DASHbed: a testbed Framework for Large Scale Empirical Evaluation of Real-Time DASH in Wireless Scenarios

Darijo Raca, Yusuf Sani, Cormac J. Sreenan, Jason J. Quinlan
School of Computer Science and Information Technology, University College Cork, Cork, Ireland
{d.raca,ys8,cjs,j.quinlan}@cs.ucc.ie

ABSTRACT

Recent years have witnessed an explosion of multimedia traffic carried over the Internet. Video-on-demand and live streaming services are the most dominant services. To ensure growth, many streaming providers have invested considerable time and effort to keep pace with ever-increasing users' demand for better quality and stall abolition. HTTP adaptive streaming (HAS) algorithms are at the core of every major streaming provider service. Recent years have seen sustained development in HAS algorithms. Currently, to evaluate their proposed solutions, researchers need to create a framework and numerous state-of-the-art algorithms. Often, these frameworks lack flexibility and scalability, covering only a limited set of scenarios. To fill this gap, in this paper we propose *DASHbed*, a highly customizable real-time framework for testing HAS algorithms in a wireless environment. Due to its low memory requirement, *DASHbed* offers a means of running large-scale experiments with a hundred competing players. Finally, we supplement the proposed framework with a dataset consisting of results for five HAS algorithms tested in various evaluated scenarios. The dataset showcases the abilities of *DASHbed* and presents the adaptation metrics per segment in the generated content (such as switches, buffer-level, *P.1203.1* values, delivery rate, stall duration, etc.), which can be used as a baseline when researchers compare the output of their proposed algorithm against the state-of-the-art algorithms.

CCS CONCEPTS

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

KEYWORDS

HTTP Adaptive Streaming, HAS, Testbed Framework, Dynamic Adaptive Streaming over HTTP, DASH, Real-Time Streaming

ACM Reference Format:
Darijo Raca, Yusuf Sani, Cormac J. Sreenan, Jason J. Quinlan. 2019. DASHbed: a testbed Framework for Large Scale Empirical Evaluation of Real-Time DASH in Wireless Scenarios. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Multimedia traffic dominates today's Internet, and this trend is expected to continue in the foreseeable future[1]. On average, users spend almost six hours a day consuming video content [2]. Furthermore, over-the-top (OTT) services experienced tremendous growth in recent years. Video-on-demand (VoD) services are the most dominant OTT service. Big players, like Netflix, Hulu, and Amazon offer subscription-based VoD services. The success of VoD technology lies, among other key factors (powerful compression methods and broadband access), in the ability to adapt streaming content to the current device (e.g., switch to lower video quality if the available link throughput plunges or does not stream content with a resolution not supported by screen capability). To enable this customisation, OTT providers employ HTTP adaptive streaming concepts at the heart of their services (in particular dominated by Apple HLS and MPEG-DASH streaming formats[3]). As multimedia traffic soars, so does user's expectation regarding video quality. The majority of users want reliable, high-quality video. Also, the most annoying factor causing streaming session abandonment is rebuffering/stalls [4]. Multimedia consumption over mobile devices is continuously rising every year [5], however, streaming high-quality and interrupt-free video content over a wireless link is a formidable challenge for both ISPs and content providers. While content can be compressed to reduce the delivery rate (H.264 to H.265), the inherent variability in the throughput of the air-interface of the wireless environment has the most impact on achievable quality at the device. As a result, most new HAS algorithms are typically judged by their performance in a wireless environment, such as over WiFi, 3G and 4G technologies.

The goal of many HAS algorithms over the last decade has been to minimise or completely abolish stalls, thus maximising achievable video quality based on the variance in the underlying network throughput. [1, 2]. These algorithms operate by estimating network conditions and adapting video quality based on available network resources.

Comparing different HAS algorithms is a non-trivial task, due in part to the variance in the underlying goals of the different HAS algorithms. Some algorithms aim for smooth streaming (minimise switching between different bitrates), others high quality, and no stalls (which are self-opposing) while others aim at improving network utilisation. Ultimately, the main aim of all HAS algorithms

[1] https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf
[2] https://www.nielsen.com/us/en/insights/reports/2018/q2-2018-total-audience-report.html
[3] https://go.bitmovin.com/download-the-bitmovin-2018-video-developer-report
[4] http://connect.mux.com/2017-streaming-perceptions-report
[5] https://www.conviva.com/research/convivas-state-streaming-tv-industry/

is to achieve the highest quality of user experience (QoE) irrespective of whether the adaptation is client focused, network focused or distributed between both. Because of its subjective nature QoE is typically difficult to measure and generalise. To help alleviate this problem, researchers have proposed various QoE models over the years, mapping various streaming video QoE-related metrics (e.g., average quality, switching, stalls) to a QoE score, typically in the range (0-5). While most of these models only take into consideration a subset of the mentioned QoE-related metrics to find an "optimal" blend between them, some efforts have been made in proposing subjectively-validated QoE models [3, 4]. The foregoing discussion notwithstanding, the first standardised model for QoE was published in 2017 [5].

