Realistic Video Sequences for Subjective QoE Analysis

Kerim Hodzic, Mirsad Cosovic, Sasa Mrdovic
Faculty of Electrical Engineering, University of Sarajevo
BiH
{kerim.hodzic,mcosovic,smrdovic}@etf.unsa.ba

Jason J. Quinlan
School of Computer Science & Information Technology, University College Cork
Ireland
j.quinlan@cs.ucc.ie

Darijo Raca
Faculty of Electrical Engineering, University of Sarajevo
BiH
draca@etf.unsa.ba

ABSTRACT

Multimedia streaming over the Internet (live and on demand) is the cornerstone of modern Internet carrying more than 60% of all traffic. With such high demand, delivering outstanding user experience is a crucial and challenging task. To evaluate user Quality of Experience (QoE) many researchers deploy subjective quality assessments where participants watch and rate videos artificially infused with various temporal and spatial impairments. To aid current efforts in bridging the gap between the mapping of objective video QoE metrics to user experience, we developed DashReStreamer, an open-source framework for re-creating adaptively streamed video in real networks. DashReStreamer utilises a log created by a HTTP adaptive streaming (HAS) algorithm run in an uncontrolled environment (i.e., wired or wireless networks), encoding visual changes and stall events in one video file. These videos are applicable for subjective QoE evaluation mimicking realistic network conditions.

To supplement DashReStreamer, we re-create 234 realistic video clips, based on video logs collected from real mobile and wireless networks. In addition our dataset contains both video logs with all decisions made by the HAS algorithm and network bandwidth profile illustrating throughput distribution. We believe this dataset and framework will permit other researchers in their pursuit for the final frontier in understanding the impact of video QoE dynamics.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Networks** → **Public Internet**; **Wireless access networks**.

KEYWORDS

QoE, Dataset, Mobility, throughput, context information, adaptive video streaming, 3G, 4G, WiFi

ACM Reference Format:
Kerim Hodzic, Mirsad Cosovic, Sasa Mrdovic, Jason J. Quinlan, and Darijo Raca. 2022. Realistic Video Sequences for Subjective QoE Analysis. In *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3524273.3532894

MMSys '22, June 14–17, 2022, Athlone, Ireland

ACM ISBN 978-1-4503-9283-9/22/06…$15.00
https://doi.org/10.1145/3524273.3532894

1 INTRODUCTION

In its early days, the Internet was conceived with the idea for fast and reliable information sharing between many remote users. Since then, the Internet has transformed beyond basic e-mail communication, becoming one of the key pillars of modern society with multimedia entertainment at its heart, representing the dominant type of traffic carried over today's networks.

Video streaming depicts the main driver behind multimedia entertainment, accounting for almost 60% of all Internet traffic in 2020. Furthermore, fuelled by a recent pandemic outbreak, forcing people to stay home, video traffic grew over the last two years, with applications such as YouTube, Netflix, Amazon Prime, Disney+ and Apple+ dominating overall traffic share [16].

The popularity of video streaming services led to user demand for high QoE of delivered content. By definition, QoE represents the magnitude of annoyance or the delight of a user's experience with an application or service [3]. However, measuring and modelling user QoE is challenging due to its subjective intrinsic component. The challenge lies in modelling impairments that contribute to total QoE score. These impairments include initial delay, stall events, average quality, switching frequency, and video duration [11]. Finding an optimal combination of these impairments to map to QoE score is not a trivial task. Common approach includes performing subjective studies devising weights for each of the impairments [5, 11, 13]. Many adaptive algorithms rely on these derived QoE models, using them as an objective function in designing adaptation logic [23, 24]. On the networks side, vendors usually rely on network metrics, such as packet loss and utilisation to map to user QoE.

