Received 15 July 2018; | Revised 27 September 2018; | Accepted -

DOI: xxx/xxxx

JOURNAL ARTICLE

Dissecting the Performance of YouTube Video Streaming in Mobile Networks

Anika Schwind[1] | Cise Midoglu[2] | Özgü Alay[2,3] | Carsten Griwodz[2,4] | Florian Wamser[1]

[1]Chair of Communication Networks, University of Würzburg, Würzburg, Germany
[2]Simula Research Laboratory, Oslo, Norway
[3]Simula Metropolitan Center for Digital Engineering, Oslo, Norway
[4]University of Oslo, Oslo, Norway

Correspondence
Anika Schwind
Email: anika.schwind@informatik.uni-wuerzburg.de
Cise Midoglu
Email: cise@simula.no
Both authors contributed equally to this work.

Summary

Video streaming applications constitute a significant portion of the Internet traffic today, with mobile accounting for more than half of the online video views. The high share of video in the current Internet traffic mix has prompted many studies that examine video streaming through measurements. However, streaming performance depends on many different factors at different layers of the TCP/IP stack. For example, browser selection at the application layer or the choice of protocol in transport layer can have significant impact on the video performance. Furthermore, video performance heavily depends on the underlying network conditions (e.g., network and link layers). For mobile networks, the conditions vary significantly, since each operator has a different deployment strategy and configuration. In this paper, we focus on YouTube and carry out a comprehensive study investigating the influence of different factors on streaming performance. Leveraging the MONROE testbed that enables experimentation with 13 different network configurations in 4 countries, we collect more than 1,800 measurement samples in operational mobile networks. With this campaign, our goal is to quantify the impact of parameters from different layers on YouTube's streaming QoE. More specifically, we analyze the role of the browser (e.g., Firefox and Chrome), the impact of transport protocol (e.g., TCP or QUIC), the influence of network bandwidth, and signal coverage on streaming QoE. Our analysis reveals that all these parameters need to be taken into account jointly for network management practices, in order to ensure a high end-user experience.

KEYWORDS:
YouTube Video Streaming, Large-scale Measurements, Mobile Networks, Streaming

1 | INTRODUCTION

Video streaming traffic is currently the most common type of Internet traffic in mobile networks [11, 19, 43, 51]. With a massive average annual growth rate of 47 % in 2016-2021 [11], operators face the challenge of managing their networks in such a way that their customers are satisfied and the available resources are used efficiently. Although aggressive cache deployment often keeps video out of the core networks, in 2016 more than half of mobile traffic was video with an estimated increase to 78 % in 2021 [11]. As the largest video portal on the Internet, YouTube is currently responsible for 20 % of the traffic in mobile networks [43] since about 75,000 YouTube videos are watched per second worldwide [19]. Video streaming is therefore one of

the most important services for mobile network operators and requires special consideration in terms of the end-users and their experienced quality, which means that operators need to understand YouTube's performance and its influence factors.

From an end-user point of view, video streaming performance can be determined by Quality of Experience (QoE) [45], and a broad consensus in literature suggests that QoE of video streaming is mainly affected by three major performance metrics: (1) stalling (video playback freezes), (2) playback quality and time on highest quality layer [47], and (3) initial delay (time from the user request to the actual playback of the video) [45]. To optimize these metrics, today's video streaming players use two effective techniques: buffer and quality adaptation. To compensate for network fluctuations and survive short-term network outages, a buffer is used on the client side to pre-store video data. Additionally, in order to meet the currently available, long-term average Internet bandwidth, the playback quality is adapted to the available network bandwidth with respect to video encoding bitrate. When both techniques are used together with the Hypertext Transfer Protocol (HTTP) protocol for information exchange, this is called HTTP Adaptive Streaming (HAS).

HAS is primarily influenced by network transmission, and adaptation and streaming performance depends on many different factors that manifest on different layers of the TCP/IP reference model. Thus, and according to measurements [2, 15, 40], video performance for example highly depends on the underlying network conditions (e.g., network and link layers). For mobile networks, the underlying network conditions vary significantly from one operator to another, since each operator has a different deployment strategy and network configuration. Furthermore, the selection of the browser at application layer and the choice of protocol in transport layer can have significant impact on the HAS video performance. Overall, video streaming performance is impacted by the mobile service provider, network throughput, transport layer protocol, and application-level playback. Everything contributes to user-level performance, which is described as QoE [45]. Many practical questions center around optimizing the video streaming QoE, so that the factors influencing HAS QoE as well as the measurement methodology in combination with the ability to measure in a large-scale and comprehensive way, are of utmost importance.

Over the past few years, several measurement studies have been carried out on this topic [2, 7, 15, 31, 36, 40, 49]. However, these studies usually concentrate on a single aspect such as content caching [2], end user experience [7], or content distribution [31]. Normally, exactly one layer is in focus, as it is difficult to measure both comprehensively [31] and across multiple layers [49, 52]. Another challenge is to conduct large-scale measurement studies in several countries, so that the measurement results are generally valid. Few existing measurement studies are unfortunately outdated, and do not take into account current developments in streaming such as adaptivity [2, 15]. The most related paper to our work is [15], which also links user satisfaction from YouTube with device-level and infrastructure influence factors in the network.

In this paper, we focus on the largest HAS portal on the Internet, YouTube, and carry out a comprehensive study investigating the influence of different factors on its streaming performance in mobile networks. Leveraging the Measuring Mobile Broadband Networks in Europe (MONROE) testbed [5], YouTube video streaming is measured in 4 countries with 13 different mobile network configurations. We collect 1,824 measurements samples in operational networks. With such a large scale measurement campaign, our goal is to quantify the impact of certain settings at different layers on YouTube's streaming QoE. More specifically, we analyze the role of the browser (e.g., Firefox and Chrome), the impact of the transport protocol (e.g., TCP or QUIC), the influence of network bandwidth, and signal coverage on streaming QoE. The contributions are as follows: first, we present a novel approach to performance evaluation, considering the influence of different layers on video streaming QoE. A measurement methodology for YouTube for several different influence factors is given and described. We design and implement a tool called *VideoMon* which is capable of dissecting the impact of different parameters from the said layers, and associating these with relevant QoE metrics. Second, a large-scale measurement study on the performance of YouTube in the European MONROE testbed is presented. MONROE provides the opportunity to run measurements on mobile as well as stationary nodes which are connected to different network operators. In this work, we focus on stationary nodes to measure the performance of video streaming in mobile networks without the influence of the mobility. Third, different influence factors are analyzed out of the measured data, from a network management perspective, the impact of each individual layer on YouTube's video streaming performance is quantified according to selected QoE metrics. Finally, we provide our source code and measurements as open data, so that our results can be reproduced, and further analyses can be conducted by the research community.

The remainder of this work is structured as follows. In Section 2, relevant background information is summarized. Related work is given in Section 7. In Section 3, the methodology to analyze YouTube influence factors and the corresponding monitoring framework is decribed. Section 4 deals with the measurement data and describes the dataset. Video streaming performance from an application layer point of view, and the impact of different influence factors on the application layer QoE are evaluated in Section 5. Section 6 provides a summary and presents an aggregated view on the topic, where Section 8 concludes the article.

2 | BACKGROUND

This section provides relevant background information to understand the following paper. In order to provide a holistic view of mobile streaming, we first introduce video streaming with the MPEG DASH streaming standard. Then, we focus on streaming QoE and the influence factors for streaming on the Internet.

Video Streaming with MPEG DASH Many video streaming applications today depend on the same principle of segment-based downloading as standardized in Dynamic Adaptive Streaming over HTTP (DASH) [13] and HLS [38]. Among most popular video streaming applications, YouTube and Netflix use MPEG DASH [25]. YouTube, for example, has been using HTML5 as the default playback option since early 2015 and as part of this, DASH in HTML5 is used wherever possible (e.g., IE11, Chrome, Safari) [25]. In MPEG DASH, the video content is provided in differently coded segments at the server side, and the client asks the appropriate segments to stream the video. The segments are selected by the client to match the available end-to-end bandwidth between client and content server. The transmitted segments are played out and, depending on the bandwidth, the video quality is adapted with this principle. The adaptation is carried out on the client side by the so-called adaptation logic. An adaptation logic can pursue many objectives where the main goal is generally to avoid negative impact on video playback for the user.

Streaming Quality of Experience and QoE Metrics The main factors that influence the video streaming performance are (1) stalling (video playback freezes), (2) playback quality and time on highest quality layer [47], and (3) initial delay (time from the user request to the actual playback of the video) [45]. These factors define the quality of streaming for the user, usually characterized in the literature by Quality of Experience (QoE). In the following, we refer to this quality parameters as *streaming QoE metrics*. In Table 1 all metrics are listed. For quality, we distinguish in this work between start quality, highest quality used during streaming, most used quality during streaming, time spent at different quality levels, and number of up and down-switches. Furthermore, we additionally look at goodput as application throughput during streaming and buffer level, which is the amount of prefetched video contents in seconds. Buffers and goodput are metrics that inherently affect other metrics. Stalling is created when the buffer has dropped to zero seconds and no data is playable. With the consideration of these additional values as metrics, the explanation of the facts later becomes easier. These metrics highly influences the QoE, if a metric gets worse, the QoE also deteriorates for the user. The extent of the degradation depends on the metric itself, e.g., initial delay has a higher influence on the QoE than the video quality [18]. Thus, it is of interest to know the factors and optimize networks against such factors. The factors influencing the metrics are of interest. The following paragraph describes the communication layers used in streaming.

Influence Factors for Streaming According to [12], the TCP/IP reference model defines four different layers for the transmission of data on the Internet, which are traversed to accommodate the data transport: the first layer is the *Host-to-Network Layer* and takes care of the conversion of the data into signals from the host to the network. Here, optimization and control mechanisms are firstly used to ensure the transmission. Relevant influence factors on this layer are signal strength, channel condition if mobile, PHY layer configuration, and access technology. On this layer, we later evaluate the signal strength as an influencing factor, see Table 1 . The second layer involved is the *Internet Layer*. Here, the data is transmitted across network boundaries and a key factor is the connection bandwidth in and between the networks, packet loss, delay, and number of hops until the destination is reached. At *Transport Layer*, video data in the TCP or QUIC protocol on YouTube is assigned to the video application and an error and congestion control is performed. All levels manifest various factors that can change the video streaming behavior. Table 1 presents the influence factors, along with their associated layer of the network stack. Our goal in this study is to dissect the influence of each parameter on the QoE metrics given in Table 1 on the left.

