AdViSE: Adaptive Video Streaming Evaluation Framework for the Automated Testing of Media Players

Anatoliy Zabrovskiy, Evgeny Kuzmin, Evgeny Petrov
Petrozavodsk State University
Lenina, 33
Petrozavodsk, Russia 185000
{z_anatoliy,kuzmin,johnp}@petrsu.ru

Christian Timmerer
Alpen-Adria-Universität Klagenfurt / Bitmovin Inc.
Universitätsstraße 65-67
9020 Klagenfurt, Austria
christian.timmerer@itec.aau.at

Christopher Mueller
Bitmovin Inc.
301 Howard Street, Suite 1800
San Francisco, California 94105
christopher.mueller@bitmovin.com

ABSTRACT

Today we can observe a plethora of adaptive video streaming services and media players which support interoperable formats like DASH and HLS. Most of the players and their rate adaptation algorithms work as a black box. We have developed a system for easy and rapid testing of media players under various network scenarios. In this paper, we introduce **AdViSE**, the **Ad**aptive **Vi**deo **S**treaming **E**valuation framework for the automated testing of adaptive media players. The presented framework is used for the comparison and testing of media players in the context of adaptive video streaming over HTTP in web/HTML5 environments.

The demonstration showcases a series of experiments with different media players under given context conditions (e.g., network shaping, delivery format). We will also demonstrate the real-time capabilities of the framework and offline analysis including several QoE metrics with respect to a newly introduced bandwidth index.

CCS CONCEPTS

•**Networks** →**Network performance modeling; Network experimentation;** Network reliability; •**Information systems** →**Multimedia streaming;**

KEYWORDS

Evaluation framework; AdViSE; Adaptive streaming; Media players; MPEG-DASH; Network emulation; Automated Testing; Mininet; Selenium; Quality of Experience; Metrics

ACM Reference format:
Anatoliy Zabrovskiy, Evgeny Kuzmin, Evgeny Petrov, Christian Timmerer, and Christopher Mueller. 2017. AdViSE: Adaptive Video Streaming Evaluation Framework for the Automated Testing of Media Players. In *Proceedings of MMSys'17, Taipei, Taiwan, June 20-23, 2017,* 4 pages.
DOI: http://dx.doi.org/10.1145/3083187.3083221

MMSys'17, Taipei, Taiwan
 978-1-4503-5002-0/17/06...$150.00
DOI: http://dx.doi.org/10.1145/3083187.3083221

1 INTRODUCTION

Adaptive video streaming over HTTP is becoming more and more the primary technology for video delivery in the open internet. For example, Netflix and YouTube alone account for more than 50% of the traffic [12] thanks to open, interoperable formats such as MPEG-DASH [14] or HLS [11]. In the past, we have witnessed many deployments in web environments implemented using Javascript by utilizing HTML5 and media source extensions (MSE) enabling a plugin free video streaming.

The technology behind adaptive video streaming over HTTP implies media representation switching (e.g., bitrate/resolution) depending on context conditions such as network characteristics and client device properties. An integral part of each player implementation is the usage of an appropriate rate adaptation algorithm which aims to follow dynamically changing context conditions. Thus, we are in a situation with many unknown variables and the problem of choosing the appropriate media player for video streaming and playback arises.

In this demo paper, we introduce ***AdViSE***, an **Ad**aptive **Vi**deo **S**treaming **E**valuation framework which enables the automated testing of media players — and, thus, rate adaptation algorithms — under various context conditions (e.g., client devices/platforms, network characteristics/conditions). The main focus of AdViSE is the provisioning of tools for easy and rapid experimentation with media players and algorithms as they appear on the market including updates thereof which become available in relatively short periods.

The rest of this paper is organized as follows. Section 2 describes the underlying scientific problems leading to the development of AdViSE. In Section 3, we present the architecture of the designed system and its components. Section 4 provides a brief description about the demo itself and Section 5 concludes the paper.

2 UNDERLYING SCIENTIFIC PROBLEM

Nowadays, there exists a large number of different players/algorithms (including commercially available ones) and most of them have been implemented in JavaScript. However, no common performance evaluation framework/system

for adaptive media players exists enabling easy/rapid integration of new players and algorithms as they become available (including updates thereof). Researchers and developers find themselves in situations where there is no public, reliable information about the performance of adaptive media players, specifically the rate adaptation algorithms of commercially available players. Mostly related is probably TAPAS [6], which focuses on algorithms and written in Python. Whereas our focus is on media players written in JavaScript for web/HTML5 environments.