The subjective evaluation of QoE represents a foundation for better understanding and modelling user experience. Few studies perform both subjective and objective QoE evaluation [5–7, 11, 18]. To estimate subjective experience, researchers design a few test sequences containing video impairments. Typically, these impairments are added artificially to the video sequence [11, 22]. However, in literature there are many datasets with bandwidth traces collected in various mobile environments under different wireless technologies [10, 15]. These datasets can be used for obtaining objective performance of adaptation algorithms including rate distribution, stall duration, and stall occurrence. Generating test video sequences based on realistic video logs complement the current literature on QoE. To the best of our knowledge, there are no datasets generated based on real traffic patterns available to the research community.

Motivated by this observation, we offer a framework for creating video sequences based on video logs collected either in real network or based on realistic bandwidth traces. Furthermore, we provide

234 video sequences based on video logs analysed over different bandwidth profiles collected from various wireless networks [17]. Video logs were generated by HAS streaming algorithms under bandwidth profiles from different networks, resulting in a realistic snapshot of decisions algorithms made, including bitrate decisions (giving us rate distribution) and stall events (number and duration of stalls).

In this paper, we present DashReStreamer [1], a framework for generating test video sequences with encoded stall and rate changes. In addition to the framework, we provide an extensive dataset containing video sequences created over 3G, 4G and WiFi networks. In total, 234 video sequences were generated with a duration of 5 minutes[2]. The dataset contains video logs and bandwidth traces used for the video sequence generation. These video sequences are suitable for subjective QoE evaluation, and can aid in the better understanding of user experience in different scenarios. To the best of our knowledge, our QoE dataset is the first publicly available dataset that contains video sequence, logs, and bandwidth traces.

The remainder of this paper is organised as follows. Section 2 describes related work regarding similar datasets and QoE-related video metrics. The overview and key features of proposed framework are explained in Section 3, while Section 4 provides an overview of the dataset generated by DashReStreamer. In Section 5 we layout future work, while Section 6 outlines our conclusion.

2 BACKGROUND AND RELATED WORK

The main goal of HAS algorithms is maximising user perceived QoE. This daunting task relies on accurate representation of subjective impact through mapping objective Quality of Service (QoS) metrics at client side (e.g., initial delay, average bitrate, re-buffering events, and switching frequency) or metrics measured at the network such as utilisation and packet-loss rate. Also, the majority of proposed HAS algorithms in literature relies on QoE models to quantitatively compare its performance to existing state-of-the art HAS algorithms. Furthermore, QoE models expressed as linear combination of impairments (1), represent a suitable candidate for designing a HAS algorithm that maximises a given QoE model. A typical approach includes modelling the QoE model as the utility function of the optimisation problem [2, 23, 24].

A typical template equation used for deriving QoE model is [5, 11, 13]:

$$\mathrm{QoE}_s = w_o \cdot \mathrm{QoE}_m - (w_t \cdot I_t + w_v \cdot I_v) + f(I_t, I_v), \qquad (1)$$

where I_t represents temporal impairment factor, and w_t represents its weight. Temporal quality impairments indicate degradation due to initial delay and stall performance (stall number and stall duration). While initial delay has a minor negative effect on QoE (up to 16 seconds), stall events have the highest negative impact on overall user experience [19]. I_v, and w_v represent visual quality impairment factors and its weight, respectively.

Average bitrate and switching behaviour model visual quality impairments. Similar to stall performance, bitrate quality amplitude has a significant effect on QoE [8], unlike switching between different qualities while retaining the same resolution [8]. However, switching between different resolutions can influence user experience [1]. QoE_m depicts the maximum (initial) value (score) for QoE or growth factor depending on the QoE model, and w_o denotes a weight for the QoE_m score. Some QoE models take into account impairments that occur simultaneously. In these scenarios, aggregate subjective effect is not a direct sum of each impairment [11]. The role of function $f(I_t, I_v)$ is to compensate for this effect.

However, these impairments (i.e., metrics) are mutually contradictory. High bitrate increases the chance of buffer underflow resulting in stall events, while streaming at low bitrate quality has a severe negative impact on perceived user experience.