3 | METHODOLOGY: FRAMEWORK FOR MONITORING YOUTUBE STREAMING

In order to evaluate the performance of video streaming within this complex mobile ecosystem in a wholesome manner, we design an active measurement framework called *VideoMon*. VideoMon incorporates YouTube headless video streaming via different browsers in a virtualized Docker environment, while also monitoring parameters in the application, transport, network, and physical layers. Characteristics of the framework are as follows.

TABLE 1 List of streaming QoE metrics (*left*) and influence factors on streaming QoE (*right*).

Metric	Description
Initial delay	Time before video playback starts
Goodput	Application throughput during streaming
Quality	Start quality
	Highest quality used during streaming
	Most used quality during streaming
	Time spent at different quality levels
	Number of up and down-switches
Buffer	Buffer size (time series or statistics)
Stalling	Number and duration of stalls

Layer	Category	Parameters
Application	Browser	Firefox or Chrome
Transport	Protocol	TCP or QUIC
Network	Bandwidth	Network bandwidth
Physical	Signal	Signal strength

- **Docker implementation:** To achieve a uniform and reproducible measurement environment so that reliable measurements can be made, the tool is designed as a Docker container. Thus, it is independent from system software settings[1].
- **Testbed-compatible scenario:** The container is able to emulate an end-user watching videos on YouTube, via different browsers. It can be run without user interaction on simple measurement nodes (e.g. downsized, headless Linux PCs as widely used in Mobile Broadband (MBB) testbeds).
- **Speedtest measurements:** Before starting the video streaming session, a speedtest is run in order to collect information regarding the network bandwidth. We detail this component in Section 3.1.
- **Application QoE monitoring:** During the video playback, different QoE metrics such as video quality, number of switches, buffer size, and stallings are monitored. We detail this component in Section 3.2
- **Server identification:** After the video playback, Round-Trip Time (RTT) measurements are run against the YouTube servers that were used for the video playback, with ICMP ping.
- **Network traces:** To capture network traffic information during experiments, the network protocol analyzer `tshark` is used.

Figure 1 demonstrates the main components of the VideoMon container. It is a combination of the tools *MONROE-Nettest* and *YoMo*, joined with a testbed-compatible wrapper which can additionally parse streaming logs online, collects metadata, and allows for conducting batch-based measurements (with multiple configurations). The container is configured using an input file or a JavaScript Object Notation (JSON)-formatted string. First, it runs the *Nettest* component to measure MBB performance, after which it runs the *YoMo* component which is the core for video streaming. If the container is run on a testbed capable of providing metadata through ZeroMQ (ZMQ) in a publish-subscribe manner[2], this is also collected throughout the whole experiment. We detail the main components *Nettest* and *YoMo* in Sections 3.1 and 3.2.

3.1 | MONROE-Nettest

MONROE-Nettest [26] is a speedtest implementation made available for testbed experimentation, which is designed primarily as an Experiment as a Service (EaaS) on top of the MONROE platform, but which can also be run as a standalone tool. In this study, we use the MONROE-Nettest core with a new wrapper as a part of the VideoMon container, in order to measure MBB performance and provide a baseline for video quality evaluation.

The Nettest component is a wrapper built around the MONROE-Nettest client core [32] and combines a number of functionalities. **Configuration:** The container is highly configurable, including test duration and number of flows. All of the MONROE-Nettest configuration parameters [32] are also available for use in the VideoMon configuration file. **Metadata:** If the

[1]The VideoMon container is based on the `monroe:web` Docker image provided by the MONROE project. It runs on `debian:stretch` (Debian 9D).

[2]ZMQ is a distributed asynchronous messaging library, which provides a publish/subscribe mechanism where consumers can subscribe to selected parts of the information published by a provider (http://zeromq.org/). For details regarding the ZMQ implementation on the MONROE platform, see Section 4.1.

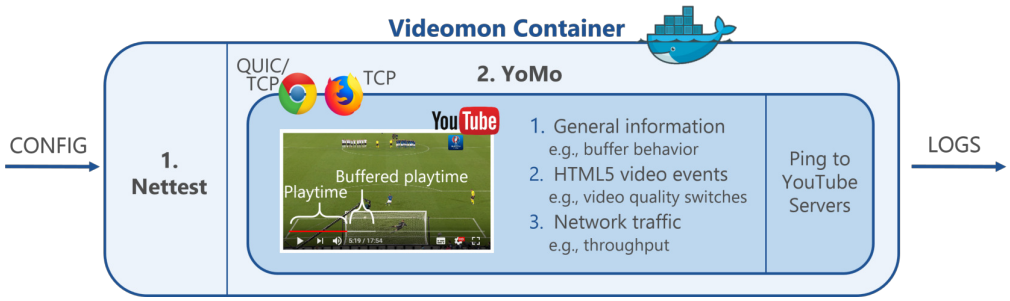


FIGURE 1 Components of the VideoMon Docker container.

experiment is run on the MONROE platform, and depending on whether the interface is fixed or cellular, it establishes that metadata information is available. It waits for a specified duration, after which execution stops if there is no metadata. **Traceroute:** It runs a `traceroute` against the measurement server specified in the configuration file. The result is parsed and the prettified version is written as an output file. **Nettest core:** After the traceroute, the MONROE-Nettest client core is run. The Nettest client core is based on RTR Multi-threaded Broadband Test (RMBT) [42], which is a codebase used by a number of National Regulatory Authoritys (NRAs) in Europe for their crowdsourced measurement applications. We choose to use this particular tool due to its testbed-friendliness, modular and configurable design. Details of the client core can be found in [26]. **Output:** Finally, output files are gathered and managed. The Nettest component produces 4 output files: ***summary.json*** presents the result fields such as calculated Downlink (DL) and Uplink (UL) data rate, and the median Transmission Control Protocol (TCP) payload RTT. It also includes a summary of the measurement configuration, including fields such as the test identifier, main timestamps, and basic input parameters, as well as metadata. ***flows.json*** includes the raw time series of bytes downloaded/uploaded versus time for each TCP flow opened during the bandwidth measurement. ***stats.json*** includes samples of the socket option `TCP_INFO` for each TCP flow, in a configurable granularity. ***traceroute.json*** includes the results from the traceroute towards the selected measurement server. If the Autonomous System Number (ASN) per-hop is not available from the traceroute, this information is added via a lookup.

3.2 | MONROE-YoMo

YoMo (YouTube Monitoring) is a tool designed to measure video streaming sessions in YouTube citeSchwind2017. Here, a user is emulated who opens a browser and streams a video from the YouTube webpage. During the streaming, several QoE performance indicators are monitored on application as well as on network layer. We update this tool and integrate it into the VideoMon framework, by adding Chrome browser compatibility, new output files, and batch measurement capability through the multiple-configuration option (e.g., one measurement with Firefox, then one measurement with Chrome using TCP, and one measurement with Chrome using QUIC). To collect additional information, the VideoMon measurement setup includes ping measurements to the used YouTube servers after the YoMo video playback. This modified version is called MONROE-YoMo.

YoMo Core

The goal of the YoMo is to collect QoE related performance indicators of YouTube video streaming without user interaction. To run the application headlessly in a testbed, which does not provide any physical screen, the X virtual frame buffer (Xvfb) is chosen. It is used to create a fake display port, so a browser can be run via the X server. To simulate a YouTube video request in a browser within the Docker container, the web browser automation tool Selenium is used.

To monitor the KPIs on application layer, a custom JavaScript program is injected when the video playback starts. This program monitors the media events and the media properties of the HTML5 player on YouTube. The monitoring on application layer is divided into two parts: The monitoring of HTML5 video player events and of playback and buffer information. The video player events were monitored using an event listener and requests of the current parameters every second. The event listener listens to player events and stores them together with their timestamp. These events include the player state like stalling, seeking, paused, playing, and ended. In order to detect changes in the video quality, duration, or volume, as well as to detect if a new video is played (change of the YouTube ID and the title), a JavaScript function to check these parameters in specified time

steps is used. Whenever a parameter has changed, the new value is logged with the corresponding timestamp. Additionally, also the YouTube ID and the title were stored. But not only player events, but also information about the buffer level is measured during the video playback. Every second a snapshot of the current timestamp, the current video playback time, the buffered video playback time, and the available playback time is taken and stored in a log file.

Not only parameters on application layer, but also parameters on network layer are monitored during the streaming. On the one side, network traffic information are collected using the network protocol analyzer *tshark*. Here, for example the TCP segment length and the frame length is stored in a log file. On the other side, all HTTP request and response headers are stored in another log file. From this log file, for example the host names of the used YouTube servers can be filtered out.

Ping to YouTube Servers

In addition to the video streaming itself, after the video playback we conducted a short ping measurement to the YouTube servers. Thus, we picked the host names of the used YouTube servers out of the HTTP request and response headers. Afterwards, for each used server we started an ICMP ping measurement to test the reachability of the hosts and, for example, the RTT, which is the time it takes for a request send by a browser to when it receives a response from a server. All results of the ping measurement are stored in a log file.

Output Files

In total, YoMo produces five output files: The **httpLog** includes all request and response headers are stored in *yomo_httpLog_C.json* for video playback using the chrome browser or *yomo_httpLog_FF.txt* for video playback using Firefox browser, respectively. Here, detailed information about the requests like the video itag, which is a specific label YouTube uses to identify the used video formats like the video resolution and frame rate, or the host names of the used YouTube servers is stored. **buffer.txt** gives information about the video playback times and the buffered video segments. Here, every second a snapshot of the current video playback time, the buffered video playback time, and the available playback time is taken. **events.txt** stores all caught events thrown by the HTML5 video player and also provide some additional information about the video. Here, details like the used quality level and quality changes, the duration of the video, as well as events like stallings are collected. **tshark.txt** shows a snapshot of the network traffic during the video playback. Here, the values are stored line by line. To give an example: the first three entries in each line show the timestamp, the TCP segment length and the frame length, which is later used to calculate the goodput. **ping.json**: gives information about the ping measurement to the used YouTube server. This includes, for example, jitter, packet loss, and average ping time.

