A Prototype of Mininet-Based System EmStream for Emulation of Dynamic Adaptive Streaming over HTTP

Anatoliy Zabrovskiy, Evgeny Petrov, Evgeny Kuzmin, Mikhail Fomichev, Natalya Sokolova
Petrozavodsk State University
Petrozavodsk, Russia
{z_anatoliy, johnp, kuzmin, fomichevm, nsokolova}@petrsu.ru

***Abstract*—With the proliferation of media systems for content delivery, there is a growing demand for sophisticated methods to research emerging technologies in this field. For instance, new solutions for streaming video can be studied using real communication networks which is not always convenient or even feasible. In this paper, we present EmStream a practical solution to investigate the delivery of media content over the Internet using a MPEG-DASH technology. In its core, EmStream combines network emulation environment Mininet with a real hardware client and a server from the IP-network. We describe the adopted approach and demonstrate a prototype of our Mininet-based system for the emulation of MPEG-DASH stream transmission over a pre-set network topology. Specifically, we will show how to embed the virtual Mininet environment into a real network infrastructure and utilize the aforementioned setting to conduct experiments. Based on our findings, we conclude that EmStream built upon Mininet is a practical tool for testing DASH-based media streaming. Moreover, our Mininet-based system is considered to be reasonably accurate for emulating DASH-content delivery in pre-set topologies as well as software-defined networks.**

I. Introduction

Dynamic Adaptive Streaming over HTTP (DASH), also known as MPEG-DASH, is the first adaptive bit rate solution based on HTTP which became an international standard in 2012 [1]. MPEG-DASH was specifically designed to deliver data streams to a user with the highest possible bit rate under the varying bandwidth conditions. It can also be used for the on-demand live streaming and time-shifted delivery [2]. The advantages of MPEG-DASH as compared to other proprietary streaming solutions were shown in [2]. Currently, the DASH standard is being widely deployed [3], especially in live streaming video systems which means that the format will play an important role in this field. It is viable that shortly MPEG-DASH will be more actively used together with such technologies as Software-defined Networking (SDN), Content Delivery Network (CDN) and Content-Centric Networking (CCN) [22], [23], [18].

In order to investigate new technologies for streaming video existing communication networks can be utilized which is not always convenient or even feasible. Thus, to overcome the aforementioned obstacle network emulators are frequently used one of which is open-source Mininet [4]. Mininet is capable of building realistic virtual topologies consisting of numerous network elements such as end hosts, switches, routers and communication links. In a nutshell, Mininet implements a concept of Software-Defined Networking (SDN). SDN is a novel networking paradigm in which a control layer is implemented in software and separated from a packet forwarding plane [5]. It is expected that the future deployment of a new SDN concept [6] will allow administrators to control network services and hardware without having to know the intricate underlying functionality.

As we expect, the rapid development of new approaches to network establishment and maintenance together with technologies for media content delivery will eventually lead to their complementary utilization. Enabling such functionality would maximize the perceived quality of experience (QoE) while watching video. With this in mind, we decided to estimate the delivery efficacy of the real MPEG-DASH traffic through Mininet.

To achieve the stated goal we developed a methodology for setting the Mininet virtual environment with bandwidth shaping functionality. Additionally, we built an experimental setup that interconnects two parts: a virtual environment established with Mininet and a real IP-network. In order to evaluate the relevance of the proposed solution we compared the process of transmitting DASH content via the Mininet-based system and the specialized Linktropy 5500 equipment [7] under a number of traffic shaping scenarios. Based on these experiments we showed that both settings yield similar results by applying methods of statistical data analysis [24]. Therefore, our solution becomes a flexible and affordable platform for researching and developing systems for adaptive streaming. Inspired by this observation we extended the Mininet-based system with a web management interface and a set of new features. The newly presented EmStream prototype consists of the following parts:

- Customized Mininet environment.
- Web management interface.
- MySQL database.

Overall, we consider the application of Mininet to emulate the delivery of media content with adaptive streaming as a direction which has considerable practical potential. The rest of the paper describes the building blocks of EmStream, its underlying functionality as well as the use case scenarios and feasible extra features.

ISSN 2305-7254

II. MININET OVERVIEW

In its core, network emulator Mininet uses a mechanism of Linux lightweight containers which allows configuring and running a virtual network within a single OS kernel [8]. The Mininet topology consists of various instances that can be set such as communication links, hosts, network switches and controllers. The host within Mininet behaves like a real computer. Therefore, it is possible to use SSH connections to remotely access a particular node and run real programs on it such as a network protocol analyzer (e.g. Wireshark [9]) or a web server. Additionally, Mininet provides a capability to configure and tune numerous parameters of a communication channel such as throughput, delay, jitter, etc.