To capture the mapping between user perceived experience and objective metrics, many studies use subjective evaluation. This evaluation relies on assessing video quality by participants in a controlled lab environment [4, 11, 13, 20]. Each participant rates a video sequence on a 100-point scale (denoted as R, where some studies use 5 or 10-point scale). The procedure is repeated for a series of test sequences. Each test sequence is embellished with one or more impairments. Finally, for each test sequence and given score R, the impairment impact is calculated as 100-R.

Subjective evaluation is an expensive, time-consuming process performed with a limited number of human subjects (usually around 30) restricting the statistical validity of collected results. Alternatively, some studies opt for a crowd-sourcing approach, where a large number of users rate video sequences online in an uncontrolled environment [5, 9, 22].

The main challenge for subjective evaluation is augmentation of the test video sequences with particular impairments. Typically, these impairments are artificially created and added to video clips. However, artificially created impairments do not necessarily reflect impairments observed in real network conditions, either their frequency (e.g., number of rate switches, number of stalls), or duration (e.g., stall duration).

There are a plethora of bandwidth datasets collected in real networks available in literature [10, 14, 15]. These datasets reflects real conditions observed in networks and can be leveraged for realistic creation of temporal and visual impairments.

Motivated by the lack of video sequences with the impairments based on real network conditions, we designed a tool for creating video sequences with impairments collected from video sessions collected over realistic bandwidth traces. We believe this dataset will aid in ongoing research to better understanding factors affecting user experience.

3 DashReStreamer OVERVIEW

DashReStreamer provides the functionality to reproduce network impact on video player performances by creating video clips including all resolution changes and re-buffering events. We achieve this functionality by utilising video logs generated by the client during the original stream of content in an uncontrolled environment (i.e., real production network).

Typically these logs include various information related to HAS QoS metrics (e.g., bitrate, switches and stall information). To illustrate, Table 1 depicts an example of a video log.

From the DashReStreamer perspective, three features are necessary for video clips generation. These features are details on:

[1] https://github.com/khodzic2/DashReStreamer
[2] https://shorturl.at/dtISV

Table 1: Sample output from the video log

Type	Description	Unit
Seg_#	Streamed segment number	-
Arr_Time	Arrival time	ms
Del_Time	Time taken to receive the segment	ms
Stall_Dur	Stall duration	ms
Rep_Level	Representation Quality	kbps
Del_Rate	Delivery rate	kbps
Act_Rate	Actual rate	kbps
Byte_Size	Size of segment	byte
Buffer_Level	Buffer level	ms

- Segment number.
- Segment bitrate: we use this information to select the subsets of downloaded segments during playback (currently DashReStreamer does not support byte-range HAS content).
- Stall events: we use stall events (occurrence and duration) to add stalls (e.g., duplicating last frame of segment) at the end of segments affected by re-buffering events.

3.1 Framework Implementation

We use the Python programming language and FFmpeg[3] library for the implementation of DashReStreamer. FFmpeg is a cross-platform multimedia framework for transforming (i.e., encoding, decoding, transcoding, mux, demux, stream, and filter) a wide range of media formats (video and image).

DashReStreamer starts by parsing video log file, where used bitrates of video segments are identified for further processing. Two methods are used for this process:

- *read_replevels_log* - takes four arguments (file path, index column name, bitrate column name, and type of separator, e.g., csv), parses the file and returns the output as a hash function (dictionary) storing the bitrate for each downloaded segment.
- *read_stalls_log* - takes four arguments and returns the position and duration of each stall, where position is related to segment when the stall happened.

The next step includes filtering a subset of streamed segments. Segments can be stored locally or remotely on a web server. In the latter case, an mpd file is used for downloading the streamed segments from the server to the local machine.

