficient, the stream can not be played without stalling. To accommodate the average playback rate to the available download rate, HTTP adaptive streaming (HAS) was introduced. HAS selects the video encoding bitrate to match the downlink bandwidth. This process, standardized as DASH [11], allows continuous streaming with a stable buffer in the highest possible quality used by, among others, YouTube and Netflix [4].

The streaming process with the YouTube android app in detail is as follows: An end-user requests video content from Google servers by YouTube, exemplary available in three qualities, in Figure 1. Since audio in YouTube is available with constant bitrate (CBR), only one quality level, AQ1, is available. The video player sends requests for content via the HTTP protocol and opens or re-uses existing encrypted TCP connections. Based on the requests and downloaded data, a video session can be reconstructed by estimating parameters like downloaded video playback, initial delay, and stalling. The detailed process of video flow and chunk request separation is described in [12]. By separating video flows based on IP-address port combinations it is also possible to distinguish between different videos, and thus different users at the same time.

Chunks: From a network layer point of view, only the requests for video and audio data of the stream, called chunks, are contained in the uplink. The requested data is in the downlink back to the end-user's device. In typical video streaming, the video and audio chunks can have up to a specific amount of data in byte. Video chunks are usually much larger than audio chunks. The size of video chunks is defined by the currently available bandwidth and the requested quality level.

Considering YouTube, there is no parallel download of chunks of the same type, although video and audio can be downloaded in parallel. This behavior is detected in our measurements for QUIC and TCP streaming. In case of TCP, two different TCP ports are used for audio and video if downloading in parallel.

Segments: In contrast to chunks, segments are the application layer portion of streaming data. Segments are separated by the duration while chunks by the size in byte. It is common to request multiple segments at network layer, usually one or two, in one chunk. For the application layer, this means that once the part of the chunk that covers an entire segment has been downloaded, this segment can be played out. Furthermore, quality changes can only be performed after a segment is downloaded completely. Typically, a video segment in YouTube contains roughly 5 s of video data, varying depending on the format. An audio segment has about 10 s.

Video Buffer: In the video buffer, several seconds of playback can be stored. After the playback starts, it is essential that the video buffer never runs empty, otherwise the playback stalls. If the buffer is full, no more video data is requested. If the buffer level decreases below a specific threshold, a lower quality is requested to adapt the video rate against the changing downlink rate to ensure smooth playback. Additionally, the usage of a buffer helps to prevent the unnecessary download of video data. This is required, since the video playback is often aborted by the user before it is played out completely.

A study in 2011 [13] showed that about 60 % of all requested videos in YouTube were watched for less than 20 % of their duration. Thus, prebuffering the complete video wastes much bandwidth. For that reason, the appropriate buffer size must be chosen according to three factors: the tolerable amount of data that can be thrown away when a user aborts; the maximum variation in end-to-end network download rate to avoid running empty; the variations in the video encoding rate. Compared to on-demand streaming, in live streaming the buffer is also used as parameter for allowed delay. A study about a proper video buffer size is presented in [14].

The mobile YouTube player has an audio and a video buffer. Each arriving chunk is split in its segments and added to the associated buffer. Regarding Figure 1, video chunk 1 contains two segments, played out after each other. The different colors show the available qualities. Audio chunk 1 contains one segment that is added to the audio buffer shown in blue.

III. Data Set

In this section, all measured data is summarized. The testbed for data collection is placed in two large cities in two different countries according to [15]. In total more than 900 h of video data are collected. First, for ground truth determination and testbed validation, 2464 measurement runs corresponding to more than 450 h of video playback time were recorded, 374 h are already published in [16]. For a better reproducibility of the results, an overview of all network layer data and network layer data flows is created and available in [17]. The focus of the first part of the dataset was correctness, usability for the estimation, validation purpose and plausibility check. Therefore, chunk request extraction, video flow determination, and QoE ground truth determination including quality changes and stalling is done according to [18]. The second focus is based on the geographical distribution of both peering points for measurement validation. The focus of the second dataset in [17] is the presentation of postprocessed data for reproducibility of the results and easier usage in the research community.

Data Set Overview: For the estimation of stalling, additionally a total of 2618 valid measurement runs equal to more than 475 h of video data were captured between April and February 2019. During playback, 877 of all video runs stalled. According to Table I, 18 different videos were measured. The duration of the videos is between 2.27 min and more than 9 h. The goal in the video selection was to cover a large set of different videos concerning duration, video topic, scene- and

