QoE Estimation–based Server Benchmarking for virtual Video Delivery Platform

Lauri Koskimies, Tarik Taleb, and Miloud Bagaa
Dept. of Communications and Networking, School of Electrical Engineering
Aalto University, Espoo, Finland
Emails: firstname.lastname@aalto.fi

***Abstract*—This paper introduces a Quality of Experience (QoE) estimation–based server benchmarking system, which can be utilized as a part of QoE-optimized resource provisioning in our envisioned virtual video delivery platform. The system has been targeted for benchmarking virtual video streaming servers, i.e., virtual server flavors deployed in a cloud environment, based on resulting QoE estimates. The paper also presents another layer to the benchmarking by showing how to optimize stream segment duration in terms of estimated QoE. The QoE estimation in the system is based on a Pseudo-Subjective Quality Assessment (PSQA) method developed for video streaming. Output of the system, i.e., QoE estimation–based benchmarks, helps to find out how different factors can affect video streaming QoE which in turn makes parameter and resource optimizations possible. Moreover, the paper presents experimental benchmarking results obtained in a cloud environment.**

I. Introduction

In today's Internet, a clear majority of all network traffic is generated by video streaming applications, including both video on demand (VOD) and live video streaming scenarios. Global traffic share of video streaming was 70% of all consumer Internet traffic in 2015 and is forecast to grow even further, being 82% by 2020 [1]. A substantial portion of video streaming traffic originates from big streaming services, such as Netflix and Youtube, which offer mainly VOD content. Currently, in North America, during peak traffic periods, Netflix and Youtube represent together over 50% of all downstream traffic on fixed networks [2]. Furthermore, live video streaming services, e.g., Twitch; a video gaming broadcasting platform, have started to become more prominent factors in the traffic generation [3].

Video streaming is therefore becoming a strong competitor against the traditional ways to consume video content including television and optical storage mediums (e.g., DVD and Blu-ray) [4][5]. The growing popularity of video streaming is a result of improved end-user experience, due in turn to advancements in both networking and streaming technologies. Faster networks have enabled users to watch high-quality video streams with lower buffering delays. Several streaming protocols have been developed to enable streaming over Internet in general. As an example, the Real-time Transport Protocol (RTP) has been traditionally popular in Internet telephony applications. More recent development of novel protocols which enable streaming over HTTP, such as Apple's HTTP Live Streaming (HLS), has helped streaming services to become further popular, also in the World Wide Web (WWW) environment.

The ever-increasing demand for video streaming services, the emergence of cloud computing, and the importance of end user satisfaction act as motivators of this work to design and implement a QoE estimation–based server benchmarking system. The system can be used to study how different parameters of interest affect video stream QoE estimates, which in turn enables QoE estimation–based parameter optimizations. In order for the system to be elastic, a QoE estimation method that works without human intervention is needed. For this purpose, we have adopted the method proposed in [6].

The QoE of a video streaming service can be affected by various factors which can be roughly divided into three categories: resources allocated for the streaming infrastructure, the video stream parameters, and outside factors. The allocated amount of computing resources has an effect on how well the system handles high user loads. In addition to bitrate and video quality in general, stream-relevant parameters, such as duration of the stream segment, may have an effect on the perceived QoE. That is due to the fact that short video segments, i.e., chunks, cause streaming clients to request segments from the server more often compared to when the segment duration is set to high values. This may ultimately cause video playout interruptions if the server is not able to respond fast enough. The outside factors, affecting the QoE, include at least the bottleneck network bandwidth between the streaming server and the client, and the number of concurrent users viewing the stream. High number of concurrent viewers generally means high server load which, in turn, may cause QoE degradation.

To test real-life applicability of our proposed system, we will carry out experimental benchmarks which focus on finding out how the server load and the stream segment duration affect the QoE of a video stream. The effect of network bandwidth is omitted because it is not usually under control of the streaming service provider. The key results from the benchmarks are presented in this paper to provide general insights on how different parameters can affect the QoE of video streams.

The remainder of this paper is organized as follows. Section II presents fundamental background topics of this work and some related research work. The implementation of the QoE estimation–based benchmarking system is described in Section III. Section IV presents results of the experiments we carried out using the implemented benchmarking system. Section V briefly describes our future research work related to enhancements to both the benchmarking system and server load simulation. The paper is concluded in Section VI.