Currently, to evaluate a proposed scheme against the state-of-the-art streaming algorithms, researchers need to build a custom made framework/testbed. Most of the time, reproducibility of their work is limited, as many researchers do not publish the source code. However, even when the code is released, the set of scenarios is typically limited, and any new test scenarios require a significant amount of effort to implement in their existing testbed.

With this in mind, in this paper, we reduce the evaluation workload by proposing *DASHbed*, a large-scale framework for testing HAS algorithms in a wireless environment. DASHbed allows automation of different scenarios under different conditions for various HAS algorithms. The framework is trace-driven, running over different wireless traces collected from real wireless networks e.g., WiFi, 3G, 4G). Furthermore, low memory requirements allow for running up to 100 competing video players on a single machine. DASHbed provides a rich set of flexibility in defining different scenarios making it appealing for researchers to use. Furthermore, DASHbed outputs result in a structured form, making it easy to analyse and compare. Finally, we include a *P.1203.1* model (mode 0) for video streaming as a standardised QoE model in our framework.

The following outlines the remainder of the paper. Section 2, presents information on related DASH datasets and QoE-related metrics and is followed by Section 3, where we provide an overview and introduce key features of our testbed framework. Section 4 provides an overview of the dataset generated by our framework and the relevant QoE-related metrics within. Section 5 concludes the paper.

2 BACKGROUND AND RELATED WORK

There are five crucial facets when analysing the performance of HAS algorithms. These are: *video content for streaming* (sufficiently large dataset of mixed content - resolutions, encoders and genres), *wireless bandwidth traces* (collected over different air-interfaces with various sample granularity), a range of *HAS algorithms* to compare against (with different demands and objectives), a subjective *QoE model* (standardised if possible) and a rich set of different scenarios for evaluation.

In the video research community, there is a relatively small number of DASH enabled datasets. Over time, the content of these datasets has evolved in resolution, encoders and bitrates. Lederer et al. [6], in 2012, released the first publicly available DASH dataset encoded with the H.264 encoder. Dataset consists of 6 Full HD clips encoded in 20 quality bitrates ranging from 50 *Kbps* to 8 *Mbps*. Also content was encoded in five different segment duration, 1, 2, 4, 6 10, and 15 seconds. The same authors later released the Distributed DASH dataset [7], which extended the selection of bitrate to encompass a diverse range of geographical content servers. The datasets of Quinlan et al. [8] provide DASH content with H.265 in addition to H.264 encoded video content. In total 23 clips were encoded with ten quality bitrates, five segment duration (2, 4, 6, 8, and 10 seconds). Similarly, Zabrovskiy et. al [9] released content encoded using multiple encoders (H.264, H.265, VP9, and AV1). This dataset consists of 4K content with quality bitrates going up to 20 *Mbps*. 4K datasets with Ultra High Definition resolutions (3840x2160) were generated [10, 11] with 40 *Mbps* maximum quality bitrate.

Similarly, there is a limited set of bandwidth traces collected over a wireless channel. The main differences in the traces are the granularity of samples and the channel and context metrics. Bokani et al. [12] collected 3G and 4G traces with 10-second resolution (traces are repeated over the same route to get statistically relevant results on network performance). Similarly, Xiao et al. and Li et al. [13, 14] collected bandwidth traces over 3G and 4G in high mobility scenarios (train and car). On the other hand, Riiser et al. [15] collected 3G traces for different mobility patterns (tram, train, metro, bus, ferry, and car). Similarly, Hooft et al. [16] used the same approach for 4G networks. All these traces contain a sample granularity in the order of seconds and provides additional information such as timestamp and GPS coordinates of the device. Meixner et al. [17] collected a large set of 5-minutes bandwidth traces with 100ms granularity collected over a 4G network and various mobility patterns. Furthermore, traces have additional channel information such as signal strength. Futhermore, Raca et al. [18] collected video-streaming based traces over two operational 4G networks with additional channel information such as velocity, SNR, CQI, RSRP and RSRQ.