4 | MEASUREMENTS

For our measurements, we instrument Europe's first transnational open-access hardware-based testbed for MBB experimentation, MONROE. In this section we first describe the MONROE platform, then detail our measurement setup and campaign details, and finally describe our rich dataset which is made available under [44].

4.1 | MONROE Testbed

H2020 MONROE [4, 39] is the first open access hardware-based platform for independent, multi-homed, large-scale experimentation in MBB heterogeneous environments. MONROE comprises a large set of custom hardware devices, both mobile (e.g., via hardware operating aboard public transport vehicles) and stationary (e.g., volunteers hosting the equipment in their homes), all multi-homed to 3 operators using commercial grade subscriptions. The platform allows for systematic and repeatable end-to-end measurements as well as experimenting with novel protocols and services, which are essential for evaluating network performance, especially for assessing the quality experienced by end users.

While existing experimental platforms can also allow for large-scale measurements, MONROE has a unique stance in terms of combining the following: experimentation in *operational MBB*, *large-scale* deployment and measurement capabilities, multi-homing (connecting to multiple networks at once), *end-to-end* measurements (representing an end-user perspective from the edge of the network), and a *controlled* and *repeatable* setup, as opposed to crowdsourcing.[3]

[3]MONROE is complementary to crowdsourcing approaches and the control over the measurement environment tackles the shortcomings of crowdsourced measurements, such as non-uniform end-devices and complex incentive mechanisms, but at the cost of a smaller geographical footprint.

MONROE enables users to run custom experiments and to schedule experimental campaigns to collect data from operational MBB and WiFi networks, together with full context information (metadata). MONROE can accommodate performance evaluation of different applications (e.g., web and video) over different networks or testing different protocols and solutions under the same conditions.

In assessing the performance of a prominent video streaming service over operational MBB networks, we rely on the following aspects of the MONROE platform.

- **Uniform hardware design:** MONROE operates around 200 nodes in 4 countries in Europe (Norway, Sweden, Italy and Spain). In terms of hardware, each node has a main board that is a small programmable computer and supports (at least) 4 interfaces: 3 3G/4G modems (CAT6-capable) and a WiFi modem. The node software is based on a Linux `debian:stretch` (Debian 9D)[4] distribution to ensure compatibility with multiple hardware configurations and to enable a large set of experiments. All measurement nodes have the same hardware and software capabilities.

- **Multi-homing support:** To have a view of different Mobile Network Operators (MNOs) and/or different wireless technologies under the same conditions, the same node should connect to multiple providers at the same time. We use all available cellular connections of the MONROE nodes, as well as the Ethernet interface for additional baseline measurements, which can be found in our dataset [44].

- **Rich context information:** While analyzing measurements, context information is crucial. The MONROE platform monitors network conditions, and registers the time and location of the experiment, as well as the metadata from the modems, including, for example, connection mode and signal strength.[5]

- **Easy instrumentation:** To provide agile reconfiguration and access to different software components, the experiments on the MONROE system run in the Docker light-weight virtualized environment. This also ensures containment of external actions in the node system. The results of the experiments are periodically transferred from the nodes to a remote repository. Access to platform resources are enabled through a user-friendly web portal [33] that allows authenticated users to leverage the scheduler to deploy their experiments.

- **Large scale and diversity:** To give a representative view of the characteristics of video streaming performance in MBB, we need to collect measurements from a large number of vantage points, including diverse geographical settings. This is achieved by the use of a large number of measurement nodes spread across 4 countries, providing a statistically significant amount of measurements from each different configuration.

- **Open source:** MONROE public software repositories contain all the files necessary to build new user experiments, as well as experiment templates. Example experiments provided by the MONROE Consortium include speedtest [26], web browsing, Dynamic Adaptive Streaming over HTTP (DASH) and WebRTC [27, 28, 34]. The source code for our tool *VideoMon* is also available in its entirety under this repository, as well as a public Docker image under [29].

4.2 | Measurement Setup

For evaluating the performance of YouTube video streaming in MBB networks, we execute streaming sessions using the *VideoMon* client in a number of different configurations. Figure 2 shows the multiplexed approach in which we conduct our measurements. On the left we can see the MONROE node on which the VideoMon docker container is run. The node (client) emulates a common end-user device, it has 2 CAT6-capable modems on an APU2 board. Each node is connected to up to three network operators op1, op2, and op3 as well as Ethernet.

In our setup, the MONROE node running the VideoMon container act as video clients, requesting a video from YouTube servers. Here, we use the original YouTube servers, no emulations, APIs, or proxies. In this measurement setup, we employ only stationary nodes, which have a fixed location. In the following, the setup is explained in more detail.

[4]Debian 4.9.110-3+deb9u6 (2018-10-08) x86_64 GNU/Linux, kernel version 4.9.0, TCP flavor: cubic

[5]The metadata distribution is implemented in a publish/subscribe pattern using ZMQ. The metadata stream is available for experiments during their execution using the ZMQ subscription mechanism. Metadata entries are generated in a single-line JSON format. Every data entry is labeled with a “topic" field. Consumers may subscribe to the whole stream of metadata or just to some topics. The metadata subscriber module runs in the nodes and subscribes to all the topics, writing JSON entries to files in a special file system location. A synchronization process transfers those files at certain intervals (tens of seconds) to the MONROE server.

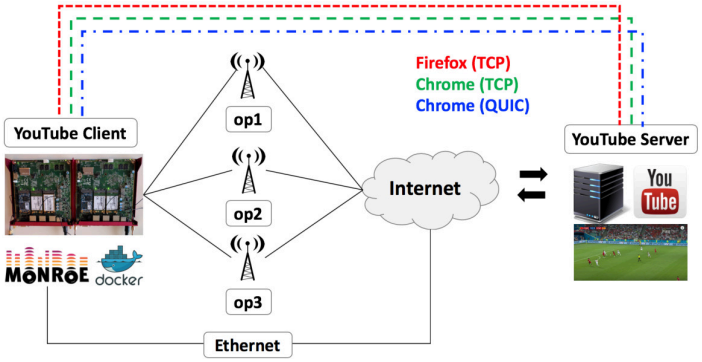


FIGURE 2 VideoMon measurement setup.

Content

For our experiments, we selected a YouTube video of the category sports having the YouTube ID (YTID) `4rp2aLQl7vg` and the title *Portugal v Spain - 2018 FIFA World Cup Russia*. The total duration of of the video is 130*s*. This video was selected as it contains no advertisements and is available different quality levels of up to `1080p`. In July 2018, this video has 57 millions views, thus it is also very popular. There are a total of 6 quality representations from `144p` up to `1080p`, having a frame rate of 25*fps* and, in addition, quality level `720p` and `1080p` are also available with 50*fps*. All quality levels are available for both Firefox and Chrome as they are available in `webm` as well as `mp4` format.

Configurations

We run the measurements in the form of *batches* on each node. A batch consists of one Nettest experiment, followed by three YoMo experiments. In the “multi-config" YoMo experiments, first, Firefox (version 56.0.1) is used to stream the video, after which Chrome (version 64.0.3282.140) with *noQUIC* option, and finally Chrome with *QUIC* enabled. The batch is repeated for all available interfaces, including Ethernet if applicable for the node. The approximate duration per batch is around 15 minutes.

Networks and Locations

We run our campaign in 4 countries: Norway, Sweden, Italy and Spain, with a total of 64 nodes (9 nodes in Norway, 19 nodes in Sweden, 4 nodes in Italy, 2 nodes in Spain). Overall, we leverage 13 mobile network configurations (including 11 native and 2 roaming scenarios), which are listed in Table 2 . Overall, 11 of the measurement nodes include an additional Ethernet interface for baseline measurements.

Table 2 presents, in compact form, the country of the MONROE nodes (clients), Mobile Country Code (MCC) and Mobile Network Code (MNC) of the active network, MCC and MNC of the Subscriber Identity Module (SIM) card, name of the active network, and the number of nodes that were used for experimentation from these configurations. The distribution of SIM cards to platform nodes, availability of nodes for scheduling measurements[6], and the number of viable results after sanity checks and filters[7] have affected the sample distribution, which can be observed from the table. In order to alleviate this bias, we use normalization in our analysis in Section 5.

Dataset

We provide our complete measurement dataset as open data under [44]. This dataset includes results and metadata corresponding to a total number of 2114 measurements (1824 cellular measurements included in the analysis). Experiment results in the dataset have been collected within a period of one week. The repository includes a `README` file explaining the organization of each directory.

[6]Depending on the availability of the platform, the measurements were conducted in multiple campaigns, spanning a total duration of 17 days between 23.06.2018 and 09.07.2018.

[7]These include checks such as wheher the node was connected to the Internet during the whole measurement run, both Nettest and YoMo experiments ran successfully, etc.

[8]Telia SE roaming in Telia NO.

[9]Wind roaming in Vodafone Spain.

TABLE 2 List of MBB networks used for the measurement campaign.

Country of Client	Network	SIM	Network Name	Number of Nodes
Norway	242-01	242-01	Telenor NO	9
	242-02	242-02	Telia NO	9
	242-02	240-01	Telia NO[8]	1
	242-14	242-14	ICE	7
Sweden	240-01	240-01	Telia SE	19
	240-02	240-02	H3G	18
	240-08	240-08	Telenor SE	16
Italy	222-01	222-01	TIM	4
	222-10	222-10	Vodafone Italy	4
	222-88	222-88	Wind	4
Spain	214-03	214-03	Orange	1
	214-04	214-04	Yoigo	2
	214-01	222-10	Vodafone Spain[9]	2

5 | EVALUATION

In this section, our goal is to quantify the influence of parameters from different layers of the TCP/IP stack on the performance of YouTube video streaming in MBB networks, in terms of objective QoE metrics, using the results from our large scale measurement campaign over the MONROE testbed. We focus on the QoE metrics introduced in Section 3.