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

II. Background and related work

A. Video stream QoE estimation

The quality of a video, or alternatively the quality of a video stream, can be assessed using either objective or subjective methods. Traditional objective image quality models, including peak signal-to-noise ratio (PSNR) and structural similarity (SSIM), can be also used to assess video quality. When a video is transmitted through a network in real-time as in the case of video streaming, a well-known objective method to estimate video quality is to monitor network performance in terms of quality of service (QoS). Common QoS parameters include packet loss and jitter which can be used as sources for estimation. In contrast, subjective methods assess quality as experienced by humans and are therefore more related to the concept of QoE. In our opinion, subjective methods should be preferred in video quality assessment because objective measures usually do not correlate well enough with subjective quality ratings given by humans. Experiments presented in [7] support the previous opinion by showing that QoS parameters alone are not enough to assess or estimate QoE of a video. The experiments show that the Mean Opinion Score (MOS), a traditional subjective measure of audio or video quality, is also affected by other factors including age and gender. Performing subjective video quality assessment is, however, costly and slow to carry out. To speed up and automate the process, methods that try to estimate QoE of a video without human intervention have been proposed by several studies. Hereunder, we briefly describe two of them.

In [6], a QoE estimation scheme based on PSQA and Random Neural Networks (RNN) for adaptive HTTP/TCP video streaming is proposed. In the scheme, QoE estimation relies on the following two metrics: Quantization Parameter (QP) used in video compression and interruptions occurred during video playout. Higher QP value means higher information loss and playout interruptions may be caused by bandwidth fluctuations or too high server load. Also, while streaming over TCP does not naturally suffer from data loss due to TCP retransmissions, the retransmission process, itself, can cause playout interruptions due to increased delay in the video data delivery. That is why playout interruptions are a relevant factor when estimating QoE of HTTP/TCP stream. In the study of [6], the relation between the two metrics and subjective perceived quality has been captured by RNN experiments.

Another RNN–based video QoE estimation scheme is proposed in [8]. It focused on wireless networks and maps Medium Access Control (MAC) level parameters, such as bit error rate and queue length, to QoE. That is justified by wireless networks being naturally error prone and queue length having its pros and cons: smaller queue can increase packet drop whereas longer queue can result in outdated information. Mapping of the parameters and QoE is done by conducting subjective tests with real users and correlating the results with MAC level parameters. This data is then used to train the RNN for QoE estimation. To proof the validity of the method, the study presented an acceptable correlation between subjective QoE and estimated QoE.

B. Benchmarking of video streaming servers

Server benchmarking generally aims to find out how many concurrent users a server can handle with an acceptable service quality. That said, a realistic load simulation is an important part of a successful and relevant benchmark. However, benchmarking results depend on many factors specific to the underlying environment such as server, operating system and network configurations, in addition to the actual benchmarking variables, e.g. size of simulated load. In that sense, benchmarking results are best applicable only in exactly identical environments.

Benchmarking of video streaming servers requires generating realistic video workloads. In [9], methodologies for generating HTTP video workloads that try to accurately model the request traffic of servers, are presented. Authors in [10] utilized those methodologies for their video streaming benchmark, and they also took into account the popularity distribution of different videos and varying available bandwidth, i.e. video bit-rate, among simulated users. Their work indicated that request mix of the workload and server machine configuration, e.g., distribution of interrupt handling between CPU cores, can have a significant impact on benchmarking results.

C. Video stream segmentation

Video streaming requires the videos to be split into segments which are sent from server to clients according to a streaming protocol. Segment duration depends in particular on streaming implementation but may affect the video quality and encoding efficiency. Small segments lead to worse encoding efficiency, but yield better user experience due to more accurate seeking while playing the video. Small segments may also increase server load because clients have to request segments more often. On the other hand, large segments may cause playout interruptions because their downloading requires more time. Recommendations also exist regarding the segment duration. For example, Apple recommends 6 seconds as a target duration for HLS segments for the stream to be better compatible with their devices [11]. Below, we mention two studies related to stream segment duration and sizing.

Authors in [12] have studied how segment duration in HTTP adaptive streaming affects video bitrate, buffer level, end-to-end delay and network load. According to their results, long segments result in higher video bitrate due to better bandwidth utilization. Because of the same reason, long segments also reduce the network load. In contrast, short segments enhance the bitrate adaptation process and also reduce the likelihood of buffer underflows.

The effect of segment duration on performance of streaming proxy servers has been studied in [13]. The authors in that study proposed a scheme that adjusts segment duration dynamically according to segment popularity. More popular segments can have long segment durations. In this way, the

proxy server can prefetch bigger portions of multimedia files into memory and fulfill future requests more efficiently.