Three methods are used for the manipulation of the streamed segments (for the case when segments are already stored locally):

- *copy_init_file* - takes two arguments, location of init mp4 file and destination where init file will be stored for further processing.
- *copy_video_segments* and *copy_audio_segments* - are methods for copying downloaded segments for further processing. Similar to *copy_init_file* method, these methods take two arguments. These methods rely on a dictionary created in the previous step by the *read_replevels_log* and *read_stalls_log* methods for appropriate identification of streamed segments and stall events.

For the case when segments are directly downloaded from a web server, the script uses the location of the mpd (i.e., URL) to retrieve only the subset of segments streamed in the video logs. This procedure is similar to the behaviour of traditional HAS client [21] (without actual decoding of the data). We use an existing library for parsing mpd files[4]. There are three main methods for preparing the data linking in this step:

- *parse_mpd* - the method that parses mpd files and stores urls of audio and video segments into a dictionary,
- *download_video_segments* - takes two arguments, location of mpd file and destination folder,
- *download_audio_segments* - similar to the previous method, method downloads audio segments.

DashReStreamer proceeds by combining segments with init file (originally segments are in m4s format). For this operation we use two methods:

- *prepare_video_init* - takes two arguments, location of video segments and init file,
- *prepare_audio_init* - prepares audio segments similar to the previous method.

The output of these methods are new audio and video segments (in avi[5] and mkv[6] format respectively) which can be played independently. Next, we combine the individual pairs of audio and video segments, using the FFmpeg library. This operation is performed by method *concat_audio_video_ffmpeg* which takes two arguments: location of segments and flag indicating should segment be rescaled to different resolution.

We create a video sequence combining segments including all bitrate/resolution changes and stall events. First, we create stall-induced segments. For the creation of stall-induced segments, we take the stall duration and the segment just before stall starts. We take the last frame of the identified segment, and add them at the end of segment for the duration of the stall. Finally, we add gif[7] as an overlay on top of the stall-induced segments. After all segments are prepared, we join them into a final mkv video file. The preceding logic is implemented in method *concat_audio_video_ffmpeg_final*. This method takes three arguments: location of segments, location of gif and destination for final video.

3.2 Example of Use

There are several options available to run DashReStreamer, either directly through the command line or using a configuration file. For command line use, Table 2 depicts the supported options for running the framework.

Case #1: For segment files stored locally, the command outlined in Listing 1 produces a video file based on the video log file.

```
# python video_log_merger.py --path_to_log video_log.log
--rep_lvl_col Rep_Level
--seg_index_col Chunk_Index
--log_separator tab
```

[3]https://www.ffmpeg.org/

[4]https://github.com/sangwonl/python-mpegdash

[5]Audio Video Interleave

[6]Matroska Multimedia Container

[7]Graphics Interchange Format

Table 2: Options for running QoE framework

Parameter	Description
path_to_log	Location of video log
rep_lvl_col	Column name used in video log for bitrate
seg_index_col	Column name used in video log for segment index
stall_dur_col	Column name used in video log for stall duration
log_separator	Separator used in video log (example: tab)
config_path	Location of config file
path_video	Location of video segments
path_audio	Location of audio segments
gif_path	Location of gif file
log_location	Flag indicating location of segments (local or remote)
dest_video	Location where to save intermediate files during processing (segments)
final_path	Location where final concated video is saved
parameter_type	Flag indicating use of command line arguments or config file
cleanup	Flag indicating removal of intermediate files (segments)
auto_scale	Options for enabling auto-scaling of segment resolution
scale_res	Rescaling segments to predetermined resolution (example: 1080p)

```
--stall_dur_col Stall_Dur
--path_video ./sintel/DASH_Files/full/
--dest_video ./tmp_files/
--path_audio ./sintel/DASH_Files/audio/full/
--gif_path ./gif.gif
--final_path ./final/ --parameter_type path
--cleanup True
```

Listing 1: Example of creating video from local segments

The depicted example in in Listing 1 utilises the open-source movie Sintel, filters segment qualities used by adaptation algorithm outlined by video log file (video_log.log file), re-creates video sequence adding stall events (with the re-buffering image) and saves the output to the folder final. This command retains native resolution for each segment causing a visual change in aspect ratio when segments of the video switch from one resolution to another.