The underlying concept of HAS is to split video content into multiple segments with segment duration ranging between 2-20 seconds. Each segment is encoded into multiple video bitrates. After the current segment is downloaded, an algorithm decides on the quality of the next requested segment. Broadly, these algorithms can be grouped into three categories: *rate-based* [19], which makes decisions based on delivery rates of previously downloaded segments; *buffer-based* [20], which monitors the state of playback buffer and makes appropriate decisions; *hybrid-based* [21] algorithms that take both approaches when deciding on the quality of the next segment.

HAS algorithms use different approaches in designing their adaptation logic, such as control theory [22], optimisation [23, 24], machine learning [25], game theory [26] and other techniques [21]. Other approaches leverage the possibility of having accurate throughput prediction and how it can help in improving HAS performance by incorporating prediction into existing algorithms [27] or designing new prediction-aware HAS algorithms [28, 29].

Many efforts have been made to analyse the performance of HAS players under different conditions [30–32]. Several key factors have been identified as a cause for low performance, such as "ON-OFF" behaviour of HAS player [33] and TCP Slow start [34]. Furthermore, these effects get exacerbated when multiple clients compete for the available resources. While many HAS algorithms exist in the literature, there is a limited number of evaluation frameworks available, as authors don't release their frameworks or frameworks are limited in their functionality. To fill this gap, we provide a DASH

testbed (*DASHbed*) framework for the evaluation of HAS algorithms in real-time.

Table 1: Ladder for the average UHD encoding rate, resolution and frame rate for the used dataset [35]

#	Bitrate	Resolution	Frame Rate
13	40Mbps	3840x1744	24, 60
12	25Mbps	3840x1744	24, 60
11	15Mbps	3840x1744	24, 60
10	4.3Mbps	1920x872	24, 60
9	3.85Mbps	1920x872	24, 60
8	3Mbps	1280x582	24, 60
7	2.35Mbps	1280x582	24, 60
6	1.75Mbps	720x328	24, 60
5	1.05Mbps	640x292	24, 60
4	750Kbps	512x234	24, 60
3	560Kbps	512x234	24, 60
2	375Kbps	384x174	24, 60
1	235Kbps	320x146	24, 60

3 THE DASHBED FRAMEWORK

Figure 1 illustrates the architecture of our DASHbed framework. Four main components are constituting our framework:

- Stored video content.
- Link throughput limiter.
- Wireless traces for link throughput emulation.
- HAS video player

In the following section, we give details for each component of our framework.

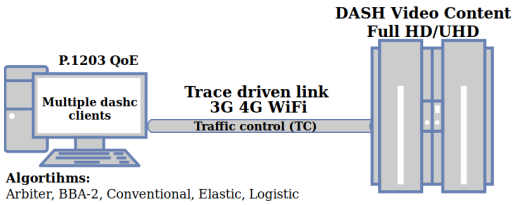

Figure 1: DASHbed framework Architecture

3.1 Main components of the framework

For video content, we use the Ultra High Definition (UHD/4K) dataset [35]. The dataset provides both AVC (H.264), and HEVC (H.265) encoded video content. However, we only generate *P.1203.1* values for H.264 encoded content. Furthermore, our framework only produces QoE values based on mode 0 of *P.1203.1*. This limits the QoE output of our model as the inclusion of bitstream parsing for Mode 1 and 2 is missing. Finally, integrating complete P.1203 model with initial delay and stall events (already produced by dashc player) is set for future work.

In [35] three well-known open-source video clips (Big Buck Bunny, Sintel, and Tears of Steel) are used for generating 4K DASH compatible content. Table 1 shows encoding settings used for the dataset (frame rates depend on video clip). The duration of the video clips ranges between 10 and 14 minutes. For segmentation and encoding dataset following tools were used: FFmpeg - encoding original content to lossless YUV format; x264/x265 codec for encoding to AVC/HEVC format; MP4Box for segmentation and multi-profile MPD generation. Furthermore, the video content is generated for five different segment durations (2, 4, 6, 8 and 10 seconds) allowing for a greater variety of experiments. This helps a researcher to better quantify the impact of different segment durations on the underlying adaptive delivery service.