According to a recent survey [13] there are many different QoE models available where each of them proposes their own metrics. In many cases, these metrics are not considered to be used in the context of real-world streaming solutions and deployments. The goal of AdViSE is the integration of existing or even new QoE metrics in our framework which allows us to get results about the QoE almost in real-time and also for offline analysis.

In terms of context conditions, adaptive media players have to cope with a heterogeneous environment and having even one change in the context condition might have a big impact on the player behavior. The context conditions can be categorized into several areas:

- Access network: wired, WiFi, mobile 3G/4G/5G;
- Network architecture/paradigm: content deliver network (CDN), software-defined networking (SDN), information-centric networking (ICN);
- Client device: desktop, laptop, mobile, TV, set-top box, virtual reality (VR)/head-mounted display (HMD);
- Client device condition/state: in motion, fixed;
- Server infrastructure: single server, direct stream from Internet of Multimedia Things (IoMT) device; and
- Content characteristics: live or video on-demand, segment size (e.g., 2s, 4s, 6s or 9s), number of representations (bitrates, resolutions, languages), video codec, content profile.

Moreover, new versions of players are released in a short period (i.e., from two weeks to several months) and also new algorithms become available. Unfortunately, all players have their own application programming interface (API) and sometimes they do not work well (e.g., provide erroneous data or no data at all). Therefore, the most critical challenge for an effective adaptive media player comparison is to have a common, ideally standardized API.

In practice, however, all clients expect to observe low video startup time, smooth and high quality video playback without interruptions (stalls/rebuffering events). To better understand the efficiency and the major features of popular media players and their rate adaptation algorithms, the AdViSE framework has been designed. It provides powerful tools to both researchers and practitioners which enables easy and rapid testing of new media player solutions and rate adaptation algorithms under a variety of context conditions.

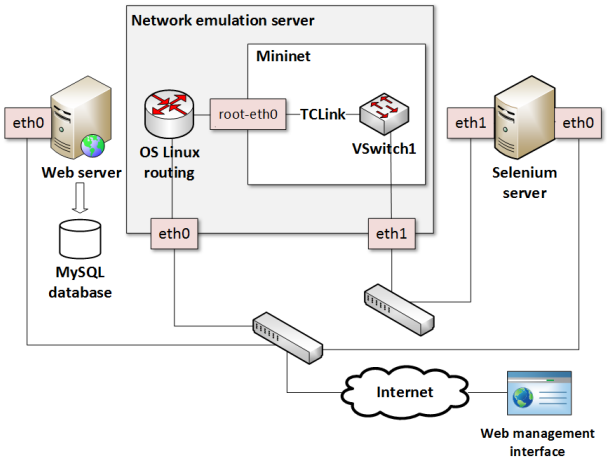

Figure 1: System Architecture.

Table 1: Overview of Adaptive HTML5 Players

Media player	Version	Web site (last access: Feb 27, 2017)
Bitmovin Player	7.0	https://bitmovin.com
dash.js	2.4.0	http://dashif.org
Flow Player	6.0.5	https://flowplayer.org
HAS Player	1.7	https://github.com/Orange-OpenSource/hasplayer.js
JW Player	7.6.1	https://www.jwplayer.com
Radiant MP	3.10.8	https://www.radiantmediaplayer.com
Shaka Player	2.0.3	https://github.com/google/shaka-player
THEOPlayer	2.8.1	https://www.theoplayer.com
VideoJS Player	5.9.2	http://videojs.com

3 DEMO SYSTEM ARCHITECTURE

Our proposed system includes the following components:

- Web server with standard HTTP hosting the segmented video content and a MySQL database.
- Network emulation server with customized Mininet [3] environment for bandwidth shaping.
- Selenium [4] server for running adaptive media players on various platforms.
- Web management interface for (*i*) conducting the experiments and (*ii*) running the adaptive media players.

The demo system architecture is shown in Figure 1 and the details for each component are described in the following subsections. It defines a flexible system that allows adding new adaptive media players relatively fast. Table 1 comprises a list of players (in alphabetic order) which already have been integrated in our system including version and web site.

Figure 2: Picture of the Server Infrastructure.