D. QoE-aware virtual resource allocation for video streaming

QoE-aware resource allocation and network management, especially management of Virtual Network Functions (VNF), have recently gained research attention [14][15][16][17]. Methods of similar mindset, i.e., QoE-aware decision making in resource management, would also be beneficial in the video streaming context. A large scale video streaming service usually requires a content distribution network (CDN) to be able to serve different geographical regions and variable number of concurrent users in each region. Therefore, it is important for CDN to be dynamic and able to adapt to current user load and performance requirements. In contrast, video streaming service providers want to concentrate on offering the video content itself rather than managing the CDN infrastructure. That has led to offering of CDN as a service (CDNaaS). For example, CDNaaS can be offered by a local network operator on top of their Telco infrastructure. CDN can contain different components including streaming servers, caches, load balancers and transcoding servers, in addition to the origin servers containing the original video content. The benchmarking system presented in this paper could be utilized as part of a QoE-aware system for the provisioning of virtual server resources in a CDNaaS architecture. Therefore, we briefly highlight two additional studies: one related to CDNaaS concept and another one focusing on QoE-aware VNF management in video streaming context.

In [18], a CDNaaS architecture for on-demand service deployment over a telco CDN is proposed, where content providers can express their requirements regarding performance, and can then lease CDN resources where the Internet Service Provider (ISP) has presence. Also, by performing certain testbed experiments, the authors evaluated the capabilities of two competing virtualization technologies, i.e., traditional full machine virtualization and containerization, both of which can be used to implement the proposed architecture. According to their results, traditional virtualization comes with a larger overhead, but on the other hand, containerization may not be mature enough as a technology yet.

Transcoding servers are an important part of large scale video streaming infrastructures, and can also be implemented as VNFs. Authors in [19] propose a scheme for QoE-aware elastic transcoding service creation targeted to Mobile Edge Computing (MEC). The scheme, where QoE assessment is based on PSQA methodology, aims to enforce the edge to maintain sufficient stream QoE by on-the-fly transcoding. Their testbed experiments predict a promising real-life applicability for the scheme.

III. QoE Estimation–based Benchmarking System

A. Architecture overview

The architectural components and operation logic of our proposed benchmarking system are depicted in Fig. 1 and elaborated below:

1) Initialization request: The QoE client requests a certain load from the load generator;
2) Load generation: The load generator starts the load towards the streaming server (i.e. running on a certain flavor of virtual resources);
3) Streaming request: The QoE client requests a video stream from the streaming server;
4) Streaming video: The streaming server starts streaming the requested video to the QoE client;
5) Computing QoE: The QoE client calculates the QoE estimates (in terms of MOS) of the video stream and sends periodical QoE reports to the orchestrator.

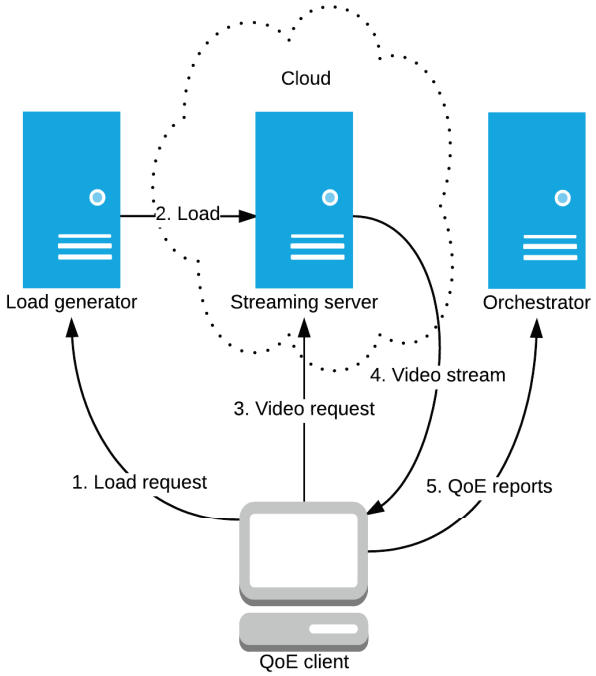

Fig. 1. Architecture and operation logic of the envisioned benchmarking system.