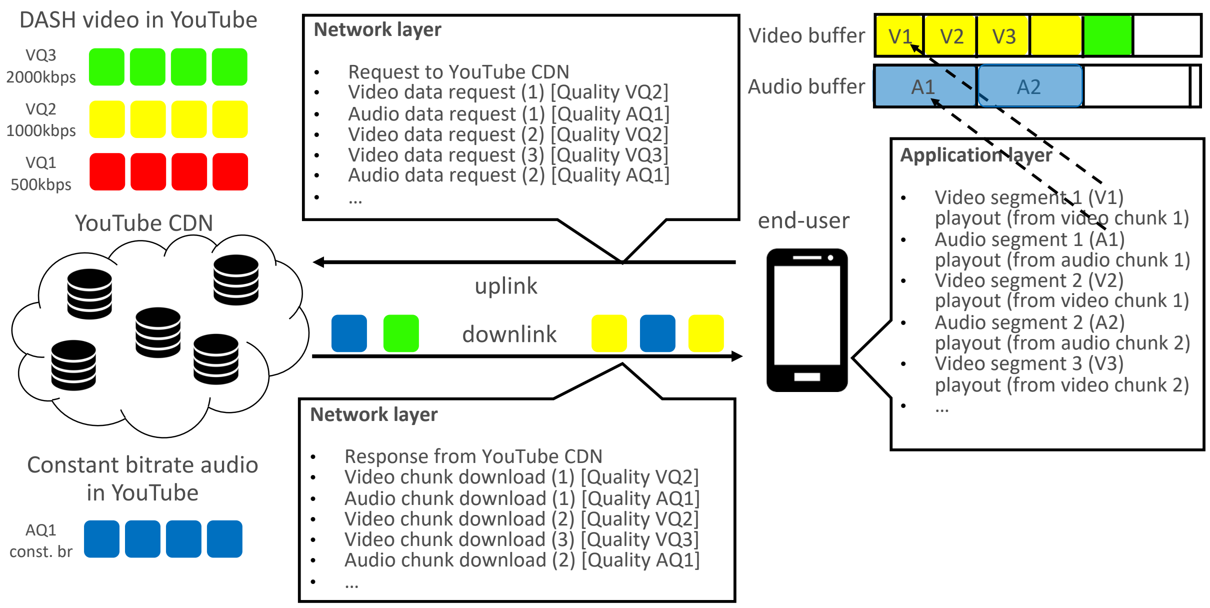
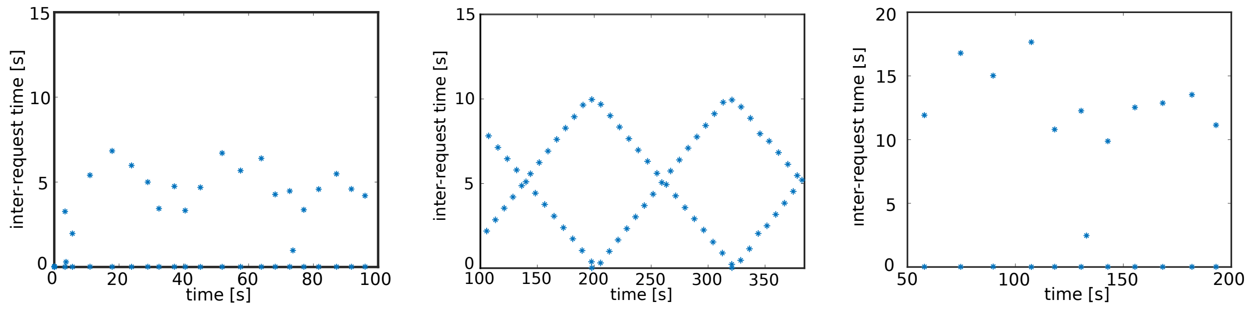

DASH video in YouTube
VQ3 2000kbps
VQ2 1000kbps
VQ1 500kbps
YouTube CDN
Constant bitrate audio in YouTube
AQ1 const. br
Network layer
Request to YouTube CDN
Video data request (1) [Quality VQ2]
Audio data request (1) [Quality AQ1]
Video data request (2) [Quality VQ2]
Video data request (3) [Quality VQ3]
Audio data request (2) [Quality AQ1]
...
uplink
downlink
Network layer
Response from YouTube CDN
Video chunk download (1) [Quality VQ2]
Audio chunk download (1) [Quality AQ1]
Video chunk download (2) [Quality VQ2]
Video chunk download (3) [Quality VQ3]
Audio chunk download (2) [Quality AQ1]
...
Video buffer
V1
V2
V3
Audio buffer
A1
A2
end-user
Application layer
Video segment 1 (V1) playout (from video chunk 1)
Audio segment 1 (A1) playout (from audio chunk 1)
Video segment 2 (V2) playout (from video chunk 1)
Audio segment 2 (A2) playout (from audio chunk 2)
Video segment 3 (V3) playout (from video chunk 2)
...

Fig. 1: YouTube streaming behavior at network and application layer



(a) Low IRT pattern (b) Repeating pattern (c) High IRT pattern

Fig. 2: Detected main traffic patterns

camera perspective changes resulting in different bitrates, and frames per second.

Furthermore, a diversity of representative streaming situations are measured and evaluated listed in Table II. For scenarios 22 to 56, each of the listed qualities are measured with each of the presented throughput limits. Additionally, the 4G bandwidth traces collected in [19] and the 3G traces of [20] are set as throughput limits to measure realistic scenarios. The other scenarios are chosen as follows:

Scenario 1 has a fixed bandwidth of 5000 kbit/s with a changing quality from 720p to 360p and back every 60 s. The goal of this scenario is to trigger quality changes and detect differences to stalling request patterns. Scenarios 2 - 6 are defined with an automated quality setup and slowly decreasing bandwidths in different time intervals. This is done to trigger quality changes by throttling the bandwidth stepwise. Scenarios 7 - 9 simulate network outages. Scenarios 10 and 11 are characterized by slowly decreasing bandwidth limits. Compared to previous scenarios, the limit nearly drops to zero to trigger stalling scenarios or decrease the quality to 144p. In scenario 11, the rate increases again after the outage. Lastly, scenarios 12 - 21 describe an automatic quality setting with different bandwidth limits without stallings. Scenarios 22 - 56 use different quality settings with different bandwidth limitation for validation. In total, we cover a wide set of typical scenarios in mobile video streaming including representative cases with stalling occurrences during video playback and realistic scenarios of real 3G and 4G bandwidth traces.