To accommodate a large number of networking instances (e.g. hosts and switches) Mininet uses process-based virtualization within a single OS kernel [4]. On top of that Mininet supports user interaction (e.g. using *ping* or *traceroute* utility) with a virtual network environment by means of a command line interface (CLI). Furthermore, Mininet can be interfaced with real networks, for instance, wired or wireless local area networks (LANs). In order to start with Mininet a laptop or a desktop running a Linux operating system is required. Moreover, a straightforward and extensible Python API is provided for network establishment and development [10].

Being designed in such a way, Mininet follows the concept of Software-Defined Networking (SDN). The main difference between the SDN and a traditional network is the separation between control and forwarding planes which results in more flexible and agile network configurations. The control layer is usually represented by so-called network controllers such as POX, ONIX, VAN SDN Controller [11], [12], [13] and others which are responsible for centralized control (e.g. a decision whether to drop, forward or buffer a particular packet) and network monitoring. The forwarding plane consists of various networking devices such as a switch, router and access point. Both layers communicate through some interface protocol such as OpenFlow [14].

A vital feature of SDN is its capability to be adjusted according to specific user or application requirements after the network has been set. That gives network administrators and developers an opportunity to design and implement their own application functionality to work with SDN controllers. For instance, various applications of HP VAN SDN Controller can fulfil actual end-to-end service requirements for network performance, monitoring, quality of service and security [15].

Due to its rich emulation capabilities and scalability, the application domain of Mininet is much broader than just SDN. A good example of using Mininet functionality for researching new network technologies is a Mini-CCNx project [17]. Mini-CCNx is a tool for fast prototyping of Information Centric Networks (ICN) based on the CCN model [16]. It extends Mininet to support the emulation of CCNx hosts and routers by running the customized *ccnd* daemon [17]. The main idea of the CCN network paradigm is to label certain content and make it directly addressable and routable throughout the network. In this case endpoints use for addressing the marked data instead of IP addresses. It is worth emphasizing that the CCN concept has received great attention from the research community, also in the directions related to MPEG-DASH [18]. Currently, Mininet together with Mini-CCNx can be used for the rapid prototyping of CCN networks.

III. EMSTREAM DEMO

In this section we will describe our EmStream system which consists of three interconnected blocks: Mininet PC, Server PC and Client PC. To start with, the Mininet PC is represented by a computer supplied with two network interfaces (*Int_1* and *Int_2*) and running Mininet software. In addition, a web server (the Server PC) which contains MPEG-DASH content is connected to the Mininet PC via the *Int_1* interface. It should be noted that the media content stored on the server side was generated in advance using the Bitcodin service [19]. Moreover, a client computer (the Client PC) which receives the information from the server is connected to the second network interface of the Mininet PC (*Int_2*) which is configured as a virtual switch port in the Mininet topology. It is important to emphasize that the Client PC is assigned an