We first take a general look at the overall dataset, and then evaluate parameters from each layer separately. Section 5.1 provides general insights into the performance of YouTube video streaming in MBB. Section 5.2 focuses on the the impact of the application layer, more specifically the effect of browser, on selected performance metrics. In order to investigate the influence of the transport layer, in Section 5.3 we compare the performance of video streaming over TCP versus over Quick UDP Internet Connections (QUIC). We evaluate the impact of the network layer in Section 5.4 by having a look at different categories of available network bandwidth. Finally, in Section 5.5, we analyze the impact of the physical layer on streaming performance, by focusing on different categories of signal coverage. This evaluation methodology allows us to dissect the influence of different layers on video streaming QoE separately, within the complex mobile ecosystem, and enables intelligent network monitoring through the use of detailed granularity in performance prediction.

5.1 | Overview

Evaluation Methodology

As mentioned in Section 3, we focus on objective QoE metrics representing the performance of YouTube video streaming, which relate to initial delay, buffer status, quality levels and transitions. Since all the measurements in our dataset come from stationary MONROE nodes, we do not focus on stalling for the scope of this study. Further, since we are primarily interested in the performance of video streaming in MBB, we limit our analyses to the part of our dataset collected from cellular networks. Thus, all references to our dataset should be read as an indication of the cellular measurement, from this point onwards. To make the measurement result more comparable, we limited the evaluation to a total playback time of 80 seconds per video.

Example Evaluation For Single Session

Figure 3 shows the streaming behavior of a YouTube video playback session which was captured using Firefox. The x-axis shows the time since the start of the video playback in seconds, while the y-axis indicates the played and the buffered playback times in seconds, respectively.

The current playtime of the video is illustrated with yellow dots, and fitted with a black line. As the playtime is constantly increasing, it is seen that there are no interruptions (stallings) during the playback. The orange curve shows the buffered playtime during the video playback. The buffer increases to about 60s buffered playtime. When all segments are downloaded, no further segments have to be requested and the buffer decreases until the video playback is finished. The dashed red vertical line indicates

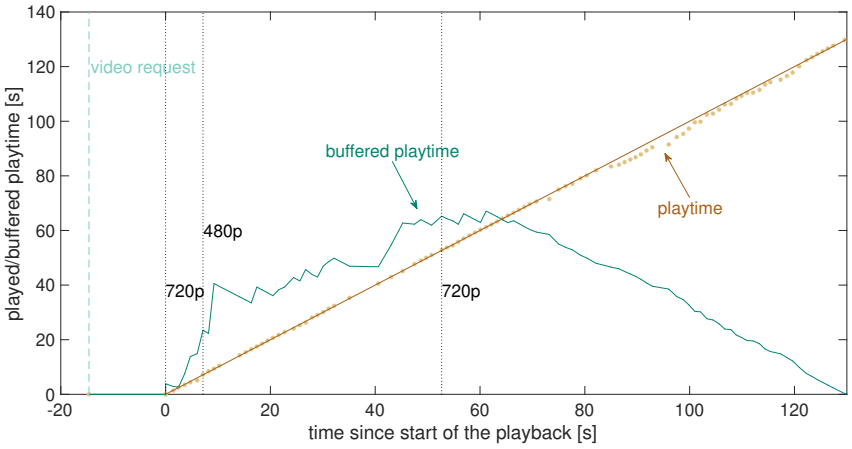


FIGURE 3 Timeline for a sample video playback session and its measured performance metrics.

TABLE 3 Influence of the used browser on the performance metrics.

	Initial delay	$\text{Buffer}_{\text{median}}$	$\text{Quality}_{\text{start}}$	$\text{Quality}_{\text{most used}}$	$\text{Quality}_{\text{max}}$	#Switches	Goodput
Firefox	13.27s	52.84s	720p	720p	720p	1.24	3.553$Mbps$
Chrome_{all}	10.00s	42.17s	720p	480p	720p	1.78	2.803$Mbps$
Chrome_{TCP}	9.50s	42.00s	720p	480p	720p	1.42	3.271$Mbps$

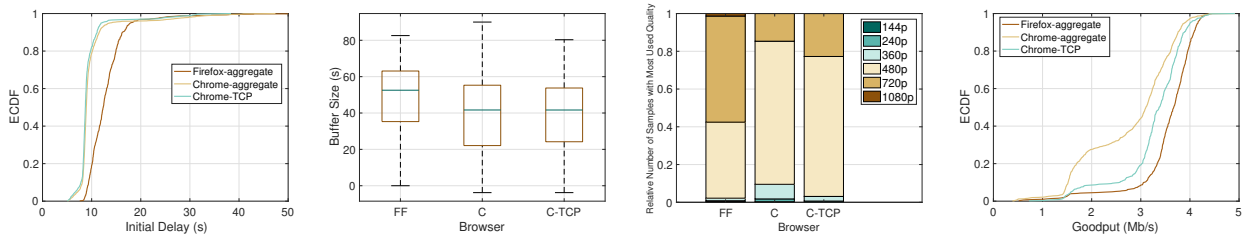
the timestamp at which the video was requested. The time between the video request and the start of the playback is referred to as *intial delay*, which is Approximately 15s for this example. The VideoMon framework not only tracks the buffer behavior during streaming, but also monitors the video quality levels. The used quality levels are marked as dashed black vertical lines, starting with a quality of 720p and down-switching to 480p over 7s, and up-switching again to 720p shortly before the video ends. For this example, the most used quality layer is 720p.

General Insights

We aggregate the performance values from specific measurements, such as the example given above, over the whole dataset for a preliminary overview. Thus, the median buffered playback time for the dataset is 45.66s with a maximum buffered playback time of 65.38s. The average initial delay is 11.07s and most of the videos have a start quality of 720p. For most of the video playbacks, no quality switches occurr in the first 10s. Overall, the absolute maximum quality is 1080p, but the most common maximum quality for a video session (*mode* of the maximum qualities) is 720p. On average, the most used quality level is 480p. The average number of quality up-switches is 0.48 and number of down-switches is 1.12. This means that an average video playback has between $\{0, 1\}$ quality up-switch and 1 quality down-switch. The average goodput during video playback is 3.048$Mbps$. Next, we take a closer look at the influence of parameters at different layers, for which we split our dataset into groups with respect to the parameter under investigation.

5.2 | Application Layer: The Influence of Browser on Streaming QoE

In this section, we investigate the effect of the browser on the video streaming performance metrics. Table 3 gives an overview of the browser's influence on selected performance metrics, listing the average of initial delay, median buffer, total number of quality switches, goodput, and the mode of start quality, most used quality, and maximum quality for all measurements which were conducted using Firefox and Chrome, as well as all measurements with Chrome using only TCP. We observe differences in initial delay, buffer, most used quality, total number of quality switches (up and down combined), and goodput with respect to browser. In the following, these differences will be analyzed in more detail.



(a) Initial delay **(b)** Buffer **(c)** Most used quality **(d)** Average goodput

FIGURE 4 Performance metrics for different browsers (FF: Firefox, C: Chrome aggregate, C-TCP: Chrome with TCP).

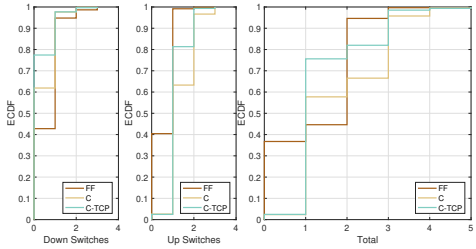


FIGURE 5 Number of quality switches, per browser (FF: Firefox, C: Chrome aggregate, C-TCP: Chrome with TCP).

Initial Delay

We see from Table 3 that there is a difference of more than 3s between the average initial delay for different browsers. To compare the effect of browsers in detail, Figure 4 a shows the distribution of the initial delays per browser. Here, the values for each group have been aggregated over cellular networks. It can be seen that the Firefox has a relatively larger initial delay than Chrome. For both Chrome settings, the minimal initial delay is 5.13s while for Firefox the minimum is 7.64s. While 78.75% of the aggregated Chrome measurements and 82.21% of the Chrome measurements using TCP have a lower initial delay than 10s, only 19% of the Firefox measurements have initial delay at this value. From these results, we conclude that the browser selection has a noticeable impact on the initial delay for video streaming.

Most Used Quality

Figure 4 c presents an overview of the most used quality during measurements, in terms of the relative number of samples with a certain quality as its most used quality. Here, the first stacked bar plot shows the most used qualities for Firefox, the second bar for Chrome, and the third bar for Chrome only using TCP. The different colors indicate different quality levels starting by `144p` up to `1080p`, which was the highest available quality for this video. Again, a clear difference can be seen for the different browsers. While 57.55% of the Firefox video playbacks were mostly played in `720p` or higher, only 14.66% for aggregated Chrome and 22.75% for Chrome with TCP were played in `720p`. In both Chrome settings, no video was mostly played at a quality level higher than `720p`. This looks different for Firefox: Here, some video streaming sessions (1.34%) also used `1080p` for most of the playback time.

Quality Switches

Figure 5 shows the Empirical Cumulative Distribution Function (ECDF)s of the number of up and down-switches as well as the total number of switches for different browsers. For Firefox, the average number of up-switches (0.64) is almost equal to the average number of down switches (0.61). For both Chrome settings (aggregated and only TCP), the number of up-switches is considerably smaller than the number of down-switches, having an average of 0.41 in comparison to 1.37 and 0.25 in comparison to 1.17, respectively. 36.74% of sessions in Firefox neither switch up nor down, compared to about 75.61% of Chrome sessions that do not switch down. However, Chrome almost always switches up at least once with 81.34% (for aggregated TCP and Quic) and 63.27% (for TCP) of sessions switching up exactly once.