IV. Traffic Patterns

Analyzing all captured data, three different traffic patterns are detected summarized in Figure 2. Traffic patterns are the

TABLE I: Video overview

Index	Type	Video ID	Duration [s]	Index	Type	Video ID	Duration [s]
V1	T1	$2d1VrCvdzbY$	561	V10	T1	$14GFLpsV068$	147
V2	T2	$N2sCbtodGMI$	559	V11	T1	$4\_9PhJ0Ao7E$	144
V3	T1	$OHOpb2fS-cM$	734	V12	T1	$1ixCkNCLfA4$	32609
V4	T1	$0y2qOYemiLs$	537	V13	T1	$3NttMN0z\_Iw$	223
V5	T1	$0mrfImiI0Q8$	425	V14	T1	$1dLuiA4T-Dc$	2316
V6	T1	$1fhdHPBqUa8$	653	V15	T2	$2ypwI56JGLM$	329
V7	T1	$3iube-nnsr8$	1615	V16	T2	$2\_jdPTfbngs$	364
V8	T1	$3ogzQjdSArw$	234	V17	T2	$3hp7xwJHEko$	211
V9	T1	$12rp64JoY6A$	418	V18	T2	$1e0gvDiPRdk$	4306

TABLE II: Scenario overview

Scenario	Quality	Throughput limit (Mbit/s)
1	720p $\overset{60\,\mathrm{s}}{\underset{60\,\mathrm{s}}{\rightleftarrows}}$ 360p	5
2	auto	$5 \overset{10\,\mathrm{s}}{\rightarrow} 3 \overset{5\,\mathrm{s}}{\rightarrow} 2.5 \overset{5\,\mathrm{s}}{\rightarrow} 2 \overset{5\,\mathrm{s}}{\rightarrow} 1.5 \overset{5\,\mathrm{s}}{\rightarrow} 1$
3	auto	$3 \overset{60\,\mathrm{s}}{\rightarrow} 2.5 \overset{5\,\mathrm{s}}{\rightarrow} 2 \overset{5\,\mathrm{s}}{\rightarrow} 1.5 \overset{5\,\mathrm{s}}{\rightarrow} 1 \overset{20\,\mathrm{s}}{\rightarrow} 0.8 \overset{5\,\mathrm{s}}{\rightarrow} 0.7 \overset{5\,\mathrm{s}}{\rightarrow} 0.5$
4	auto	$3 \overset{30\,\mathrm{s}}{\rightarrow} 2.5 \overset{5\,\mathrm{s}}{\rightarrow} 2 \overset{5\,\mathrm{s}}{\rightarrow} 1.5 \overset{5\,\mathrm{s}}{\rightarrow} 1 \overset{20\,\mathrm{s}}{\rightarrow} 0.8 \overset{5\,\mathrm{s}}{\rightarrow} 0.6 \overset{5\,\mathrm{s}}{\rightarrow} 0.3$
$\{5,6\}$	auto	$2.5 \overset{\{60,30\}\,\mathrm{s}}{\rightarrow} 2 \overset{5\,\mathrm{s}}{\rightarrow} 1.5$
$\{7,8\}$	auto	$2.5 \overset{45\,\mathrm{s}}{\rightarrow} 0.05 \overset{\{30,15\}\,\mathrm{s}}{\rightarrow} 0.3$
9	auto	$2.5 \overset{45\,\mathrm{s}}{\rightarrow} 0.1 \overset{15\,\mathrm{s}}{\rightarrow} 0.3$
10	auto	$2.5 \overset{45\,\mathrm{s}}{\rightarrow} 2 \overset{10\,\mathrm{s}}{\rightarrow} 1.5 \overset{10\,\mathrm{s}}{\rightarrow} 1 \overset{15\,\mathrm{s}}{\rightarrow} 0.5 \overset{15\,\mathrm{s}}{\rightarrow} 0.25$
11	auto	$2.5 \overset{20\,\mathrm{s}}{\rightarrow} 1.5 \overset{5\,\mathrm{s}}{\rightarrow} 1 \overset{5\,\mathrm{s}}{\rightarrow} 0.5 \overset{5\,\mathrm{s}}{\rightarrow} 0.2 \overset{5\,\mathrm{s}}{\rightarrow} 0.1 \overset{5\,\mathrm{s}}{\rightarrow} 0.05 \overset{60\,\mathrm{s}}{\rightarrow} 2.5$
12-21	auto	$\{0.5 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 8 \mid 10 \mid 25 \mid 400\}$
22-56	$\{144p \mid 240p \mid 360p \mid 480p \mid 720p\}$	$\{1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 25 \mid 400\}$

mapping of the video behavior at application layer to network traffic. A pattern is a sequence of network packets corresponding to the requested network chunks. The understanding of the traffic patterns forms the basis for the stalling estimator in this document. In the following, the detected traffic patterns are presented in detail in Figure 2 based on an example measurement and afterwards mapped to streaming phases. All subfigures present the inter-request time (IRT) at the y-axis and the measurement time of the example run at the x-axis.