Alternatively, we can mandate that all segments have the same output resolution through the option of autoscaling. We support two types of autoscaling: scaling to the highest resolution observed in the log file, or scaling to predetermined resolution given by parameter *scale_res*. The Listing 2 example shows how to create an output video file with a fixed 1080p resolution for all segments.
Case #2: Creating video file with same predetermined resolution is depicted in Listing 2

```
# python video_log_merger.py --path_to_log video_log.log
--rep_lvl_col Rep_Level
--seg_index_col Chunk_Index
--log_separator tab
--stall_dur_col Stall_Dur
--path_video ./sintel/DASH_Files/full/
--dest_video ./tmp_files/
--path_audio ./sintel/DASH_Files/audio/full/
--gif_path ./gif.gif
--final_path ./final/ --parameter_type path
--scale_resolution 1080p
--auto_scale 2
--cleanup True
```

Listing 2: Example of creating video with same resolution for all segments

Similar to Listing 1, we recreate an output video clip from the video log file, with the difference that we scale each segment to a Full HD resolution. This option is achieved by setting auto_scale to 2 (where we have three supported values 0,1,2), and setting scale_res to 1080p.

The DASHReStreamer framework also supports the use of a configuration file as input to the python script. Listing 3 illustrates an example configuration file. Note that all the input parameters are the same as the parameters used for the command-line input.

```
[parameters]
parameters = config
path_to_log = <path>
rep_lvl_column = Rep_Level
chunk_index_column = Chunk_Index
stall_dur_column = Stall_Dur
log_separator = tab
path_audio = <path to audio segments>
path_video = <path to video segments>
dest_video = <where to save/download segments>
gif_path = <path to gif file>
final_path = <where to save final video>
mpd_path = <url for mpd file>
auto_scale = 0
log_location = local
```

Listing 3: Example of config file

4 QOE DATASET OVERVIEW

This section gives a short overview of the dataset used for generating various video sequences in different wireless conditions. The majority of the video sequences contain at least one re-buffering event as those cases are the most interesting for QoE modelling.

4.1 Video Logs Generation

We use video logs generated by experiments in [17] for the creation of the video sequences. The video logs are generated based on bandwidth traces collected from real operational networks. Figure 1 illustrates a generalised testbed used for producing video logs.

The testbed consists of a server machine, an intermediate device, e.g., Wireless Acess Point (WAP) and one or more wireless-capable end devices (i.e., mobile device). The server machine performs two roles, one as web server for video content, and second as traffic

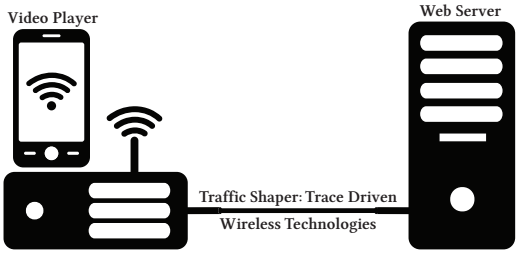

Figure 1: The data-driven generation testbed.

shaper for link between server and intermediate device. The traffic shaping procedure includes the use of traffic shaping tools like Linux traffic control (tc) for emulation of different bandwidth profiles from collected bandwidth logs. The intermediate device connects to the end device via WiFi channel. Finally, the end device streams content from the server via a bottleneck link creating the video log after streaming finishes.

A 4K encoded animation clip is used as the video content stored at the server. The clip is encoded, using the H.264/AVC codec, into thirteen bitrates (from 235kbps to 40Mbps) across eight resolutions.