Average Goodput

From Table 3 , we see a difference of up to 0.7*Mbps* in terms of average goodput between different browsers, in favour of Firefox. Figure 4 d details this difference in the form of an ECDF, for aggregated Firefox measurements, aggregated Chrome measurements, and Chrome measurements over TCP (excluding measurements over QUIC). Minimum goodput of Firefox, Chrome-aggregate, and Chrome-TCP are 0.7, 0.4, and 0.5*Mbps*, where the maximum is 4.4, 4.9, and 4.9*Mbps*, respectively. It is possible to see from the shape of the ECDF curves that the majority of the difference between the protocols appears in low to mid-range goodput values (1.5-3.5*Mbps*). At the median range, the difference between the Firefox and Chrome aggregate curves is around 0.5*Mbps*, which corresponds to more than 13%.

One observation here is the consistency of the performance results in terms of average goodput and time spent at the most used quality (Figure 4 c). Time spent at higher qualities is understandably related to the average goodput achieved during streaming, in that, the higher the average goodput, the larger the time spent at higher qualities. Another important observation is that the performance of Chrome is worse when the aggregate including QUIC measurements is considered (as opposed to considering TCP measurements only). The difference at the median range is 0.2*Mbps*. We will elaborate on this phenomenon further in the next section, in the context of the influence of the transport layer on streaming QoE.

Discussion

We observe a clear difference between Firefox and Chrome in terms of initial delay: on average, the video playback starts later for Firefox. However, looking at the start quality, we see that both browsers tend to start streaming with `720p`, indicating that either Firefox buffers more aggressively before starting playback, or Chrome is faster in terms of rendering time. In terms of the latter, we hypothesize that this might be due to the technical features of the browsers, such as the implementation of video codecs, and the use of libraries. When we check the `itag`s of each streamed segment, we observe that in all of our measurements, Chrome uses the `webm` format where Firefox uses `mp4`. Assuming that the initial delay depends on video encoding and network bandwidth, we conclude that one of the main reasons for the difference in the initial delay between Firefox and Chrome is the use of the `mp4` and `webm` formats when streaming from YouTube, respectively. For the former, we take a closer look at the buffer distribution of different browsers. Figure 4 b shows the boxplot of the buffer sizes, from which it is possible to see that Firefox has a higher buffer on average.

We conclude that at startup, Firefox buffers more data on average, and starts playback later than Chrome. This way, it has the possibility to use higher qualities during streaming (seen from Figure 4 c, where the most used quality for Firefox is higher than for Chrome). Firefox stays longer at the most used quality, and thus its total number of switches is also lower. Accordingly, the goodput of Firefox is higher (since it streams the segments in higher quality). In summary, we can say that Chrome opts for better initial delay performance at the expense of lower overall quality, whereas Firefox opts for better overall quality at the expense of larger initial delay. Overall, we conclude that browser selection is a significant influence factor on initial delay, buffer, most used quality, total number of quality switches, and average goodput.

5.3 | Transport Layer: Influence of TCP and QUIC on Streaming QoE

In this section we investigate the effect of different transport protocols on the video streaming performance. We leverage the YouTube measurements conducted with Chrome (excluding Firefox measurements) and compare the streaming performance over TCP (i.e. QUIC disabled in Chrome settings) versus QUIC (i.e. QUIC enabled in Chrome settings). In every measurement batch, TCP and QUIC measurements with Chrome are conducted one after another, which provides controlled experimentation where both protocols can be assumed to run under similar network conditions. To eliminate the risk of possible distortion, the cache on application layer was disabled. Table 4 gives an overview of the influence of the transport protocol on the selected performance metrics, listing the average of initial delay, median buffer, total number of quality switches, goodput, and the mode of start quality, most used quality, and maximum quality for all measurements which were conducted using TCP and QUIC. We observe differences in initial delay, number of quality switches, and goodput with respect to protocol.

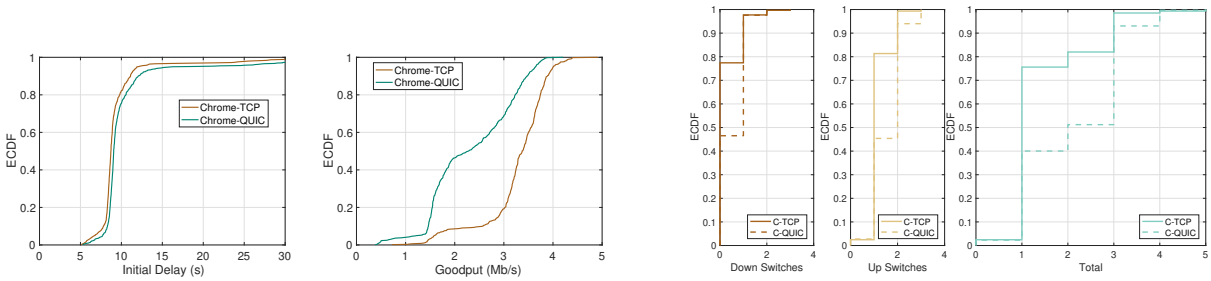
Initial Delay

In Table 4 , a difference of 1*s* in initial delay is observed. Thus, we will have a look at this difference in more detail with Figure 6 a, which presents the ECDF of initial delay for measurements in different networks, over TCP and QUIC. We observe that streaming over QUIC generally incurs a slightly larger initial delay. The difference is lower than 1*s* until the 90th percentile range, after which it grows to around 7*s* at the 95th percentile, indicating that there is a considerable difference between the

TABLE 4 Influence of the transport protocol on the performance metrics.

	Initial delay	$\text{Buffer}_{\text{median}}$	$\text{Quality}_{\text{start}}$	$\text{Quality}_{\text{most used}}$	$\text{Quality}_{\text{max}}$	#Switches	Goodput
TCP	9.50*s*	42.00*s*	720p	480p	720p	1.42	3.27*Mbps*
QUIC	10.50*s*	42.34*s*	720p	480p	720p	2.14	2.34*Mbps*

initial delay incurred by QUIC versus that by TCP, only for a relatively low percentage of measurements. The minimum initial delay is around 5*s* for both TCP and QUIC.



(a) Initial Delay **(b)** Goodput **(c)** Quality Switches

FIGURE 6 Performance metrics for different protocols, TCP and QUIC.

Quality Switches

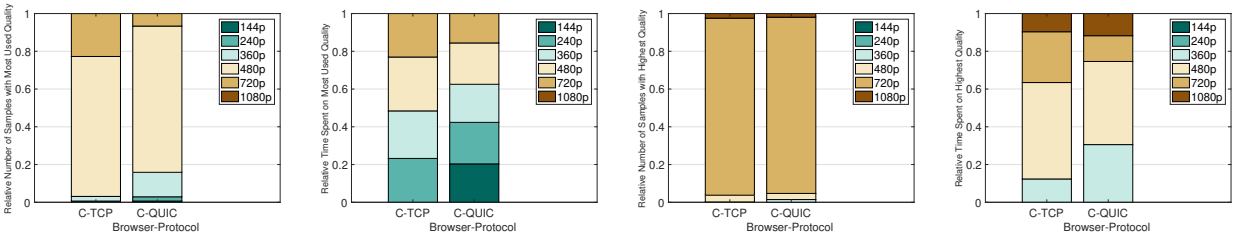
From Table 4 , we see a difference in the average number of switches between QUIC and TCP, where QUIC has 0.5 more switches in total, on average. This is demonstrated by Figure 6 c, which shows the ECDF of the number of up and down-switches, as well as the total number of switches for different protocols, in stairwise fashion. We see that the QUIC curve lies to the right of that for TCP, for both up and down-switches (i.e. QUIC has a larger number of switches in both directions overall). The minimum number of up and down-switches are 0 for both, but with 77% for TCP and 46% for QUIC. Both protocols have more up-switches than down-switches on average. The mode of the total number of switches for TCP is 1 where this is 3 for QUIC.

Average Goodput

TCP and QUIC have a difference of around 1*Mbps* on average in terms of goodput (Table 4). Figure 6 b demonstrates this difference in the form of an ECDF, where it is possible to observe that at the median range, the difference between the average goodput for QUIC and TCP is around 1.1*Mbps*, which corresponds to 33% of the median value for TCP, which is quite significant. Minimum goodput of TCP and QUIC are 0.7*Mbps* and 0.4, where the maximum is 4.9*Mbps* and 4.2 , respectively. It is possible to see from the shape of the ECDF curves that the majority of the difference between the protocols appears in mid-range goodput values (1.5-3.5*Mbps*). The results from this analysis are in accordance with the previous section, where it was observed that the Chrome browser over TCP performs better in terms of goodput compared to Chrome over both protocols aggregated (Figure 4 d).

Discussion

We do not observe a significant difference between TCP and QUIC in terms of initial delay. The protocols are also similar in terms of start quality: both use 720p most often. However, the protocols differ in terms of the overall use of different qualities, which is possible to see from the bar plots of most used and highest qualities, shown in Figure 7 . Here, we observe that TCP uses 720p and 480p as the most used quality for longer than QUIC on average, whereas QUIC has a non-negligible amount of samples with 144p (lowest quality) as the most used quality. Since QUIC uses lower qualities, it also has lower average goodput than TCP, as shown in Figure 6 b. In terms of quality switches, we observe that QUIC consistently has more switches than



(a) Most Used Quality (Count) **(b)** Most Used Quality (Duration) **(c)** Highest Quality (Count) **(d)** Highest Quality (Duration)

FIGURE 7 Quality statistics for different protocols, Chrome-TCP and Chrome-QUIC.

TABLE 5 Influence of the network bandwidth on performance metrics.