Chunk Request Traffic Patterns: Figure 2a shows an irregular pattern. Most of the IRTs are close to 0 s or distributed around 5 s. The maximum IRT is about 7 s, the mean is 2.13 s. Thus, the chunks are requested very frequently without high delay. Figure 2b shows a repeating pattern with a mean IRT of 4.89 s. The time between two consecutive requests is 5 s at 120 s of the measurement snippet, followed by an alternating pattern between smaller and larger values. At 200 s the IRT between two values is 10 s. Afterwards, the duration is alternating back to 5 s at 260 s and the pattern is repeated. Last, Figure 2c shows a more irregular pattern with a mean IRT of 6.64 s. Between most requests, there is an IRT of more than 10 s or a value close to 0 s.

Audio and Video Request Separation: The traffic patterns presented above represent video and audio chunk requests. For an accurate stalling estimation it is essential to separate the video from the audio stream. This is mainly due to three reasons: First, the video buffer is the primary buffer. For stalling estimation, it is enough to consider the video stream since the YouTube player will not stream if the video buffer is empty. Second, the buffer level is estimated wrong for a complete chunk size if a chunk is classified falsely as audio or video. Last, transition detection between video phases presented above is more accurate if the video and audio stream is separated.

Since audio in YouTube is streamed with a CBR, each audio chunk is about the same size. For that reason, it is sent in the same amount of packets, to our findings in 116, 117, or 118 packets. To prove this, Table III shows statistics about the packets of each chunk for all measured data. The first column shows the chunk type or the quality level, the second one the mean amount of packets for each chunk of that specific quality. The third column shows the mean absolute deviation (MAD) and the last column the amount of video chunks received for each quality. The presented statistics show an increasing variance in the amount of packets for each video chunk with increasing video quality. Comparing 144p video with 720p, the MAD increases from 3.81 to more than 230. Compared to that there is only small variance in the amount of packets for each audio chunk. Based on these findings, audio and video chunks of the steady phase of Figure 2b are separated exemplary in

TABLE III: Packets per chunk statistics

Quality	mean	MAD	# chunks
audio	117.07	0.13	8774
144p	89.71	3.81	2877
240p	172.47	20.59	679
360p	308.57	49.63	743
480p	533.23	108.60	708
720p	1080.00	230.63	2092

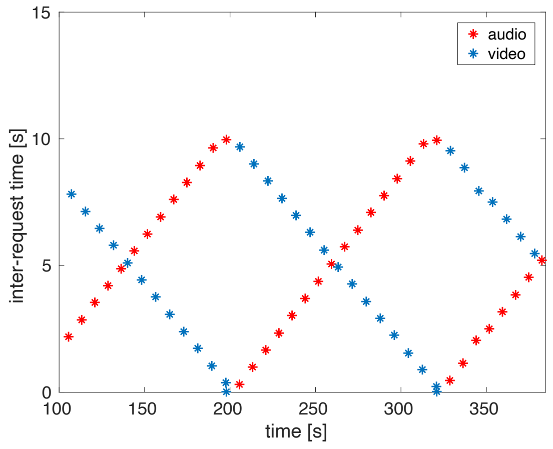


Fig. 3: Audio and video requests

Figure 3 while all chunks sent in 116, 117, or 118 packets are classified as audio during our measurements. This value results from the CBR in audio and thus, segments of the same size in byte transmitted in the same amount of packets. The amount of packets is determined by the maximum transmission unit (MTU) size that is 1500 B.

Relation between Traffic Patterns and Video Streaming Phases: After separating audio and video, the traffic patterns can be mapped to streaming phases that are states of the internal streaming process. From an application point of view, three phases in video streaming can be defined according to [21]: an initial buffering phase, a periodic buffer refill phase and a buffer depletion phase. In the initial buffering phase or filling phase, video data is requested and delivered in a best effort manner until the target buffer threshold is reached. A small IRT like shown in Figure 2a is a typical reference for this phase. When the buffer is filled completely, the player switches to the periodic buffer refill phase or steady state phase. It corresponds to the normal streaming behavior, where as much data is downloaded as taken out from the buffer for playback. This results in a very periodic uplink request pattern like shown in Figure 2b. The reason for this pattern is the concatenation of video and audio chunk requests with slightly different but almost fixed intervals. In the buffer depletion phase, the buffer level decreases. The uplink request pattern can be very irregular or have a high IRT pattern, depending

Algorithm 1: Stalling estimation algorithm

```
Data: $arr_{diff}$: array with inter-request times of all video
        chunk requests,
      $r$: requesting threshold [s],
      $B$: target buffer size [s],
      $U$: remaining buffer level [s],
      $Z$: video chunk duration [s]
      $f = 0$: prestall flag
1  foreach $j \in arr_{diff}$ do
2      if $j > r$ and $f = 0$ then
3          $f = 1$,
4          $U = U - j + Z$,
5          if $U \leq 0$ then
6              add $pos_{curr} + U$ to stalling event positions
7      if $f = 1$ then
8          $U = U - j + Z$,
9          if $U > B$ then
10             $f = 0$
11         if $U \leq 0$ then
12             add $pos_{curr} + U$ to stalling event positions
13 return stalling event positions;
```

on the currently available bandwidth, c.f. Figure 2c.