For modelling link throughput, we use the Linux Traffic Control (tc) tool to modify the link capacity between video server and video clients. Depending on the bandwidth traces, link throughput is changed on the order of seconds (4G and WiFi every second, while 3G every couple of seconds). For bandwidth traces we use 4G [18] and 3G [15] bandwidth traces, while we generated our own WiFi traces. The 3G/4G traces are collected over various mobility patterns (static, pedestrian, car, tram, train). Modelling link capacity from real traces implicitly reflect impact of cross-traffic on available throughput. Using real traces implicitly reflect the impact of cross-traffic on available throughput. However, as cross-traffic characteristics are unknown, further qualification of impact, either on video performance by cross-traffic or vice versa, is not possible. On the other hand, full network emulation allows for further analysis of the interaction between video and different types of traffic. The current framework can be easily extended to support full network emulation through mininet [6] and tools for synthetic traffic generation (e.g., distributed internet traffic generator [36]). This is left as future work.

For the video player, we select dashc [37]. Dashc emulates a HAS video player by adopting all of the player's logic, except the option to decode the video content. This feature results in low memory requirements allowing for the running of a large number of concurrent players at the same time.

Dashc implements the following adaptation algorithms:

- *Conventional* - Conventional represents a rate-based adaptation algorithm. It makes its choice on the next segment, by using an exponential moving average of past segments delivery rate.
- *Elastic* [22] - Elastic uses a harmonic average of the recent five segment rates. The harmonic average is a conservative estimate of available throughput. Furthermore Elastic employs control-theory to combine throughput estimate and buffer levels when making video rate selection decisions.
- *Arbiter* [38] - Arbiter belongs to the hybrid category of HAS adaptive streaming algorithms. It employs exponential weighted moving average of the delivery rate of the last ten segment's to estimate the available throughput for the next segment. However, to adapt quickly to sudden changes in throughput (especially in a wireless environment), Arbiter employs adaptive scaling, such that its estimate is based on the second moment of the throughput sample. The additional scaling factor reflects the buffer state of the player (i.e., full buffer signals good conditions forcing the player to be less conservative by requesting higher video quality).

[6] http://mininet.org/

- *BBA-2* [20] and *Logistic* [39] - both algorithms select the next segment quality by mapping the buffer level to a target segment quality. Furthermore, BBA-2 incorporates a throughput-based decision in its "startup" phase while taking into account future video segment sizes. Logistic uses a logistic model to map buffer levels to video bitrate.

3.2 Key features of the framework

This section outlines the main characteristics of the DASHbed framework. Further details are available on the following webpage [7] [8]. DASHbed is designed to run completely automatically. To run experiments, one needs to execute a python3 control script and pass the required input arguments. These arguments consist of a physical interface (the interface linked to the machine containing dashc), number of clients, number of runs, duration of the segment, list of adaptation algorithms, duration of the segment, and list of movies. After running a control script, the script proceeds in the following order:

(1) Perform necessary checks
(2) Parse input arguments
(3) Create a config file for each client (adaptation algorithm and movie URL)
(4) Load trace files from trace folder
(5) For each trace, start bandwidth emulation using tc tool
(6) Start all video players
(7) Repeat steps 4-6 for defined number of runs

After the video player finishes streaming, a DASHbed log file is saved. Table 2 shows an example of the output from DASHbed. The logs contain values for each segment selection decision. Within this log file, we provide a rich set of various metrics, which can be used for later analysis. In addition to the standard performance metrics (quality rate, buffer level, and stall duration) which are commonly used to generate QoE related metrics, such as the average video quality, switching rate (i.e., instability) and video stalls (i.e., total stall duration and number of stalls). We provide detailed information for each segment (its delivery rate, actual rate, which is a function of segment size and segment duration, encoding format, segment resolution, and round trip time).

We combine the output of the generated DASHbed log files and create a dataset of content adaptation for the five algorithms currently implemented in the testbed (further details in Section 4). Finally, for each segment, we output a QoE score. For the QoE score, we use the recently standardised ITU-T Rec. *P.1203.1* HAS QoE model [5].

The P.1203.1 QoE model is derived from subjective studies. It uses MOS (mean opinion score, 5-point scale) to capture user experience of the streaming session, taking into account both video and audio impairments. The main drawback of the standardised model is its limitation to H.264 *(AVC)* encoding content to a maximum resolution of Full HD (1920x1080).

Many of the state-of-the-art streaming algorithms require information about segment sizes in advance. Availability of this information depends on the type of manifest profile, such as full, main, live, onDemand, byte-range, etc. Alternatively, a player needs to request all the per-segment transmission costs for all the segments, of all the available representation rates. DASHbed employs two approaches to get this information. First, by default, if the algorithm requires segment size information, the player will obtain information for each segment at the beginning by sending *head* requests for each segment across all quality levels. However, in experiments with limited bandwidth, this approach can take a long time (order of minutes in some cases) resulting in unusable results for comparison with other adaptation algorithms. To overcome this limitation, DASHbed also provides an alternative approach where information of the segment sizes are downloaded, once, before the start of an experiment, and are saved to a local file. For future requests, the segment size information is extracted from the local file.