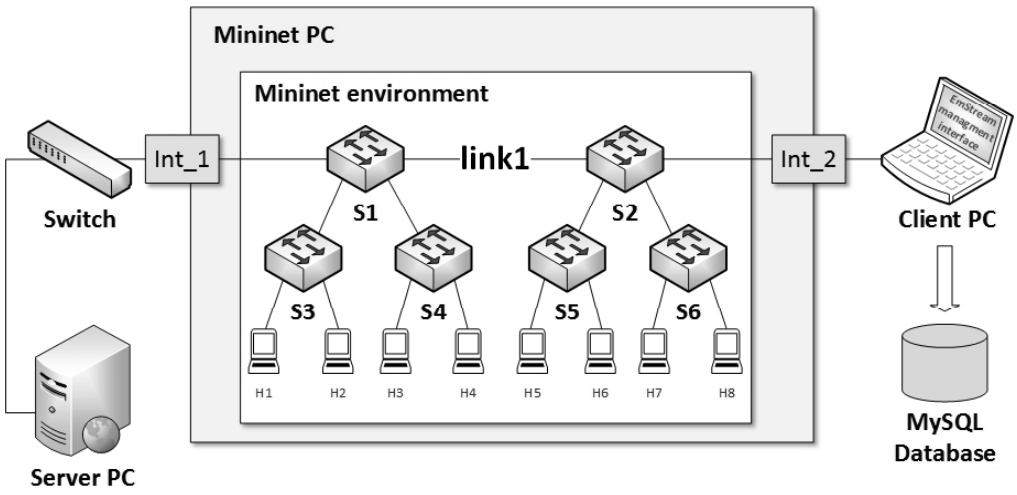

Fig. 1. EmStream demo setting

IP address from the range of Mininet network. Therefore, in the presented setting the client computer is considered as a part of the Mininet virtual IP-network. Whereas, the Server PC belongs to another (external) network which should be reachable from the Mininet network. The high-level overview of the described setup is depicted in Fig. 1.

The demonstrated EmStream setting works as follows. Initially, Python scripts are used to automatically create the specified virtual Mininet topology and interconnect it with a real-IP network. On to top of that we implemented functionality that allows varying channel characteristics such as bandwidth, packet loss and delay according to a predefined scenario. Particularly, the channel properties of any virtual link (e.g. *link1* in Fig. 1) can be tuned at certain moments in time. Thus, the video streams destined for the Client PC can be influenced and the corresponding effect observed and estimated.

On the client side we opted for the Bitmovin bitdash player [20] as a media player for the video streams playback. The player is embedded into the EmStream management interface and configured to send application playback parameters directly to a MySQL database. Among the parameters that we capture are the bit rate of the currently played video segment (*videoBitrate* in kbps) and the bit rate of the downloaded video segment. For instance, videoBitrate values are obtained through the player's API provided by the developer [21]. At the present time, EmStream supports only the Bitmovin bitdash player, although the integration of other playback solutions is also possible.

The EmStream management interface is illustrated in Fig. 2. As can be seen the interface is built in a user-friendly manner which enables to design an experiment in a convenient way and easily run it. Additionally, such functionality as monitoring playback parameters and displaying the results in real-time is also supported by the EmStream interface. Throughout the whole experiment the playback parameters are recorded each second and stored in the MySQL database. The current version of our prototype allows configuring the following parameters and options:

- the duration of an experiment and the overall number of experiments in succession;
- link selection: choosing a Mininet virtual link to which the specified parameters would apply;
- link characteristics:bandwidth, delay and packet loss;
- a bandwidth shaping scenario;
- settings of MySQL database connection;

With EmStream we designed several experiments which we are going to demonstrate. One of the examples can be described using the setting presented in Fig. 1. The whole experiment takes 120 seconds during which the bandwidth of a communication channel (i.e. *link1*) is varied according to a predefined scenario that we set with the EmStream management interface. Specifically, each 30 seconds the bandwidth changes according to the following sequence: 1 Mbps, 2 Mbps, 3 Mbps and 1 Mbps. Such a pattern of bandwidth shaping inevitably causes the bit rate switch of various DASH-based streams which we can observe with our system. More details regarding the experiments and capabilities of the EmStream system will be provided during the demo session.

IV. Conclusion and Future Work

Our results show that the EmStream system based on Mininet can be viably used for emulating content delivery using Adaptive Streaming over HTTP.

In our future research we are planning to incorporate complex network topologies within the Mininet environment. Hence, we could conduct more sophisticated experiments in the delivery of DASH-based content that would reflect large heterogeneous infrastructures typical for real-life scenarios. On top of that we consider integrating the network emulation profiles recommended by the implementation guidelines [25] into the future versions of EmStream. Yet another direction that we deem fruitful is the development of EmStream features which could simultaneously communicate with the Mininet environment and APIs provided by the vendors of media players. Systems with such functionality would be able to

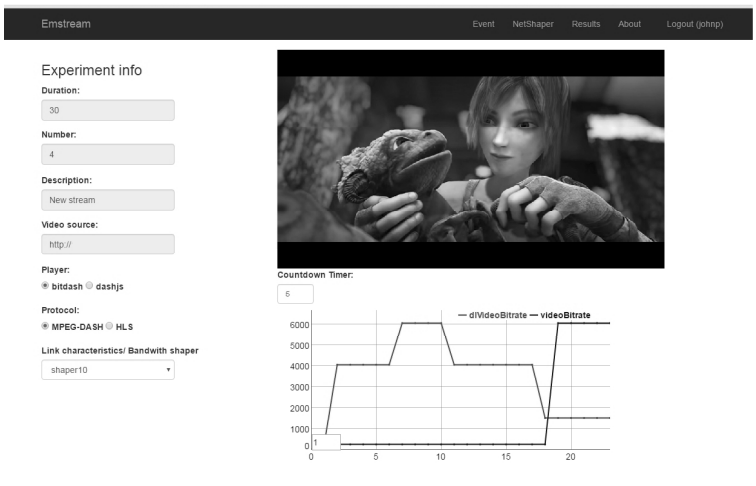

Fig. 2. EmStream management interface

automatically conduct experiments for the delivery of MPEG-DASH content, taking into account different network impairments and various test profiles. Thus, having such capability built-in opens new opportunities for testing adaptive control algorithms implemented in current media players as well as developing new algorithms.