	Initial delay	$\text{Buffer}_{\text{median}}$	$\text{Quality}_{\text{start}}$	$\text{Quality}_{\text{most used}}$	$\text{Quality}_{\text{max}}$	#Switches	Goodput
Very High BW	10.41s	46.24s	720p	480p	720p	1.50	3.16$Mbps$
High BW	11.60s	46.23s	720p	480p	720p	1.65	2.97$Mbps$
Medium BW	11.25s	45.61s	720p	480p	720p	1.64	3.07$Mbps$

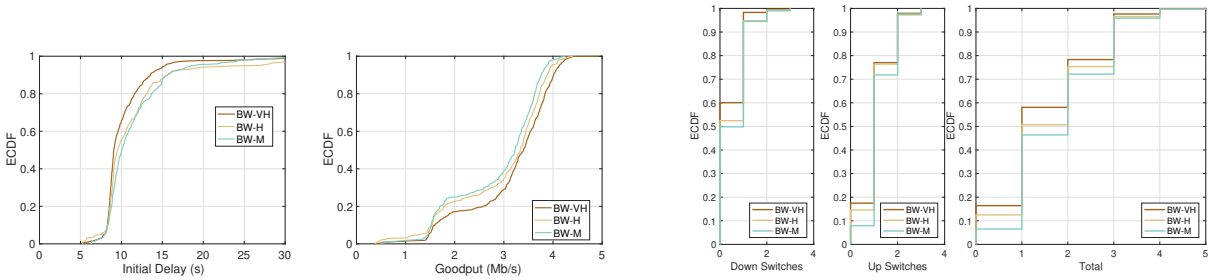
TCP, which might be due to it struggling to use the available network bandwidth and falling to a lower quality, leading to more switches, as well as lower average goodput.

In summary, even though the measurements are run under similar network conditions and are balanced in terms of sample size (617 and 610 measurements for QUIC and TCP respectively), we observe poorer performance from QUIC. One of the possible causes for this is the *proxy effect*, in that many MNOs are known to deploy Performance Enhancing Proxies (PEPs) targeting TCP [23], whereas, due to its relatively recent deployment, such middleboxes have not come into wide use for QUIC. Another possible cause is QUIC's use of the cubic congestion control with *packet pacing* by default (unlike TCP, which uses cubic as congestion control, but without packet pacing by default). Esteban et al. [14] establish that pacing performs poorly for streaming over TCP, which might also be the case for QUIC. We believe that both of these might have an effect on the performance gap observed here, but we leave it to future work to analyze the influence of proxy and packet pacing in detail. Overall, we conclude that transport protocol is a significant influence factor on most used quality, total number of quality switches, and average goodput.

5.4 | Network Layer: Influence of Network Bandwidth on Streaming QoE

In this section we investigate the effect of network bandwidth on the video streaming performance. In this evaluation, the measurement results of Firefox and Chrome, as well as TCP and QUIC are aggregated. We leverage all measurements with respect to the following bandwidth categories: For the category *Very High (VH)*, the bandwidth was greater or equal to 30$Mbps$. We are talking of *High (H)* bandwidth for measured values between 30$Mbps$ and 15$Mbps$. *Medium (M)* bandwidth is defined for values between 5$Mbps$ and 15$Mbps$, while the category *Low (L)* is used for a bandwidth lower than 5$Mbps$. In theory, VH enables 4K streaming, while H allows for 2K streaming, M allows 720p and 1080p, and L can only support 480p and lower qualities from a pure encoding point of view. Since we do not have many samples for L (only 26 measurements compared to more than 250 for all other categories), the evaluation results for L are not as meaningful as the others. For that reason we skip the low category in the further study.

Table 5 gives an overview of the impact of the network bandwidth on the selected performance metrics, listing again the average of initial delay, median buffer, total number of quality switches, goodput, and the mode of start quality, most used quality, and maximum quality for all measurements. Some differences with respect to network bandwidth can be seen for the averages of in initial delay, number of quality switches, and goodput. Thus, in the following these metrics will be analyzed in more detail.



(a) Initial Delay **(b)** Goodput **(c)** Quality Switches

FIGURE 8 Performance metrics for different bandwidth categories.

Initial Delay

As Table 5 indicates a difference of up to 1.19s for the initial delay, we will have a closer look at the distribution of the initial delays per category. Figure 8 a shows the ECDF for Very High bandwidth in red, High in green, and Medium in blue. All curves follow the same trend by increasing fast for the first 60% and flattening afterwards. While the curves for the M and H categories are close together, the initial delay for VH is always lower. 65.44% of category VH have a lower initial delay than 10s, in contrast to 55.21% of H and 48.80% of M. Thus, a trend can be seen that the higher the bandwidth the lower the initial delay. This trend is also visible at the 90% quantiles: For most of all video playbacks in VH, the initial delay was shorter than 13.54s, and for bandwidth category H 15.77s.

Quality Switches

The number of up and down switches in relation to the available bandwidth is shown in Figure 8 c. Here, the ECDFs of the different categories (red for VH, green for H, and blue for M) are shown for up and down switches in separate plots as well as a in a combined plot in the right. For all categories the number of down switches is appreciably higher than the number of up switches. Additionally, in all three plots, we see that the number of switches in both directions increase as network bandwidth decreases. Having a look at the total number of quality switches, 41.92% of the measurements in category VH had one or more quality switches during the playback. For category H and M, the number increases to 49.31% and 53, 61%. This means, the higher the available bandwidth, the stable is the video quality during the streaming.

Average Goodput

The distribution of average goodput for each bandwidth category is illustrated in Figure 8 b. Here again, the red curve corresponds to VH bandwidth, green to H, and blue to M bandwidth. We can see that all curves follow the same trend, having an increased goodput for a higher available bandwidth. For VH to M bandwidth, 10.91%, 4.56%, and 2.06% of the measurements had a higher goodput than 4$Mbps$, respectively. Hence, a positive correlation between the available bandwidth and the goodput can be seen.

Discussion

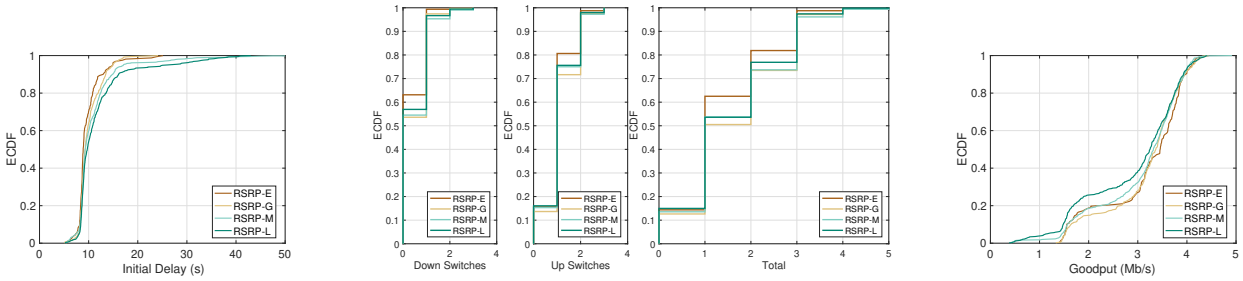
On the network layer, we observe a weak negative correlation between the available bandwidth and the initial delay. This means that the higher the network bandwidth, the lower the initial delay. Our measurements also show the influence of network bandwidth on the number of quality switches: the higher the available bandwidth, the fewer switches during streaming. The network bandwidth is positively correlated to the average goodput. In summary, it can be said that a higher bandwidth leads to a better user experience. Overall, we conclude that the network bandwidth is a significant influence factor on initial delay, number of quality switches, and average goodput.

5.5 | Physical Layer: Influence of Signal Coverage on Streaming QoE

In this section we investigate the effect of signal strength on the video streaming performance. Here again, all browsers and protocols are analyzed jointly. We leverage all 4G measurements with respect to the following Reference Signal Received Power

TABLE 6 Influence of the signal strength on performance metrics.

Signal Strength	Initial delay	$\text{Buffer}_{\text{median}}$	$\text{Quality}_{\text{start}}$	$\text{Quality}_{\text{most used}}$	$\text{Quality}_{\text{max}}$	#Switches	Goodput
Excellent	9.84s	46.44s	720p	480p	720p	1.43	3.164$Mbps$
Good	10.16s	45.04s	720p	480p	720p	1.66	3.159$Mbps$
Medium	11.25s	45.61s	720p	480p	720p	1.64	3.066$Mbps$
Low	11.84s	46.07s	720p	480p	720p	1.57	2.918$Mbps$



(a) Initial Delay **(b)** Quality Switches **(c)** Goodput

FIGURE 9 Performance metrics for different signal strength categories.

(RSRP) categories: Measurements using a higher or equal RSRP than $-80dB$ are called *Excellent (E)*, measurements having a RSRP between $-80dB$ and $-90dB$ are grouped to *Good (G)*, measurements with a RSRP between $-90dB$ and $-100dB$ belong to category *Medium (M)*, while measurements with a lower RSRP than $-100dB$ belong to category *Low (L)*. We align our categories according to the mid 20% section of *Table 9.1: RSRP measurement report mapping* in [1].

Table 6 gives an overview of the influence of the signal strength on the selected performance metrics, listing the average of initial delay, median buffer, total number of quality switches, goodput, and the mode of start quality, most used quality, and maximum quality for all measurements. We observe differences in initial delay, number of quality switches, and goodput with respect to signal coverage.

Initial Delay

The maximum difference in terms of initial delay, occurring between signal strength categories *Excellent* and *Low*, is 2 s according to Table 6 . We investigate this behavior in Figure 9 a, where not much difference overall, especially towards the lower end of the delay spectrum. The minimum initial delay is 5.5s, 5.1s, 5.4s, and 5.2s for Excellent, Good, Medium, and Low coverage respectively. However, the effect of signal strength becomes more apparent towards the higher end, with a difference of around 18% (2.5s) at the 85th percentile between Excellent and Low categories. Thus, a negative correlation is noticeable.

Quality Switches

From Table 6 , we see that the differences between the total number of switches for the different signal strength categories are not consistent. However, in order to understand the behavior more clearly and identify characteristics, we show the ECDF of switches using Figure 9 b. We see that overall, the Excellent category shows the best performance (minimum number of switches). However, there is no order among the performance of categories in consistent form, e.g., E-H-M-L. Only the curves for Excellent and Low categories appear in order for both types of switches, with Excellent appearing to the right and showing lower number of switches, and Low appearing to the right. The minimum number of both up and down-switches for all categories is 0. All categories have more up-switches than down-switches on average. Overall minimum (total number of switches) is also 0, where the maximum is 5 for all categories. The mode of the total number of switches is 1 for all categories.

TABLE 7 Influence of the parameters from different layers on selected streaming performance metrics (* for correlation, - for no correlation).