Code 1 illustrates a template for running the video streaming experiments. This code is executed at the content machine:

Listing 1: Template for running DASHbed framework

```
# python3 runExperimentsDASHCMD.py interface \
numOfCClients numRuns streamingDuration \
[list of algorithms] segmentDuration [movieNames]
```

where arguments used are following: *interface* - physical interface for tc (bandwidth modulation) at the content machine; *numOfCClients* - number of competing clients to run on the client machine; *numRuns* - number of repeating run (to get statistically significant results) for each trace; *streamingDuration* - duration of experiment for each trace; *[list of algorithms]* - list of streaming algorithms used by competing clients (list must equal the number of clients). No spaces in list are permitted; *segmentDuration* - duration of the segment; *[movieNames]* - list of movies for each client (list must equal the number of clients).

The flexibility of the framework allows for running heterogeneous scenarios with different types of streaming algorithms competing for bandwidth resources.

4 OVERVIEW OF DATASET

To complement the proposed DASHbed framework, we produce a dataset using our framework covering a large variety of scenarios. As stated, we gather the output of the generated testbed log files and create a DASH dataset of content adaptation for the five algorithms currently implemented in DASHbed [9].

While this dataset can be used for further analysis and comparison with other adaptive algorithms, its primary goal is to showcase the potential and flexibility of DASHbed. The list of covered scenarios is not exhaustive by any means, but we feel it does illustrate the capabilities beyond current practice. The current version of dashc, available at the following website [10]. In addition, a Vagrant configuration file for creating the complete testbed across multiple VMs is available and contains the build scripts, dependencies and links to content to stream using dashc and produce the output logs as mentioned above.

Table 3 depicts all scenarios we have run with DASHbed. For bandwidth traces, we randomly select thirty 4G traces [18], and twenty five 3G traces [15]. Additionally, we collect five traces over

[7] http://www.cs.ucc.ie/misl/research/datasets/dashbed
[8] https://github.com/uccmisl/DASHbed
[9] http://cs1dev.ucc.ie/misl/dashbed/dashbed-dataset
[10] http://cs1dev.ucc.ie/misl/dashbed/dashc-updated-algorithms.zip

Table 2: Sample trace output from modified dashc

Seg_#	Arr_time	Del_Time	Stall_Dur	Rep_Level	Del_Rate	Act_Rate	Byte_Size	Buff_Level	RTT	Codec	Width	Height	FPS	Seg_Dur	Start	*P.1203.1*
1	109	109	0.000000	232	9070	248	124131	4.000	0.005969	h264	320	240	24	4.000	0.000	1.936882
2	1375	59	0.000000	232	18704	276	138452	8.000	0.006278	h264	320	240	24	4.000	4.000	1.936882
3	3116	533	0.000000	4275	39881	5323	2661696	11.466	0.034981	h264	1920	1080	24	4.000	8.000	2.669001
4	4621	268	0.000000	4275	47542	3187	1593595	15.198	0.075885	h264	1920	1080	24	4.000	12.000	2.997820
5	6012	113	0.000000	4275	53917	1524	762041	19.085	0.016670	h264	1920	1080	24	4.000	16.000	3.331018

Table 3: 60 Evaluated Scenarios

# of clients	# of algorithms	# of runs	segment dur. (s)
1	1	5	2, 4, 8
4	1	5	2, 4, 8
5	5	5	2, 4, 8
10	5	5	2, 4, 8

WiFi. Selected traces are included with the framework. All experiments are five minutes long (we only stream first five minutes of clips and not for whole duration). As the content is streamed in real-time, in total our evaluation took 62.5 days to run.

We start with a range of standard experiments, i.e., running a single algorithm against the wireless traces. This experiment provides for a comparison between the performance of each algorithm. Furthermore, all algorithms are evaluated for different segment durations, allowing for quantifying the impact of different segment duration on overall algorithms performance (e.g., longer segments are more "costly" as a wrong decision could result in a higher adverse effect on QoE compared to shorter segments). Next, the experiments are repeated in the same conditions with four clients, each with the same algorithm. These experiments can be used to analyse the fairness of each algorithm. While we do not undertake any evaluation of the output generated in our datasets, a commonly used fairness metric, Jain fairness index [40] or the recent F-index [41], a QoE fairness metric, could be utilised.