The server infrastructure comprises three servers with distinct functionalities running Ubuntu OS (version 16.04 LTS) and connected using Gigabit Ethernet switches. The *Web server* hosts the video content for the adaptive streaming and also hosts a MySQL database for collecting all the performance measurements. The *Selenium server* provides means for the automation of adaptive media players and requests video content through the *Network emulation server* based on Mininet. Finally, the *Web management interface* is hosted on the *Web server* and allows to configure and conduct the actual experiments. It is accessible from outside the controlled environment, everything else is within a controlled environment in order to avoid any cross-traffic that may influence the experiments. A picture of the real system is shown in Figure 2.

3.1 Web Server

The Web server hosts the video content for adaptive streaming over HTTP and currently AdViSE supports MPEG-DASH [14] and HLS [11] as delivery formats.

In addition to the video content, the web server also hosts a MySQL database for collecting all the performance measurements and the web management interface for configuring and conducting the experiments.

3.2 Selenium Server

The Selenium server is an open source software testing framework for web applications [4] which is used to automatically conduct our experiments with different adaptive media players running within a web browser. In our case we adopted the Google Chrome browser but it is also possible to use other browsers on various platforms (desktop, mobile, different operating systems). The Selenium server is activated through the web management interface to run the various experiments automatically according to a given configuration.

3.3 Network Emulation Server

For the network emulation server we have adopted the Mininet emulator [3]. Although this emulator is basically used for emulating software-defined network (SDN) environments, it has been also used for adaptive streaming environments [15]. It provides a straightforward and extensible Python API for network creation and prototyping. We have utilized that functionality to create a virtual link with changeable network throughput characteristics.

Our network emulation server comprises two network interfaces (eth0, eth1) and we have installed Mininet version 2.3.0d1. A python script is used to create a virtual network which consists of one switch connected to the real network using a TCLink [5]. By creating a virtual node in a root name space, connecting it to the virtual switch and assigning an IP address from a separate network we connect our virtual environment with the real network. The network emulation server has the role of the router between the real and the virtual networks (cf. Figure 1).

A real network interface eth1 of the server is configured as a port of that virtual switch. This network interface and the eth1 network interface of the Selenium server are connected to the same switch and, thus, they are in the same broadcast domain. We assign an IP address of our virtual network to the eth1 network interface of the Selenium server and, consequently, the traffic to that network interface goes trough the TCLink.

With the help of the Minievents framework [2] we setup our TCLink to change its characteristics at the specified moments of time. The schedule is stored within a file using the JSON format. To make this schedule configurable through our web management interface, a special PHP script was prepared and is accessible within the network. This script retrieves the schedule of TCLink parameters through the HTTP POST request in JSON format, prepares the configuration file, and runs our python script. If the test is already started, it replies with error and ignores the request.

3.4 Web Management Interface

The web management interface provides two functions, (*i*) one for configuring and conducting the experiments and (*ii*) one for presenting the player with real-time information about the currently conducted experiment. The former allows to define the following items and parameters:

- management of network emulation profiles including the configuration of the bandwidth trajectory, packet loss, and packet delay;
- specification of the number of runs of an experiment; and
- selection of the adaptive media player and the utilized adaptive streaming protocol (MPEG-DASH or HLS).

The result page provides a list of conducted experiments. The analytics section provides various metrics of the conducted experiments and it is possible to generate graphs of

the results by using Highcharts [1] and exporting the raw values for further offline analysis.

Before starting the experiment we need to create a bandwidth trajectory profile. For each profile we can define duration of each stage, bandwidth, delay, and packet loss. As soon as we start an experiment within the web management interface, the Google Chrome browser is automatically launched on the Selenium server and the selected network profile including the link parameters is sent to network emulation server. The actual requests for the video content towards the web server goes through the network emulation server.

When running an experiment it is possible to display the currently selected adaptive media player (including the video streaming) and real-time information about the currently conducted experiment and its parameters.

4 DEMONSTRATION

During our demonstration we are going to run a number of experiments on our testbed. Remote access from the demo presentation room to the system will be organized by the TeamViewer remote access tool or VNC Viewer. Throughout all experiments the quality parameters are recorded in a MySQL database. It is possible to create a new bandwidth trajectory scheme or use a predefined one which is used by the network emulation server to adjust the bandwidth, e.g.: 750 kbps, 350 kbps, 2500 kbps, 500 kbps, 700 kbps, 1500 kbps, 2500 kbps, 3500 kbps, 2000 kbps, 1000kbps and 500 kbps. Such a scheme of bandwidth trajectory inevitably causes quality switches of DASH/HLS streams.