For traffic shaping, bandwidth logs were used from three different wireless technologies, 3G, 4G and WiFi, including various mobility patterns (static, pedestrian, bus, car and tram). Table 3 shows summary statistics (average and standard deviation of measured bandwidth) for these logs [17].

Table 3: Throughput Statistics for collected bandwidth logs

Technology	Average (Mbps)	Standard Deviation (Mbps)
3G	1.26	0.97
4G	11.32	13.17
WiFi	18.71	17.73

4.2 Video Sequences Generation

We utilise video logs explained in the previous section and our proposed tool (see Section 3) to generate 234 video impaired clips. For video content, we select three open-source clips from [12]. These clips are Big Buck Bunny (BBB)[8], Sintel[9], and Tears of Steel (TOS)[10]. Big Buck Bunny is an animation clip with a duration of 10 minutes and 34 seconds. The content is composed of animated characters with a non intricate background, encoded with a maximum 4K resolution of 3840x2160, at 60fps. Similarly, Sintel is an animation clip with a duration of 14 minutes and 48 seconds. The content is composed of complex animated characters and scenery, encoded with a maximum 4K resolution of 3840x2160, at 24fps. Finally, Tears of Steel is a movie-alike clip enhanced with digital visual effects of 12 minutes and 14 seconds duration. The content is composed of real actors and superimposed digital effects, encoded with maximum 4K resolution of 3840x2160, at 24fps.

All clips are encoded in thirteen bitrates and eight different resolutions as depicted in Table 4 and sourced from [12]. Also, all clips are encoded with the sound of five minutes plus total stall duration. We select 27, 25, and 26 video logs generated from 3G, 4G, and WiFi network traces, respectively. Table 5 depicts video QoS metric statistics for the selected logs.

Table 4: Ladder for the average encoding rate, and resolution for the used dataset

No.	Bitrate	Resolution
13	40Mbps	3840x2160
12	25Mbps	3840x2160
11	15Mbps	3840x2160
10	4.3Mbps	1920x1080
9	3.85Mbps	1920x1080
8	3Mbps	1280x582
7	2.35Mbps	1280x582
6	1.75Mbps	720x328
5	1.05Mbps	640x292
4	750kbps	512x234
3	560kbps	512x234
2	375kbps	384x174
1	235kbps	320x146

Video logs from 3G network traces have the highest number of stalls and stall duration followed by 4G and WiFi network traces. This result is intuitive as indicated by throughput statistics in Table 3. Also, WiFi network traces are mostly collected in a static environment thus having the highest average throughput.

5 FUTURE WORK

While our framework currently offers a mechanism to generate an adaptive video dataset, which can be used in subjective testing or similar research settings, typically using a five-point MOS scale, future work will include the calculation of Video Quality Metrics such as PSNR, SSIM, VMAF and P.1203 [13] for each generated clip. Creating additional KPIs through which video QoE and Network QoS can be determined.

Furthermore DashReStreamer currently only supports the *full* profile of the DASH standard. Future work includes adding support for remaining mpd profiles (i.e., main, live, onDemand, and byte range) and other HAS datasets available in the literature. We also plan on adding realistic video clips for different HAS segment durations to our Dataset, including segment durations of between 2 and 10 seconds, allowing for a much richer and diversified QoE video dataset.

6 CONCLUSIONS

In this paper, we present DashReStreamer, an open-source cross-platform framework for reproducing adaptively streamed video from real operational networks. DashReStreamer allows re-creating video clips with all bitrate/quality changes and stall events. Generated video clips mimic decisions made by HAS adaptation algorithms, and the selected bitrates chosen under realistic time-varying conditions observed in the network. The framework utilises video logs produced by HAS adaptation algorithm to re-create video clips. Furthermore we generate 234 video clips mimicking the behaviour

[8]https://peach.blender.org/
[9]https://durian.blender.org/about/
[10]https://mango.blender.org/about/

Table 5: Average QoS metrics for selected video logs

Network	Bitrate (Mbps)	Num. Switches	Num. Stalls	Stall Dur. (s)
3G	1.6	19.6	3.4	53.9
4G	5.8	18.8	0.96	14.3
WiFi	6.3	12.5	0.77	1.95

of various HAS adaptation algorithms under three different wireless technologies (i.e., 3G, 4G, and WiFi), producing a dataset with realistic bitrate changes and stall events. We believe this dataset will help researchers in better understanding factors affecting user experience for HAS multimedia technologies, aiding its use in both objective and subjective QoE evaluation.