V. Stalling Estimator Algorithm

Based on the findings presented above, this section describes a network layer stalling estimation approach based on chunk requests in encrypted network traffic. The variables required for the stalling estimation approach presented in Algorithm 1 are defined as follows: the requesting threshold r is used for detecting the buffer depletion phase start. If no request is received for r seconds, the player is assumed to be in the depletion phase. The target buffer level B is a specific buffer threshold. If the remaining buffer U drops below B, new data is requested. Z describes the amount of video seconds received with one chunk download. f is a boolean variable which is set when the player is in the buffer depletion phase.

Practical Implementation: Algorithm 1 shows the pseudo-code for the stalling estimation approach. The input arguments are the IRTs of video chunk requests in the uplink (arr_{diff}) and the variables defined above. All parameters can be set to fixed values at the beginning of the estimation or variable, based on the current video playback position. (*Note:* since the stalling algorithm works independently of the used player, the parameters must be defined for each player and adaptation algorithm with preliminary measurements.) The buffer level U is assumed to be equal to B, if the player is in the steady state phase before the buffer depleting phase is reached. For data cleaning, all empty chunk requests are deleted indicating server changes and empty requests.

For the stalling algorithm, all chunk request IRTs are compared to the requesting threshold r to check if the player enters the buffer depleting phase. If $f = 0$, and the IRT is larger than the requesting threshold r, f is set to 1 in line 3. Then, the player is assumed to be in the buffer depletion phase. The remaining buffer level U is decreased by the IRT j while increased by the next chunk duration Z. If U drops to zero, an outrunning buffer is estimated and the player is assumed to stall. Thus, at $pos_{curr} + U$, a stalling event is estimated, while pos_{curr} describes the current video position. Since events are only triggered when a new chunk is requested, U might become a negative value that has to be added to the current position. If $f = 1$ each IRT j during the buffer depletion phase is taken into consideration. In this situation, either U increases until the buffer is filled again completely and $f = 0$ described in line 9 and 10 of the algorithm. If the buffer depletes the player stalls according to line 11 and 12.

VI. Evaluation

The evaluation presented in this section is threefold. First, results of a video session study are shown followed by the presentation of the stalling detection accuracy with the approach described in Algorithm 1. At the end of this section, the estimator performance is discussed.

Video Session Study: For the video session study, three main observations are done: the average chunk size in different streaming phases, the error made while tracking the complete video, and the buffer state B during the steady phase. Therefore, the average chunk size in the filling, steady, and depleting phase is analyzed. Two different video types are detected regarding downloaded video playback time in one chunk, listed in Table I. For simplicity and presentation overview purpose, only videos V1, V2, and V3 are depicted. The other videos behaved similar according to the listed types.

The average chunk duration for video V1 representing the first type and video V2 representing the second type is presented in Figure 4. All lines in the figure show the average chunk duration during a specific phase as a cumulative distribution function (CDF). Type 1 requests about two segments with each chunk in the steady phase shown by the solid black line. For V1, each segment contains 5.34 s of video while the average requested chunk duration is 10.46 s. The difference between the duration of two segments and the chunk duration can be explained by requesting smaller chunks at the end of the video and before a bandwidth degradation. There, a complete segment can not be downloaded anymore. The dashed black line shows the chunk duration in the buffer filling phase with 11.33 s on average. Thus, to fill the buffer two or more segments are requested in one chunk. The dotted black line shows the chunk duration in the buffer depletion phase with an average value of 9.42 s representing the download of one, two, or three segments in a chunk. This is caused by different bandwidths and quality levels requested in this phase.

Regarding the second type, shown in orange and represented by video V2, mainly one segment is requested by one chunk in the steady phase. The average playback time requested is 5.42 s for V2, while the segment duration is 5.12 s. Thus, sometimes more than one segment is requested. With 10.11 s on average, a larger amount is requested in the filling phase. The results for the depletion phase is presented with the dotted line. In about 40 % on average about half a segment, in close to 60 % 1.5 - 2 segments are requested by a single chunk. The average value is 6.61 s. Requesting less than one segment is possible, for example, if only the duration between two key-frames is requested.

Based on the amount of chunk requests in each streaming phase and the average video playback time downloaded by one chunk, the video duration can be estimated according to Figure 5. The x-axis shows the tracking error, while 0 indicates no deviation between estimation and real video duration. It is shown, that for all three exemplary videos, more than 60 % are captured with less than 5 % deviation from the real value. More than 90 % of all runs are tracked with less then 10 % error in positive or negative direction. The negative deviation can be described by wrongly classified chunks. This can occur for small qualities, when the amount of packets for one video segment is the same than for one audio segment. Additionally, one reason is the large variance in chunk durations in the buffer filling and depleting phase. Another reason for the overestimation is not played out video before quality changes in case of bandwidth degradations.

Additionally, an accurate buffer size determination is required for stalling estimation. Figure 6 shows the CDF of the average buffer size in the steady phase for all runs of V1,V2, and V3. The buffer size is between 119.5 s and 125 s for all videos and all runs, independent on the defined type. In the steady phase, the variance of the overall buffer size is slightly larger. Figure 7 shows the buffer size in the steady phase as a box plot for V4, V5, and V6 presenting T1 and V15, V16, and V17 for T2. It is to mention that similar results are detected for other videos of the same type. The variance in the buffer size is larger for T1 than T2, explained by the higher chunk duration in the steady phase according to Figure 4. Furthermore, the buffer is always kept above 120 s.