	Initial delay	$\text{Buffer}_{\text{median}}$	$\text{Quality}_{\text{start}}$	$\text{Quality}_{\text{most used}}$	$\text{Quality}_{\text{max}}$	#Switches	Goodput
Application	*	*	-	*	-	*	*
Transport	-	-	-	*	-	*	*
Network	*	-	-	-	-	*	*
Physical	*	-	-	-	-	-	-

Average Goodput

Table 6 shows that in terms of average goodput, the higher signal categories are scarcely distinguishable where the main difference lies between the Low category and the rest. We present the ECDF of the average throughput for all categories in Figure 9 c, from which it is possible to affirm this finding. Especially in the low to mid-range goodput values (0.5-3.0*Mbps*), the curve for the Low category consistently presents itself to the left of all others. Minimum goodput for Excellent, Good, Medium, and Low categories is 1.41, 1.34, 0.42, and 0.39*Mbps*, where the maximum is 4.45, 4.39, 4.92, and 4.40*Mbps*, respectively. Overall, the curves slightly follow the E-G-M-L order, suggesting a weak positive correlation between increased signal strength and increased average goodput. The curves converge to one another towards higher goodput values, indicating that signal coverage might have a more significant influence when the goodput is low.

Discussion

In summary, only for the initial delay a clear influence can be seen. Here, the signal strength is negatively correlated to the initial delay, showing a lower initial delay for better signal strength. Overall, we conclude that the signal coverage is an influence factor on initial delay.

6 | SUMMARY AND DISCUSSIONS

Table 7 summarizes our findings from the measurement campaign and analysis, regarding the influence of different layers on the video streaming QoE metrics. A number of conclusions can be drawn from a network management perspective. First, it is made clear that some of the streaming QoE metrics are closely tied to the Video-on-Demand (VoD) service provider and cannot, for the most part, be influenced by the network operator. These include the start quality of streaming and the maximum quality achieved during streaming. Second, a combination of the effects of multiple layers have an impact on many of the metrics such as initial delay, most used quality during streaming, total number of quality switches, and goodput. Therefore, the performance in these aspects can only be improved with a cross-layer approach.

The largest influence on initial delay comes from the application layer, and more specifically the selection of the browser, mainly due to the type of encoding in use. This might also be influenced by the VoD service provider, since browsers have to adapt their behavior in accordance with what is available from the service. The buffer is mainly influenced by the application layer, with little difference with respect to transport protocol, network bandwidth, or signal coverage. The most used quality is influenced by both the application and the transport protocol in use, suggesting that it might be possible for network operators to improve the end-user experience through appropriate deployment of PEPs in their network. The number of switches is affected by all layers except physical, with the most clear influence shown with respect to network bandwidth, indicating that provisioning for higher available bandwidth can help network operators attain high streaming QoE. Goodput during streaming, largely representative of the selection of different quality levels for the whole duration, is again influenced by all layers except physical, and can be improved by the network operators with sufficient provisioning.

However, the main outcome of this study is an understanding of the following: no single layer in itself operates independently in the complex mobile ecosystem of today. Therefore, individual improvements or per-layer provisioning cannot improve the QoE of video streaming for end-users alone. Interested parties require a paradigm shift from Quality of Service (QoS)-centric approaches, such as benchmarking operators with “speedtest” tools and employing these as a monitoring mechanism for end-user experience (employed as a common practice by operators and regulators alike), towards QoE-centric approaches which take into account complex cross-layer dynamics.

Despite its scale and holistic approach, our study has a number of limitations. Our main shortcomings were the necessity of *black-box experimentation* with respect to YouTube's streaming strategy and internal adaptation decisions, inability to actively control all parameters during measurements (such as network bandwidth and signal coverage) due to experimentation in *operational* MBB networks, and the challenges in attaining a large-scale in measurements while preserving fairness and comparability with respect to parameters under investigation.

7 | RELATED WORK

In recent years, many studies on video streaming have been published. Studies related to our paper can be broadly grouped into four categories: (1) QoE and application layer measurements, (2) related work on the MPEG DASH video streaming standard, (3) DASH QoE studies, (4) existing tools for measuring application performance towards QoE, and (5) large-scale video streaming measurements.

The following works deal with video streaming measurements [2, 7, 15, 31, 40, 49]. They typically focus on a variety of individual aspects such as smartphone performance [7], content caching [2], end user experience [7, 49], or content distribution [31]. Usually, exactly one layer of the network stack is in focus. In contrast to these measurement designs, in our work, we investigate the influence of different layers on streaming performance. Our measurement framework allows to focus not only on the application layer, but also transport, network, and physical layer.

There is also a lot of research on adaptive video streaming. Studies concerning the influence of active adaptation on QoE precede the proliferation of HTTP adaptive streaming, as [54]. In addition to playback quality and startup delay, the dominance of playback interrupts, i.e. stallings, is important for QoE degradations, as observed in [30]. It was found that the influence of startup delay on QoE depends highly on user expectations arising from the performance promised by the network they use [24]. There are also several studies analyzing QoE for mobile networks and smartphones with endpoint measurements and crowdsourcing measurements [6, 7, 17, 18, 21]. For the QoE measurement, ITU-T recommends two quality assessment models, PEVQ-S and Hybrid-FR model, as a result of their competition for a hybrid QoE overall assessment tool [20].

Different approaches predict the QoE of mobile users using passive in-network and in-device measurements by applying machine learning techniques to obtain mappings between QoS and QoE [3, 8, 37]. A comprehensive survey on adaptive video streaming QoE is provided in [16, 22, 45]. In [53], the authors examine several HAS video-on-demand services on mobile networks and [41] characterizes the traffic of Netflix and YouTube. The authors of [46] developed an Android QoE monitoring application which allows to use YouTube and passively measures streaming parameters on the application layer.

There already exist some tools for measuring application performance or QoE, which collect QoE-relevant KPIs on endpoints. YoMo [48] and YoMoApp [50, 52] are passive measurements tools in which a YouTube video playback is monitored and performance indicators are collected. The web browser plug-in YouSlow [35]is able to detect and report live buffer stalling events. A system for on-line monitoring of YouTube QoE in cellular networks is described in [9]. Compared to these tools, our measurement approach doesn't run in the background, it actively runs measurements and monitors the KPIs. This allows for larger and more controlled measurement datasets to be generated. The tool QoE Doctor [10] measures and analyzes mobile app QoE, also based on active measurements. In contrast to our work, this tool is not specialized for one application, it can be used for different applications. Thus, it only evaluates the user-perceived latency, mobile data consumption, and the energy consumption, but no other streaming QoE metrics like the used video quality or buffer.

Another challenge is to conduct large-scale measurement studies in several countries, so that the measurement results become generally valid. Unfortunately, a few existing measurement studies are outdated and do not take into account current trends in streaming such as adaptivity [2, 15]. Nevertheless, the most of related papers to our work might be [15, 53]. The first one takes user satisfaction on YouTube into account with device-level and factors in the network, and [53] looks at several HAS services in mobile networks.

8 | CONCLUSION

In this study, we investigate the performance of YouTube video streaming in MBB networks with a holistic approach, considering the impact of parameters from different layers of the TCP/IP stack on selected QoE metrics such as initial delay, buffer, quality levels, and goodput. Our contributions are as follows: first, we present a novel approach to performance evaluation, considering the influence of different layers on video streaming QoE. We design and implement a tool called *VideoMon* which is capable

of dissecting the impact of different parameters from the said layers and associating them with relevant QoE metrics. Third, we run a large-scale measurement campaign using the prominent European MBB testbed MONROE, comprising over 1,800 experiments, and collect results and metadata from 12 mobile networks, with respect to different browsers, transport protocols, network bandwidth, and signal coverage. Fourth, we evaluate these results from a network management perspective and quantify the impact of each individual layer on selected QoE metrics. Finally, we provide our source code and measurements as open data, so that our results can be reproduced, and further analyses can be conducted by the research community.

Our evaluation shows that there is a complex interplay of different layers on performance with respect to most of the selected QoE metrics, network management practices need to be implemented in a cross-layer fashion, jointly optimizing parameters at multiple layers.

As future work, we plan to conduct a more detailed analysis of the influence of transport layer on video streaming performance, focusing on the effects of the QUIC protocol, and extending the scope of our study towards additional scenarios including high-speed mobility, and different content types.

ACKNOWLEDGMENTS

This work is funded in the framework of the EU ICT project MONROE (H2020-2014-ICT-644399 and the open call project Mobi-QoE), and by the Norwegian Research Council project No. 250679 (MEMBRANE). The authors alone are responsible for the content.

CONFLICT OF INTEREST

We have following conflicts of interest: Florian Metzger, Stefan Valentin.

References

[1] 3GPP TS 136.133 (Release 12). *https://www.etsi.org/deliver/etsi_ts/136100_136199/136133/09.16.00_60/ts_136133v091600p.pdf*.

[2] V. K. Adhikari, S. Jain, and Z.-L. Zhang. “YouTube Traffic Dynamics and its Interplay with a Tier-1 ISP: an ISP Perspective”. In: *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. ACM. 2010, pp. 431–443.

[3] V. Aggarwal, E. Halepovic, J. Pang, S. Venkataraman, and H. Yan. “Prometheus: Toward Quality-of-Experience Estimation for Mobile Apps from Passive Network Measurements”. In: *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications (HotMobile ’14)* (2014).

[4] Ö. Alay, A. Lutu, M. Peón-Quirós, V. Mancuso, T. Hirsch, K. Evensen, A. Hansen, S. Alfredsson, J. Karlsson, A. Brunstrom, et al. “Experience: An Open Platform for Experimentation with Commercial Mobile Broadband Networks”. In: *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. ACM. 2017, pp. 70–78.

[5] Ö. Alay, A. Lutu, R. Garcia, M. Peon-Quiros, V. Mancuso, T. Hirsch, T. Dely, J. Werme, K. Evensen, A. Hansen, S. Alfredsson, J. Karlsson, A. B. A. S. Khatouni, M. Mellia, M. A. Marsan, R. Monno, and H. Lonsethagen. “Measuring and Assessing Mobile Broadband Networks with MONROE”. In: *IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)* (2016).