ACKNOWLEDGMENTS

The authors acknowledge the support of the Ministry of Education, Science and Youth of Sarajevo Canton, while also acknowledging the support of CSIT, University College Cork, Ireland.

REFERENCES

[1] A. Asan, W. Robitza, I. h. Mkwawa, L. Sun, E. Ifeachor, and A. Raake. 2017. Impact of video resolution changes on QoE for adaptive video streaming. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*. 499–504. https://doi.org/10.1109/ICME.2017.8019297

[2] Abdelhak Bentaleb, Ali C. Begen, Saad Harous, and Roger Zimmermann. 2018. Want to Play DASH? A Game Theoretic Approach for Adaptive Streaming over HTTP. In *Proceedings of the 9th ACM Multimedia Systems Conference* (Amsterdam, Netherlands) *(MMSys '18)*. Association for Computing Machinery, New York, NY, USA, 13–26. https://doi.org/10.1145/3204949.3204961

[3] Kjell Brunnström et. al. 2013. Qualinet White Paper on Definitions of Quality of Experience. https://hal.archives-ouvertes.fr/hal-00977812 Qualinet White Paper on Definitions of Quality of Experience Output from the fifth Qualinet meeting, Novi Sad, March 12, 2013.

[4] Manri Cheon and Jong-Seok Lee. 2018. Subjective and Objective Quality Assessment of Compressed 4K UHD Videos for Immersive Experience. *IEEE Transactions on Circuits and Systems for Video Technology* 28, 7 (2018), 1467–1480. https://doi.org/10.1109/TCSVT.2017.2683504

[5] J. De Vriendt, D. De Vleeschauwer, and D. Robinson. 2013. Model for estimating QoE of video delivered using HTTP adaptive streaming. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. 1288–1293.

[6] Zhengfang Duanmu, Wentao Liu, Zhuoran Li, Diqi Chen, Zhou Wang, Yizhou Wang, and Wen Gao. 2020. Assessing the Quality-of-Experience of Adaptive Bitrate Video Streaming. arXiv:2008.08804 [eess.IV]

[7] Deepti Ghadiyaram, Janice Pan, and Alan C. Bovik. 2019. A Subjective and Objective Study of Stalling Events in Mobile Streaming Videos. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 1 (2019), 183–197. https://doi.org/10.1109/TCSVT.2017.2768542

[8] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner. 2014. Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming. In *2014 Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*. 111–116. https://doi.org/10.1109/QoMEX.2014.6982305

[9] S. Shunmuga Krishnan and Ramesh K. Sitaraman. 2012. Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs. In *Proceedings of the 2012 Internet Measurement Conference* (Boston, Massachusetts, USA) *(IMC '12)*. Association for Computing Machinery, New York, NY, USA, 211–224. https://doi.org/10.1145/2398776.2398799

[10] Li Li, Ke Xu, Dan Wang, Chunyi Peng, Qingyang Xiao, and Rashid Mijumbi. 2015. A measurement study on TCP behaviors in HSPA+ networks on high-speed rails. In *2015 IEEE Conference on Computer Communications (INFOCOM)*. 2731–2739. https://doi.org/10.1109/INFOCOM.2015.7218665

[11] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao. 2015. Deriving and Validating User Experience Model for DASH Video Streaming. *IEEE Transactions on Broadcasting* 61, 4 (Dec 2015), 651–665.