Finally, we evaluate five heterogeneous clients competing for throughput resources, with each client using a different adaptation algorithm. These experiments permit analysis on how algorithms are actually "fair" against other algorithms. This scenario is the most realistic one. In practice, one would expect multiple users are streaming from different providers at the same time. Table 4 illustrates the range for average bitrate, stall duration and P.1203.1 score for all tested algorithms across all scenarios.

Table 4: Range of average values for selected QoE-related metrics and P.1203.1 model for all scenarios (8-sec segment duration)

Algorithm	Bitrate (*Kbps*)	Stall dur. (s)	P.1203.1 (mode 0)
Arbiter	[517, 1421]	[1.47, 4.71]	[2.27, 2.85]
BBA	[238, 588]	[2.56, 3.45]	[2.02, 2.39]
Conv	[1235, 2427]	[1.97, 5.63]	[2.51, 3.18]
Elastic	[532, 1777]	[0.82, 5.53]	[2.1, 3.01]
Logistic	[662, 1943]	[1.86, 3.68]	[2.23, 3.11]

While these metrics showcase the dataset results, other metrics are necessary (e.g., fairness metric, instability) for full comparison.

5 CONCLUSION

In this paper, we release a flexible, scalable, and customizable real-time framework for testing HAS algorithms in a wireless environment, which we call DASHbed. Due to its low memory requirement DASHbed offers a means of running large-scale experiments with a hundred competing players, across a number of competing HAS algorithms. In the framework, we include the inline generation of QoE scores using the recently released standardised ITU-T Rec. P.1203.1 (mode 0) HAS QoE model. The value provided by this Empirical Evaluation is two-fold: 1) it provides a framework to validate the design and functionality of future HAS algorithms and 2) it offers an insight into how the design of existing HAS algorithms benefit from specific context, which can in some instances fail when generalised.

Finally, we supplement the framework with a dataset of generated content for the implemented algorithms across a range of scenarios, thus providing a means of benchmarking and validating future algorithms. Future work will encompass additional state-of-the-art HAS algorithms to dashc, importing all modes of P.1203 QoE model [42] and support for additional codecs (VP9, HEVC) [11]. Furthermore DASHbed will be extended with Google ExoPlayer, thus adding an Android-based mobile device to the range of players offered by DASHbed.

ACKNOWLEDGEMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 13/IA/1892.

REFERENCES

[1] Yusuf Sani, Andreas Mauthe, and Christopher F. Edwards. Adaptive bitrate selection: A survey. *IEEE Communications Surveys & Tutorials*, 19:2985–3014, 2017.

[2] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann. A survey on bitrate adaptation schemes for streaming media over http. *IEEE Communications Surveys Tutorials*, pages 1–1, 2018. ISSN 1553-877X. doi: 10.1109/COMST.2018.2862938.

[3] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao. Deriving and validating user experience model for dash video streaming. *IEEE Transactions on Broadcasting*, 61(4):651–665, Dec 2015.

[4] Stefano Petrangeli, Jeroen Famaey, Maxim Claeys, Steven Latré, and Filip De Turck. Qoe-driven rate adaptation heuristic for fair adaptive video streaming. *ACM Trans. Multimedia Comput. Commun. Appl.*, October 2015.

[5] Alexander Raake, Marie-Neige Garcia, Werner Robitza, Peter List, Steve Göring, and Bernhard Feiten. A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1. In *Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, Erfurt, May 2017. IEEE. ISBN 978-1-5386-4024-1. doi: 10.1109/QoMEX.2017.7965631. URL http://ieeexplore.ieee.org/document/7965631/.

[6] Stefan Lederer, Christopher Müller, and Christian Timmerer. Dynamic Adaptive Streaming over HTTP Dataset. In *Proceedings of the 3rd Multimedia Systems*

[11] https://github.com/Telecommunication-Telemedia-Assessment/itu-p1203-codecextension

Conference, MMSys '12, pages 89–94, New York, 2012. ISBN 978-1-4503-1131-1. doi: 10.1145/2155555.2155570. URL http://doi.acm.org/10.1145/2155555.2155570.

[7] Stefan Lederer, Christopher Mueller, Christian Timmerer, Cyril Concolato, Jean Le Feuvre, and Karel Fliegel. Distributed dash dataset. In *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys '13, pages 131–135, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1894-5. doi: 10.1145/2483977.2483994. URL http://doi.acm.org/10.1145/2483977.2483994.