Eventually, we aim for the mature EmStream system which would easily allow implementing new test profiles related to the delivery of MPEG-DASH content and incorporating different networking paradigms such as SDN, CDN and CCN.

V. Acknowledgments

This work was supported by the Strategic Development Program of Petrozavodsk State University (2012-2016).

References

[1] Dynamic Adaptive Streaming over HTTP,
Web: http://mpeg.chiariglione.org/standards/mpeg-dash

[2] Dynamic Adaptive Streaming over HTTP (DASH) - A Flexible and Efficient Solution for Mobile Multimedia,
Web: http://www.qualcomm.com/media/documents/files/dynamic-adaptive-streaming-over-http-dash-.pdf

[3] The State of MPEG-DASH Deployment,
Web: http://www.streamingmediaglobal.com/Articles/Editorial/Featured-Articles/The-State-of-MPEG-DASH-Deployment-96144.aspx

[4] Mininet Overview,
Web: http://mininet.org/overview

[5] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li: "Software defined networking: State of the art and research challenges", *Computer Networks*, vol.72, 2014, pp. 74–98.

[6] M. Liyanage, A. Gurtov, and M. Ylianttila. *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture*. Wiley, 2015.

[7] Linktropy 5500 Hardware Guide - Apposite Technologies,
Web: http://www.apposite-tech.com/assets/pdfs/linktropy-5500-hwguide.pdf

[8] B. Heller. *Reproducible network research with high-fidelity emulation*. PhD thesis, Stanford University, 2013.

[9] Wireshark,
Web: https://www.wireshark.org

[10] Mininet Python API Reference Manual,
Web: http://mininet.org/api/annotated.html

[11] POX Wiki,
Web: https://openflow.stanford.edu/display/ONL/POX+Wiki

[12] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inouye, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks", *In Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, 2010.

[13] HP VAN SDN Controller Software,
Web: http://h17007.www1.hpe.com/us/en/networking/products/network-management/HP_VAN_SDN_Controller_Software

[14] W. Braun and M. Menth. Software-defined networking using openflow: Protocols, applications and architectural design choices. *Future Internet*, 2014.

[15] Technical white paper. HP Virtual Application Networks SDN Controller,
Web: http://h17007.www1.hp.com/docs/networking/solutions/sdn/4AA4-8807ENW.PDF

[16] C. Cabral, C. E. Rothenberg, and M. F. Magalhães. Mini-CCNx: Fast prototyping for named data networking. *In Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, 2013.

[17] Mini-CCNx,
Web: https://github.com/chesteve/mn-ccnx/wiki

[18] Y. Liu, J. Geurts, JC. Point, S. Lederer, B. Rainer, C. Müller, C. Timmerer, and H. Hellwagner. Dynamic adaptive streaming over CCN: A caching and overhead analysis. In *2013 IEEE International Conference on Communications (ICC)*, 2013.

[19] Video Encoding Service for Adaptive Streaming,
Web: https://www.bitcodin.com

[20] Bitmovin bitdash player,
Web: http://www.dash-player.com

[21] JavaScript API Documentation,
Web: https://bitmovin.com/player-documentation/player-api/

[22] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race. Towards network-wide QoE fairness using openflow-assisted adaptive video streaming. *In Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*, 2013.

[23] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj. Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network. *Signal Processing: Image Communication*, 2012.

[24] A. Zabrovskiy, E. Kuzmin, E. Petrov, M. Fomichev. Emulation of Dynamic Adaptive Streaming over HTTP with Mininet. *Proceedings of the FRUCT18*, 2016.

[25] Guidelines for Implementation: DASH-AVC/264 Test cases and Vectors,
Web: http://dashif.org/wp-content/uploads/2015/04/DASH-AVC-264-Test-Vectors-v09-CommunityReview.pdf