Estimation Accuracy: Based on the findings above, Algorithm 1 is used for stalling estimation in the presented data set. The requesting threshold for detecting the buffer depletion phase is set to 17.9 s exposed as the best value during the evaluation. For the chunk duration Z of video T1, 10.5 s is used covering the average chunk duration best. For type two, 6 s is used. At the beginning, the buffer level is tracked until 120 s are reached. If the buffer depletion phase is reached before the buffer is filled completely, the estimated buffer level is used as value U for the algorithm, otherwise $U = B = 120s$. Additionally, when the buffer depletion phase is reached, the real buffer level is measured.

The stalling estimation result for the estimated buffer [EB] and the measured buffer [MB] for different videos, scenarios, and overall is summarized in Table IV. The first column shows the constrains on the dataset. The first four lines show different videos. V3 is used for video type T1, V2 for T2. These videos are selected to show the correct detection but also outliers, other videos of the same type show similar results. The second column shows correct stalling detection called

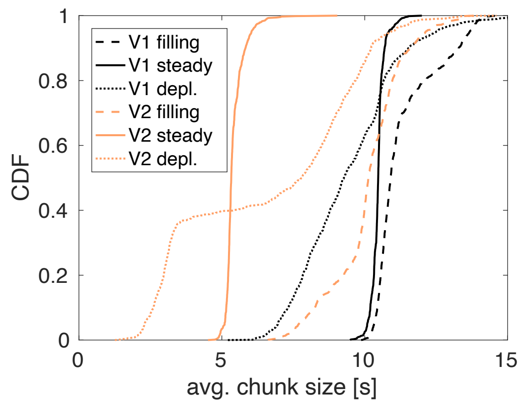


Fig. 4: Avg. chunk size comparison

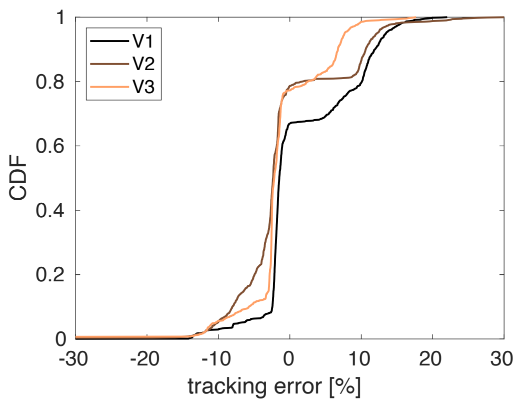


Fig. 5: Video duration tracking error

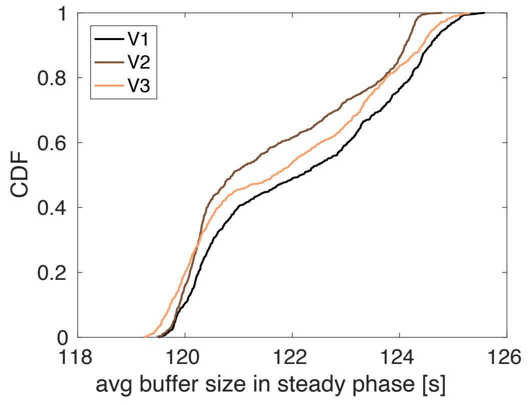


Fig. 6: Avg. buffer steady

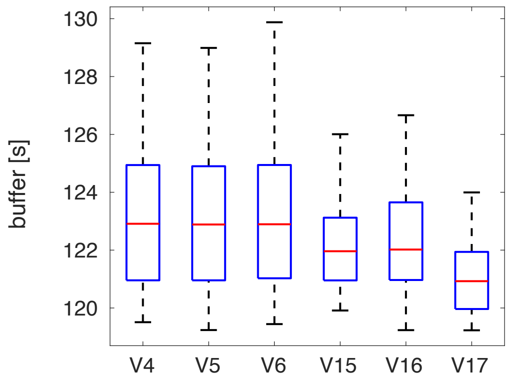


Fig. 7: Overall buffer steady

true positive rate. The third one presents video measurements characterized falsely as stalling named *false positive rate*. The fourth column shows videos that did not stall, classified correct as *true negative rate*. The last column lists the percentages that videos did not stall classified as stalling named *false negative rate*.

The table shows that the *true positive rate* is more than 90 % for all scenarios. The exact determination of the buffer shows a higher true positive rate than estimating it. The lowest value with 91.58 % for the *true positive rate* is received in slowly decreasing bandwidth scenarios. There, the player resides in the buffer depletion phase very long where the variance in chunk durations is high. Compared to that, the *true negative rate* is received with 83.81 % to 97.22 %. The highest value is determined for video V2. There, less video seconds are requested with one chunk. Thus classifying one chunk wrongly has less influence on the difference between estimated and real remaining buffer. The different phenomena are shown for video V3, where 13.5 % *false negative rate* is received. Especially the exact knowledge of the buffer size at the beginning of the buffer depletion phase has a large influence on the received result for this video type. If the buffer level is overestimated in this case, short stallings can not be detected. The same is visible for slowly decreasing buffer scenarios. Regarding the overall correct detection, an estimation true positive rate of more than 90 % for all scenarios is measured. From a QoE perspective, the correct stalling detection is of high relevance, done with more than 93 % with the presented approach. In scenarios with obvious stalling (scenario 7 - 9 and 11) and obvious no stalling (scenario 1, high bandwidth scenarios) all valid runs are classified correctly. But to react on changing network conditions and avoid stallings, buffer depletion phases must be detected, that was possible with the algorithm for all measurement runs. Thus, issues at the network layer can be identified in advance, and as soon as the issue is detected at network layer, additional bandwidth can be provisioned before stalling occurs. This avoids stalling as the most severe QoE degradation factor.