[8] Jason J. Quinlan, Ahmed H. Zahran, Cormac J. Sreenan. Datasets for AVC (H.264) and HEVC (H.265) Evaluation of Dynamic Adaptive Streaming over HTTP (DASH). In *MMSys '16 Proceedings of the 7th ACM Multimedia Systems Conference*, May 2016.

[9] Anatoliy Zabrovskiy, Christian Feldmann, and Christian Timmerer. Multi-codec dash dataset. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 438–443, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5192-8. doi: 10.1145/3204949.3208140. URL http://doi.acm.org/10.1145/3204949.3208140.

[10] J. Le Feuvre, J-M. Thiesse, M. Parmentier, M. Raulet, and C. Daguet. Ultra High Definition HEVC DASH Data Set. In *Proceedings of the 5th ACM Multimedia Systems Conference*, MMSys '14, 2014. ISBN 978-1-4503-2705-3. doi: 10.1145/2557642.2563672. URL http://doi.acm.org/10.1145/2557642.2563672.

[11] Jason J. Quinlan and Cormac J. Sreenan. Multi-profile ultra high definition (uhd) avc and hevc 4k dash datasets. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 375–380, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5192-8. doi: 10.1145/3204949.3208130. URL http://doi.acm.org/10.1145/3204949.3208130.

[12] Ayub Bokani, Mahbub Hassan, Salil S. Kanhere, Jun Yao, and Garson Zhong. Comprehensive mobile bandwidth traces from vehicular networks. In *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16, pages 44:1–44:6. ACM, 2016. ISBN 978-1-4503-4297-1. URL http://doi.acm.org/10.1145/2910017.2910618.

[13] Q. Xiao, K. Xu, D. Wang, L. Li, and Y. Zhong. Tcp performance over mobile networks in high-speed mobility scenarios. In *2014 IEEE 22nd International Conference on Network Protocols*, pages 281–286, Oct 2014. doi: 10.1109/ICNP.2014.49.

[14] L. Li, K. Xu, D. Wang, C. Peng, Q. Xiao, and R. Mijumbi. A measurement study on tcp behaviors in hspa+ networks on high-speed rails. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2731–2739, April 2015. doi: 10.1109/INFOCOM.2015.7218665.

[15] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen. Commute path bandwidth traces from 3g networks: Analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys '13, pages 114–118, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1894-5. doi: 10.1145/2483977.2483991.

[16] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters*, 20(11):2177–2180, 2016.

[17] Britta Meixner, Jan Willem Kleinrouweler, and Pablo Cesar. 4g/lte channel quality reference signal trace data set. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 387–392, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5192-8. doi: 10.1145/3204949.3208132. URL http://doi.acm.org/10.1145/3204949.3208132.

[18] Darijo Raca, Jason J. Quinlan, Ahmed H. Zahran, and Cormac J. Sreenan. Beyond throughput: A 4g lte dataset with channel and context metrics. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 460–465, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5192-8. doi: 10.1145/3204949.3208123. URL http://doi.acm.org/10.1145/3204949.3208123.

[19] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. *IEEE/ACM Trans. Netw.*, 22(1):326–340, February 2014. ISSN 1063-6692. doi: 10.1109/TNET.2013.2291681. URL https://doi.org/10.1109/TNET.2013.2291681.

[20] T. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 187–198, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2836-4. doi: 10.1145/2619239.2626296.

[21] A. H. Zahran, D. Raca, and C. Sreenan. Arbiter+: Adaptive rate-based intelligent http streaming algorithm for mobile networks. *IEEE Transactions on Mobile Computing*, pages 1–1, 2018. ISSN 1536-1233. doi: 10.1109/TMC.2018.2825384.

[22] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo. Elastic: A client-side controller for dynamic adaptive streaming over http (dash). In *2013 20th International Packet Video Workshop*, pages 1–8. IEEE, Dec 2013. doi: 10.1109/PV.2013.6691442.

[23] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. *SIGCOMM Comput. Commun. Rev.*, 45(4), August 2015. ISSN 0146-4833.

[24] Ahmed H. Zahran, Jason Quinlan, Darijo Raca, Cormac J. Sreenan, Emir Halepovic, Rakesh K. Sinha, Rittwik Jana, and Vijay Gopalakrishnan. Oscar: An optimized stall-cautious adaptive bitrate streaming algorithm for mobile networks. In *MoVid*. ACM, 2016.

[25] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 197–210, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4653-5. doi: 10.1145/3098822.3098843. URL http://doi.acm.org/10.1145/3098822.3098843.

[26] Abdelhak Bentaleb, Ali C. Begen, Saad Harous, and Roger Zimmermann. Want to play dash?: A game theoretic approach for adaptive streaming over http. In *MMSys*. ACM, 2018.