[12] Jason J. Quinlan and Cormac J. Sreenan. 2018. Multi-Profile Ultra High Definition (UHD) AVC and HEVC 4K DASH Datasets. In *Proceedings of the 9th ACM Multimedia Systems Conference* (Amsterdam, Netherlands) *(MMSys '18)*. Association for Computing Machinery, New York, NY, USA, 375–380. https://doi.org/10.1145/3204949.3208130

[13] A. Raake, M. Garcia, W. Robitza, P. List, S. Göring, and B. Feiten. 2017. A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1. In *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. 1–6. https://doi.org/10.1109/QoMEX.2017.7965631

[14] Darijo Raca, Jason J. Quinlan, Ahmed H. Zahran, and Cormac J. Sreenan. 2018. Beyond Throughput: A 4G LTE Dataset with Channel and Context Metrics. In *Proceedings of the 9th ACM Multimedia Systems Conference* (Amsterdam, Netherlands) *(MMSys '18)*. Association for Computing Machinery, New York, NY, USA, 460–465. https://doi.org/10.1145/3204949.3208123

[15] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. 2013. Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications. In *Proceedings of the 4th ACM Multimedia Systems Conference* (Oslo, Norway) *(MMSys '13)*. Association for Computing Machinery, New York, NY, USA, 114–118. https://doi.org/10.1145/2483977.2483991

[16] Sandvine. 2020. *The Global Internet Phenomena Report: COVID-19 Spotlight*. Technical Report.

[17] Yusuf Sani, Darijo Raca, Jason J. Quinlan, and Cormac J. Sreenan. 2020. SMASH: A Supervised Machine Learning Approach to Adaptive Video Streaming over HTTP. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. 1–6. https://doi.org/10.1109/QoMEX48832.2020.9123139

[18] Kalpana Seshadrinathan, Rajiv Soundararajan, Alan Conrad Bovik, and Lawrence K. Cormack. 2010. Study of Subjective and Objective Quality Assessment of Video. *IEEE Transactions on Image Processing* 19, 6 (2010), 1427–1441. https://doi.org/10.1109/TIP.2010.2042111

[19] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hobfeld, and P. Tran-Gia. 2015. A Survey on Quality of Experience of HTTP Adaptive Streaming. *Communications Surveys Tutorials, IEEE* 17, 1 (Firstquarter 2015), 469–492. https://doi.org/10.1109/COMST.2014.2360940

[20] Zaixi Shang, Joshua P. Ebenezer, Alan C. Bovik, Yongjun Wu, Hai Wei, and Sriram Sethuraman. 2021. Assessment of Subjective and Objective Quality of Live Streaming Sports Videos. arXiv:2106.08431 [eess.IV]

[21] Thomas Stockhammer. 2011. Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems* (San Jose, CA, USA) *(MMSys '11)*. Association for Computing Machinery, New York, NY, USA, 133–144. https://doi.org/10.1145/1943552.1943572

[22] Babak Taraghi, Abdelhak Bentaleb, Christian Timmerer, Roger Zimmermann, and Hermann Hellwagner. 2021. Understanding Quality of Experience of Heuristic-Based HTTP Adaptive Bitrate Algorithms. In *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video* (Istanbul, Turkey) *(NOSSDAV '21)*. Association for Computing Machinery, New York, NY, USA, 82–89. https://doi.org/10.1145/3458306.3458875

[23] Praveen Kumar Yadav, Abdelhak Bentaleb, May Lim, Junyi Huang, Wei Tsang Ooi, and Roger Zimmermann. 2021. *Playing Chunk-Transferred DASH Segments at Low Latency with QLive*. Association for Computing Machinery, New York, NY, USA, 51–64. https://doi.org/10.1145/3458305.3463376

[24] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. *SIGCOMM Comput. Commun. Rev.* 45, 4 (Aug. 2015), 14 pages.