Estimator Performance: The estimator is capable of a worst case prediction. By adding the buffer level U to the current video position when reaching the buffer depletion phase, the earliest possible stalling position can be detected.

TABLE IV: Stalling estimation accuracy [EB: est. buffer; MB: measured buffer]

	true positive rate	**false positive rate**	**true negative rate**	**false negative rate**
V3 EB	94.12	5.88	86.49	13.50
V3 MB	94.80	5.20	96.54	3.46
V2 EB	92.88	7.12	96.47	3.53
V2 MB	94.80	5.20	97.22	2.78
bw slowly dec. EB	91.58	8.42	83.81	16.19
bw slowly dec. MB	93.77	6.23	94.26	5.74
overall EB	93.16	6.84	90.75	9.25
overall MB	93.61	6.39	96.44	3.56

This prediction can be extended to a real time observation if receiving additional video chunks during the buffer depleting phase. Then, for each request during this phase, one additional video chunk duration Z is added to the currently tracked buffer. If the calculated buffer runs empty, the player is assumed to stall. The accuracy of the worst case prediction and the real time observation method depends on the accurate buffer tracking and the chunk size Z.

VII. Related Work

A deep understanding of parameters influencing the QoE is required for a detailed analysis. A comprehensive overview is given by Seufert et. al. [7]. Furthermore, stalling is detected as the parameter influencing the QoE in the most negative way [8]. For that reason, a diversity of techniques quantifying QoE in video streaming are in literature. Hoßfeld et. al. uses crowdsourcing to quantify the QoE by subjective studies [22], [23].

Especially for YouTube, several methods and monitoring applications exist to receive data for QoE determination [24], [25], [26]. Nevertheless, it is essential to not only monitor application layer parameters. The network traffic for requesting and receiving a video from YouTube for a widespread understanding of the streaming process is observed in [27]. Additionally, with the increasing amount of traffic encryption in the Internet, new methods arise to detect video flows on transport layer [28], [29] or by machine learning approaches [30], [31] and classify video QoE from encrypted traffic [32], [33], [34], [35]. Especially in transport layer monitoring approaches, it is important to keep track of the complete downlink traffic by monitoring throughput, packet inter-request times, or packet sizes.

Compared to current literature, we detect video flows, classify them as video or audio and draw conclusion about the complete video session with an easy and accurate approach. Based on this session reconstruction process, it is possible for us to estimate QoE parameters on network layer with a very lightweight measuring instance. The main advantage of this approach compared to other literature like [36] or [37] is, that almost only uplink traffic has to be taken into consideration. For a 10 minutes video, only about 115 uplink request packets must be analyzed in this approach, independent on the playback quality. With state-of-the art approaches, using all packets in downlink direction for stalling prediction, about 20.000 packets are analyzed for the same video in 144p quality. For a 720p video, it is more than 100.000 packets. After detecting the requesting pattern in the steady state phase, the approach is independent of downlink traffic monitoring. In case of insufficient bandwidth or a not completely filled buffer, it is only necessary to distinguish between audio and video according to the amount of packets per request. The average chunk size can be determined in advance. No modification to the native YouTube app is required that is, to the best of our knowledge, not defined in this detail in current literature.

VIII. Conclusion

Based on a large amount of video data collected with the native YouTube mobile app, this paper presents a lightweight approach to detect stalling events from encrypted network traffic. By analyzing the characteristics of HTTP-adaptive streaming in detail, different application parameters are extracted from network traffic. First, video flows are detected and audio and video requests are separated. By analyzing the received requests, three different chunk request patterns are observed and mapped to downloaded playback time. Second, based on these information, a real time buffer tracking approach is presented used to detect video stalls. The results show correct stalling detection over all scenarios for more than 90 % of all video runs. Furthermore, this simple approach shows the possibility to detect changing streaming conditions at network layer early by only monitoring the uplink request pattern. For that reason, a network operator or cloud streaming provider has time to react on a buffer depletion situation. At the end, further investigation of the worst case stalling prediction are pointed out.

References

[1] Sandvine. (2018) The global internet phenomena report. Accessed: 2018-10-17. [Online]. Available: https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf

[2] YouTube. (2018) YouTube for press - statistics. Accessed: 2018-10-17. [Online]. Available: https://www.youtube.com/intl/en-GB/yt/about/press/

[3] Cisco. (2017) Cisco visual networking index: Forecast and methodology. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf. Accessed: 2018-10-17.

[4] Bitmovin. (2018) Why YouTube and Netflix use MPEG-DASH. Accessed: 2018-10-17. [Online]. Available: https://bitmovin.com/status-mpeg-dash-today-youtube-netflix-use-html5-beyond/

[5] C. Moldovan, F. Metzger, S. Surminski, T. Hoßfeld, and V. Burger, "Viability of wi-fi caches in an era of HTTPS prevalence," in *IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2017.

[6] F. Wamser, P. Casas, M. Seufert, C. Moldovan, P. Tran-Gia, and T. Hossfeld, "Modeling the YouTube stack: From packets to quality of experience," *Computer Networks*, 2016.

[7] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP adaptive streaming," *IEEE Communications Surveys & Tutorials*, 2015.