[27] Darijo Raca, Ahmed H. Zahran, Cormac J. Sreenan, Rakesh K. Sinha, Emir Halepovic, Rittwik Jana, Vijay Gopalakrishnan, Balagangadhar Bathula, and Matteo Varvello. Incorporating prediction into adaptive streaming algorithms: A qoe perspective. In *NOSSDAV*. ACM, 2018.

[28] Xuan Kelvin Zou, Jeffrey Erman, Vijay Gopalakrishnan, Emir Halepovic, Rittwik Jana, Xin Jin, Jennifer Rexford, and Rakesh K. Sinha. Can accurate predictions improve video streaming in cellular networks? In *HotMobile*. ACM, 2015.

[29] Tarun Mangla, Nawanol Theera-Ampornpunt, Mostafa Ammar, Ellen Zegura, and Saurabh Bagchi. Video through a crystal ball: Effect of bandwidth prediction quality on adaptive streaming in mobile environments. In *MoVid*. ACM, 2016.

[30] Saamer Akhshabi, Ali C. Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, MMSys '11, pages 157–168, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0518-1. doi: 10.1145/1943552.1943574. URL http://doi.acm.org/10.1145/1943552.1943574.

[31] Saamer Akhshabi, Sethumadhavan Narayanaswamy, Ali C. Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptive video players over http. *Image Commun.*, 27(4):271–287, April 2012. ISSN 0923-5965. doi: 10.1016/j.image.2011.10.003. URL http://dx.doi.org/10.1016/j.image.2011.10.003.

[32] Theodoros Karagkioules, Cyril Concolato, Dimitrios Tsilimantos, and Stefan 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*, NOSSDAV'17, pages 1–6, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5003-7. doi: 10.1145/3083165.3083170. URL http://doi.acm.org/10.1145/3083165.3083170.

[33] Saamer Akhshabi, Lakshmi Anantakrishnan, Ali C. Begen, and Constantine Dovrolis. What happens when http adaptive streaming players compete for bandwidth? In *Proceedings of the 22Nd International Workshop on Network and Operating System Support for Digital Audio and Video*, NOSSDAV '12, pages 9–14, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1430-5. doi: 10.1145/2229087.2229092. URL http://doi.acm.org/10.1145/2229087.2229092.

[34] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard. In *Proceedings of the 2012 Internet Measurement Conference*, IMC '12, pages 225–238, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1705-4. doi: 10.1145/2398776.2398800. URL http://doi.acm.org/10.1145/2398776.2398800.

[35] Jason J. Quinlan and Cormac J. Sreenan. Multi-profile ultra high definition (uhd) avc and hevc 4k dash datasets. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 375–380, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5192-8. doi: 10.1145/3204949.3208130. URL http://doi.acm.org/10.1145/3204949.3208130.

[36] Alessio Botta, Alberto Dainotti, and Antonio Pescapè. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531–3547, 2012.

[37] Aleksandr Reviakin, Ahmed H. Zahran, and Cormac J. Sreenan. Dashc: A highly scalable client emulator for dash video. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 409–414, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5192-8. doi: 10.1145/3204949.3208135. URL http://doi.acm.org/10.1145/3204949.3208135.

[38] A. H. Zahran and C. J. Sreenan. Arbiter: Adaptive rate-based intelligent http streaming algorithm. In *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–6, July 2016. doi: 10.1109/ICMEW.2016.7574709.

[39] Y. Sani, A. Mauthe, and C. Edwards. Modelling video rate evolution in adaptive bitrate selection. In *2015 IEEE International Symposium on Multimedia (ISM)*, pages 89–94, Dec 2015. doi: 10.1109/ISM.2015.65.

[40] R. Jain, D. Chiu, and W. Hawe. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. *eprint arXiv:cs/9809099*, September 1998.

[41] T. Hoßfeld, L. Skorin-Kapov, P. E. Heegaard, and M. Varela. Definition of qoe fairness in shared systems. *IEEE Communications Letters*, 21(1):184–187, Jan 2017. ISSN 1089-7798. doi: 10.1109/LCOMM.2016.2616342.

[42] Werner Robitza, Steve Göring, Alexander Raake, David Lindegren, Gunnar Heikkilä, Jörgen Gustafsson, Peter List, Bernhard Feiten, Ulf Wüstenhagen, Marie-Neige Garcia, et al. Http adaptive streaming qoe estimation with itu-t rec. p. 1203: open databases and software. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 466–471. ACM, 2018.