[6] P. Casas, R. Schatz, F. Wamser, M. Seufert, and R. Irmer. “Exploring QoE in Cellular Networks: How Much Bandwidth do you Need for Popular Smartphone Apps?” In: *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges (AllThingsCellular ’15)* (2015).

[7] P. Casas, M. Seufert, F. Wamser, B. Gardlo, A. Sackl, and R. Schatz. “Next to You: Monitoring Quality of Experience in Cellular Networks From the End-Devices”. In: *IEEE Transactions on Network and Service Management* 13.2 (2016).

[8] P. Casas, A. D'Alconzo, F. Wamser, M. Seufert, B. Gardlo, A. Schwind, P. Tran-Gia, and R. Schatz. "Predicting QoE in Cellular Networks using Machine Learning and in-Smartphone Measurements". In: *Ninth International Conference on Quality of Multimedia Experience (QoMEX)* (2017).

[9] P. Casas, M. Seufert, and R. Schatz. "YOUQMON: A System for On-line Monitoring of YouTube QoE in Operational 3G Networks". In: *ACM SIGMETRICS Performance Evaluation Review* 41 (2013).

[10] Q. A. Chen, H. Luo, S. Rosen, Z. M. Mao, K. Iyer, J. Hui, K. Sontineni, and K. Lau. "QoE Doctor: Diagnosing Mobile App QoE with Automated UI Control and Cross-layer Analysis". In: *Proceedings of the 2014 Conference on Internet Measurement Conference (IMC '14)* (2014).

[11] *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021*. San Jose, CA, USA. Mar. 2017.

[12] M. Dye, R. McDonald, and A. Rufi. *Network Fundamentals, CCNA Exploration Companion Guide*. Cisco Press, 2007.

[13] *Dynamic Adaptive Streaming over HTTP (DASH)*. Standard. ISO/IEC, May 2014.

[14] J. Esteban, S. A. Benno, A. Beck, Y. Guo, V. Hilt, and I. Rimac. "Interactions Between HTTP Adaptive Streaming and TCP". In: *Proceedings of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video*. ACM. 2012, pp. 21–26.

[15] 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 on Internet Measurement Conference (IMC '11)*. ACM. 2011, pp. 345–360.

[16] M. Haddad, E. Altman, R. El-Azouzi, T. Jiménez, S. E. Elayoubi, S. B. Jamaa, A. Legout, and A. Rao. "A Survey on YouTube Streaming Service". In: (2011), pp. 300–305.

[17] T. Hoßfeld, M. Seufert, C. Sieber, T. Zinner, and P. Tran-Gia. "Identifying QoE Optimal Adaptation of HTTP Adaptive Streaming Based on Subjective Studies". In: *Computer Networks* 81 (2015), pp. 320–332.

[18] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz. "Quantification of YouTube QoE via Crowdsourcing". In: *ISM* (2011).

[19] internetlivestats.com. *74,570 YouTube videos viewed in 1 second*. July 2018.

[20] *J.343.6 : Hybrid-FR objective perceptual video quality measurement for HDTV and multimedia IP-based video services in the presence of a full reference signal and non-encrypted bitstream data*. 2014.

[21] P. Juluri, V. Tamarapalli, and D. Medhi. "Measurement of Quality of Experience of Video-on-Demand Services: A Survey". In: *IEEE Communications Surveys & Tutorials* 18.1 (), pp. 401–418.

[22] T. Karagkioules, C. Concolato, D. Tsilimantos, and S. Valentin. "A Comparative Case Study of HTTP Adaptive Streaming Algorithms in Mobile Networks". In: *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM. 2017, pp. 1–6.

[23] A. S. Khatouni, M. Mellia, M. A. Marsan, S. Alfredsson, J. Karlsson, A. Brunstrom, Ö. Alay, A. Lutu, C. Midoglu, and V. Mancuso. "Speedtest-like Measurements in 3G/4G Networks: The MONROE Experience". In: *29th International Teletraffic Congress (ITC 29)*. Vol. 1. IEEE. 2017, pp. 169–177.

[24] S. S. Krishnan and R. K. Sitaraman. "Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-experimental Designs". In: *Proceedings of the 2012 Internet Measurement Conference*. IMC '12. New York, NY, USA: ACM, 2012, pp. 211–224.

[25] S. Lederer. *Why YouTube & Netflix use MPEG-DASH in HTML5*. Tech. rep. Feb. 2015.

[26] C. Midoglu, L. Wimmer, A. Lutu, Ö. Alay, and C. Griwodz. "MONROE-Nettest: A Configurable Tool for Dissecting Speed Measurements in Mobile Broadband Networks". In: *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2017).

[27] C. Midoglu, K. Kousias, C. Griwodz, and Ö. Alay. *Evaluation of DASH Rate Adaptation Algorithms in Operational Mobile Networks*. Poster at TMA PhD School. 2017.

[28] C. Midoglu, M. Moulay, V. Mancuso, Ö. Alay, A. Lutu, and C. Griwodz. "Open Video Datasets over Operational Mobile Networks with MONROE". In: *Proceedings of the 9th ACM Multimedia Systems Conference (MMSys '18)* (2018).

[29] C. Midoglu, A. Schwind, and F. Wamser. *VideoMon Container*. https://github.com/VideoMon/VideoMon-IJNM2018. 2018.

[30] R. K. Mok, E. W. Chan, X. Luo, and R. K. Chang. "Inferring the QoE of HTTP Video Streaming from User-viewing Activities". In: *Proceedings of the First ACM SIGCOMM Workshop on Measurements up the Stack (W-MUST '11*. New York, New York, USA: ACM Press, 2011, p. 31.

[31] R. K. Mok, V. Bajpai, A. Dhamdhere, and K. Claffy. "Revealing the Load-Balancing Behavior of YouTube Traffic on Interdomain Links". In: *International Conference on Passive and Active Network Measurement*. Springer. 2018, pp. 228–240.

[32] MONROE-Nettest Core. *https://github.com/MONROE-PROJECT/Experiments/tree/master/experiments/nettest/nettest-client/nettest-core*.

[33] MONROE Web Scheduler. *https://www.monroe-system.eu/*.

[34] M. Moulay and V. Mancuso. "Experimental Performance Evaluation of WebRTC Video Services over Mobile Networks". In: *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2018).

[35] H. Nam, K. Kim, H. Schulzrinne, and D. Calin. "YouSlow: A Performance Analysis Tool for Adaptive Bitrate Video Streaming". In: *Proceedings of the 2014 ACM Conference on SIGCOMM* (2014).

[36] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov. "A Machine Learning Approach to Classifying YouTube QoE Based on Encrypted Network Traffic". In: *Multimedia Tools and Applications* 76.21 (2017), pp. 22267–22301.

[37] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov. "YouTube QoE Estimation Based on the Analysis of Encrypted Network Traffic Using Machine Learning". In: *5th IEEE International Workshop on Quality of Experience for Multimedia Communications (QoEMC)*. Washington, DC, USA, 2016.

[38] R. Pantos and W. May. *HTTP Live Streaming*. Tech. rep. IETF, Sept. 2011.

[39] M. Peón-Quirós, V. Mancuso, V. Comite, A. Lutu, Ö. Alay, S. Alfredsson, J. Karlsson, A. Brunstrom, M. Mellia, A. Safari Khatouni, et al. "Results from Running an Experiment as a Service Platform for Mobile Networks". In: *Proceedings of the 11th Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization*. ACM. 2017, pp. 9–16.

[40] L. Plissonneau, E. Biersack, and P. Juluri. "Analyzing the Impact of YouTube Delivery Policies on User Experience". In: *Proceedings of the 24th International Teletraffic Congress*. International Teletraffic Congress. 2012, p. 28.

[41] 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, p. 25.

[42] RMBT Documentation. *https://www.netztest.at/doc/*.

[43] Sandvine. *2016 Global Internet Phenomena: Latin America & North America*. Tech. rep. June 2016.

[44] A. Schwind, C. Midoglu, Ö. Alay, C. Griwodz, and F. Wamser. *VideoMon Dataset*. https://zenodo.org/record/1415521. 2018.

[45] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia. "A Survey on Quality of Experience of HTTP Adaptive Streaming". In: *IEEE Communications Surveys Tutorials* 17.1 (2015).

[46] M. Seufert, P. Casas, F. Wamser, N. Wehner, R. Schatz, and P. Tran-Gia. "Application-layer Monitoring of QoE Parameters for Mobile YouTube Video Streaming in the Field". In: *IEEE Sixth International Conference on Communications and Electronics (ICCE)* (2016).

[47] M. Seufert, T. Hoßfeld, and C. Sieber. "Impact of Intermediate Layer on Quality of Experience of HTTP Adaptive Streaming". In: *11th International Conference on Network and Service Management (CNSM)* (2015), pp. 256–260.

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

[49] F. Wamser, P. Casas, M. Seufert, C. Moldovan, P. Tran-Gia, and T. Hoßfeld. "Modeling the YouTube Stack: From Packets to Quality of Experience". In: *Computer Networks* 109 (2016), pp. 211–224.

[50] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz. “Poster: Understanding YouTube QoE in Cellular Networks with YoMoApp - a QoE Monitoring Tool for YouTube Mobile”. In: *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom ’15)* (2015).

[51] F. Wamser, R. Pries, D. Staehle, K. Heck, and P. Tran-Gia. “Traffic Characterization of a Residential Wireless Internet Access”. In: *Telecommunication Systems* 48.1-2 (2011), pp. 5–17.

[52] 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: *European Conference on Networks and Communications (EuCNC)* (2015).

[53] S. Xu, S. Sen, Z. M. Mao, and Y. Jia. “Dissecting VOD Services for Cellular: Performance, Root Causes and Best Practices”. In: *Proceedings of the 2017 Internet Measurement Conference (IMC ’17)* (2017), pp. 220–234.

[54] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz. “Subjective Impression of Variations in Layer Encoded Videos”. In: *Proceedings of the 11th International Conference on Quality of Service (IWQoS’03)*. 2003, pp. 137–154.