[8] T. Hoßfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, "Initial delay vs. interruptions: Between the devil and the deep blue sea," in *Fourth International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 2012.

[9] C. Moldovan and F. Metzger, "Bridging the gap between QoE and user engagement in HTTP video streaming," in *28th International Teletraffic Congress (ITC 28),*. IEEE, 2016.

[10] M. Fiedler, K. De Moor, H. Ravuri, P. Tanneedi, and M. Chandiri, "Users on the move: On relationships between QoE ratings, data volumes and intentions to churn," in *2017 IEEE 42nd Conference on Local Computer Networks Workshops (LCN Workshops), Singapore*. IEEE, 2017.

[11] I. O. for Standardization. (2014) Iso/iec 23009-1:2014. Accessed: 2018-08-31. [Online]. Available: https://www.iso.org/standard/65274.html

[12] F. Loh, F. Wamser, C. Moldovan, B. Zeidler, T. Hoßfeld, D. Tsilimantos, and S. Valentin, "From click to playback: a dataset to study the response time of mobile YouTube," in *Proceedings of the 10th ACM Multimedia Systems Conference*. ACM, 2019.

[13] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "YouTube everywhere: Impact of device and infrastructure synergies on user experience," in *Proceedings of the 2011 ACM SIGCOMM conference*. ACM, 2011.

[14] T. Hoßfeld, C. Moldovan, and C. Schwartz, "To each according to his needs: Dimensioning video buffer for specific user profiles and behavior," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015.

[15] T. Karagkioules, D. Tsilimantos, S. Valentin, F. Wamser, B. Zeidler, M. Seufert, F. Loh, and P. Tran-Gia, "A public dataset for YouTube's mobile streaming client," pp. 1–6, 2018.

[16] ——. (2018) Dataset: A public dataset for YouTube's mobile streaming client. Open Dataset. [Online]. Available: http://qoecube.informatik.uni-wuerzburg.de/

[17] F. Loh, F. Wamser, C. Moldovan, B. Zeidler, T. Hoßfeld, D. Tsilimantos, and S. Valentin. (2019) From click to playback: a dataset to study the response time of mobile YouTube. Open Dataset. [Online]. Available: https://go.uniwue.de/initialdelaydataset

[18] M. Seufert, B. Zeidler, F. Wamser, T. Karagkioules, D. Tsilimantos, F. Loh, P. Tran-Gia, and S. Valentin, "A wrapper for automatic measurements with YouTube's native android app," in *2018 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 2018.

[19] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks," *IEEE Communications Letters*, 2016.

[20] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3G networks: analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*. ACM, 2013.

[21] D. Tsilimantos, T. Karagkioules, A. Nogales-Gómez, and S. Valentin, "Traffic profiling for mobile video streaming," in *IEEE International Conference on Communications (ICC)*. IEEE, 2017.

[22] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of YouTube QoE via crowdsourcing," in *IEEE International Symposium on Multimedia (ISM)*. IEEE, 2011.

[23] T. Hossfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, and P. Tran-Gia, "Best practices for QoE crowdtesting: QoE assessment with crowdsourcing," *IEEE Transactions on Multimedia*, 2014.

[24] P. Casas, M. Seufert, and R. Schatz, "Youqmon: a system for on-line monitoring of YouTube QoE in operational 3G networks," *ACM SIGMETRICS Performance Evaluation Review*, 2013.

[25] B. Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle, "Yomo: a YouTube application comfort monitoring tool," *New Dimensions in the Assessment and Support of Quality of Experience for Multimedia Applications, Tampere, Finland*, 2010.

[26] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz, "Yomoapp: A tool for analyzing QoE of YouTube HTTP adaptive streaming in mobile networks," in *2015 European Conference on Networks and Communications (EuCNC)*. IEEE, 2015.

[27] 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.

[28] F. Li, J. W. Chung, and M. Claypool, "Silhouette: Identifying YouTube video flows from encrypted traffic," in *Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2018.

[29] D. Tsilimantos, T. Karagkioules, and S. Valentin, "Classifying flows and buffer state for YouTube's HTTP adaptive streaming service in mobile networks," in *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM, 2018.

[30] J. Khalife, A. Hajjar, and J. Díaz-Verdejo, "Performance of opendpi in identifying sampled network traffic," *Journal of Networks*, 2013.

[31] T. T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and continuous machine-learning-based classification for interactive IP traffic," *IEEE/ACM Transactions On Networking*, 2012.

[32] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "emimic: Estimating HTTP-based video QoE metrics from encrypted network traffic," *Network Traffic Measurement and Analysis Conference (TMA)*, 2017.

[33] R. Dubin, A. Dvir, O. Pele, and O. Hadar, "I know what you saw last minute-encrypted HTTP adaptive video streaming title classification," *IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2017.

[34] W. Pan, G. Cheng, H. Wu, and Y. Tang, "Towards QoE assessment of encrypted YouTube adaptive video streaming in mobile networks," in *IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. IEEE, 2016.

[35] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "A machine learning approach to classifying YouTube QoE based on encrypted network traffic," *Multimedia tools and applications*, 2017.

[36] V. Krishnamoorthi, N. Carlsson, E. Halepovic, and E. Petajan, "Buffest: Predicting buffer conditions and real-time requirements of HTTP (S) adaptive streaming clients," in *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 2017.

[37] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for HTTPS and QUIC," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018.