In our experiments we use the Big Buck Bunny video sequence which is also commonly used in similar testbeds within this experimentation domain [8]. Other sequences can be added easily. In the demo the content is encoded and prepared according to DASH/HLS utilizing two different profiles. The first comprises a *FullHD* profile with five different representations: 426x238 pixels (400kbps), 640x360 (800), 854x480 (1200), 1280x720 (2400), and 1920x1080 (4800). For the second configuration we reverse-engineered the *Amazon* Prime video service which offers 15 different representations: 400x224 (100), 400x224 (150), 512x288 (200), 512x288 (300), 512x288 (500), 640x360 (800), 704x396 (1200), 704x396 (1800), 720x40 (2400), 720x40 (2500), 960x540 (2995), 1280x720 (3000), 1280x720 (4500), 1920x1080 (8000), and 1920x1080 (15000). In both cases we adopted a segment length of four seconds as it provides the best trade-off with respect to streaming performance and coding efficiency [8] which is also used in commercial deployments like Netflix.

Demonstration attendees will be able to see the media player launched on a Selenium server in a Google Chrome browser and to observe real-time statistics about the currently conducted experiment. After the completion of the experiment the results will be presented in the result web interface of the system in tables and charts. The following quality parameters and metrics are available:

- download video bitrate (or selected video quality);
- video buffer length (or video buffer level);
- video startup time;
- stalls (or buffer underruns);
- number of quality switches;
- instability and inefficiency [7];
- average video bitrate;
- Bandwidth index combining instability, inefficiency, and average video bitrate;
- QoE metrics: $QoE_{Mäki}$ [9], which takes into account the number of stalls, total stalling time; QoE_{Mok} [10], which depends on video start-up time, stalling frequency and average duration of a stalling event.

5 CONCLUSIONS

AdViSE allows us to conduct many experiments in an automatic mode and to make a deep analysis of the obtained results using different video streaming parameters and QoS/QoE metrics/models. Our future work will include adding new players and QoE metrics, investigating how different adaptive media players compete for the available bandwidth in a shared network.

REFERENCES

[1] Highcharts. http://www.highcharts.com/, (Online: accessed January 24, 2017).

[2] Minievents framework. https://github.com/cgiraldo/minievents, (Online: accessed January 24, 2017).

[3] Mininet. http://mininet.org/, (Online: accessed January 24, 2017).

[4] Selenium server. http://www.seleniumhq.org/, (Online: accessed January 24, 2017).

[5] TCLink. http://mininet.org/api/classmininet_1_1link_1_1TCLink.html, (Online: accessed January 24, 2017).

[6] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo. TAPAS: A Tool for rApid Prototyping of Adaptive Streaming Algorithms. In *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming*, VideoNext '14, pages 1–6, New York, NY, USA, 2014. ACM.

[7] J. Jiang, V. Sekar, and H. Zhang. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive. *IEEE/ACM Trans. Netw.*, 22(1):326–340, Feb. 2014.

[8] S. Lederer, C. Müller, and C. Timmerer. Dynamic Adaptive Streaming over HTTP Dataset. In *Proceedings of the 3rd Multimedia Systems Conference*, MMSys '12, pages 89–94, New York, NY, USA, 2012. ACM.

[9] T. Mäki, M. Varela, and D. Ammar. A Layered Model for Quality Estimation of HTTP Video from QoS Measurements. In *2015 11th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, pages 591–598, Nov 2015.

[10] R. K. P. Mok, E. W. W. Chan, and R. K. C. Chang. Measuring the Quality of Experience of HTTP Video Streaming. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 485–492, May 2011.

[11] R. Pantos and W. May. HTTP Live Streaming. Internet-Draft draft-pantos-http-live-streaming-20, Internet Engineering Task Force, Sept. 2016. Work in Progress.

[12] Sandvine. 2016 Global Internet Phenomena Report: Latin America & North America, 2016. Online: *http://sandvine.com/*.

[13] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys Tutorials*, 17(1):469–492, Firstquarter 2015.

[14] I. Sodagar. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE MultiMedia*, 18(4):62–67, 2011.

[15] A. Zabrovskiy, E. Kuzmin, E. Petrov, and M. Fomichev. Emulation of Dynamic Adaptive Streaming over HTTP with Mininet. In *2016 18th Conference of Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT)*, pages 391–396, April 2016.