The QoE client is executed on a Ubuntu Linux installed in a VirtualBox Virtual Machine (VM). The implementation of the QoE client consists of a Python program, shell scripts, a modified VLC player and a MOS calculation tool. The load generator is deployed on a Debian machine powerful enough to generate load, and is implemented as a Python program which utilizes a wrk2 [20] load generation tool. The streaming server and the orchestrator are executed inside an OpenStack cloud environment as cloud instances. The orchestrator does not necessarily need to be deployed in the cloud, as Fig. 1 suggests. The streaming server is deployed as a HTTP server, whereas the orchestrator program is implemented with Python.

B. Streaming technology

As a streaming technology for the envisioned system, we chose HTTP-based HLS protocol because of its wide adoption in general, and also because it suits best the modified VLC player we used to gather necessary data for MOS calculations.

One of the reasons behind the wide adoption of HTTP-based streaming is that older streaming protocols, such as Real Time Streaming Protocol (RTSP), require the server to keep states of user sessions while HTTP allows stateless servers and is therefore better for horizontal scalability.

HLS streaming can be implemented by any server capable of serving HTTP requests. Therefore, we chose a widely adopted NGINX [21] HTTP server as our streaming server software. Nginx was configured to serve HLS specific file types including transport stream (.ts) files, i.e., streaming segments, and playlist (.m3u8) files containing the list of segments in the stream. In addition, NGINX was configured to spawn worker processes so that the number of processes equals the number of CPUs on the server, following NGINX recommendations.

C. Stream encoding and segmentation method

As a test video for our experiments, we chose a 5 minutes and 36 seconds long video which we prepared for HLS streaming using FFmpeg [22] software. Preparation included the following steps:

1) The video was re-encoded from 1080p to 360p resolution which reduced the bitrate to 873 Kbps. By this, we ensured that available bandwidth within our environment, measured with iPerf [23] tool, won't become a bottleneck under the highest load scenarios.
2) The video was segmented into five separate streams each with a certain constant segment duration, to be able to study the effect of segment duration on QoE.

While we defined the segment duration as constant in time units, i.e., seconds, segment size expressed in size on disk, i.e., bytes, still varies. However, this did not matter in our case because we wanted to study specifically the effect of segment duration on QoE as it has the effect on overall request rate generated by clients.

D. Load generation methods

In our system, a single QoE client plays the actual video stream and calculates the MOS values, while the rest of the load is created by the load generator. In our experiments, we tried two different methods to generate the load. Firstly, we generate loads with high HTTP request rates by requesting a very small file from the server repeatedly. Secondly, we generate more realistic streaming loads by simulating certain number of concurrent users, i.e. streaming clients. In the latter method, we create a certain number of concurrent HTTP connections to the streaming server and request a file whose size is equal to the average segment size, in bytes, of our test video stream. The number of connections to be created is determined by (1), and the request rate to be generated is determined by (2). Here, we make an assumption that a real streaming client would make a request for segment every n seconds, where n equals the segment duration.

$$number\_of\_connections = number\_of\_users \quad (1)$$

$$request\_rate = \frac{number\_of\_users}{segment\_duration} \quad (2)$$

E. QoE estimation and reporting method

To obtain QoE estimates of the video stream, the QoE client calculates MOS values ranging from one (1) to five (5) by utilizing a separate software tool which implements the PSQA-based technique presented in [6]. In the technique, as mentioned earlier, the factors affecting MOS are playout interruptions and Quantization Parameters of the video. To gather interruption and QP data, the QoE client uses a VLC player modified for this purpose. Interruption data include interruption start and end timestamps in addition to interruption durations. QP data is collected as timestamped samples of QP values. MOS is calculated for each 16 seconds of played video in addition to possible interruptions. This is defined as MOS calculation window in [6]. In case the stream cannot start for at least 30 seconds due to server overload, we assume average MOS to be minimum (1) in that particular load scenario.

MOS calculation and reporting are triggered after each played segment of the video. At that moment, the QoE client determines passed MOS calculation windows, calculates the MOS for each passed window, and reports the MOS values to the orchestrator. In addition to MOS value for each window, the QoE report includes current average, maximum and minimum MOS since the beginning of the video. To gather as much data as possible, we also included interruption statistics to the report. Statistics consist of interruption count and durations in addition to average, minimum and maximum duration. In this manuscript, the purpose of the orchestrator is limited to collecting the QoE reports. However, we envision the orchestrator to have a more central role in our future research work, as described in Section V.

IV. Performance evaluation

A. Impact of server load on QoE

In the first experiment, we used the first load generation method described in Section III-D. The objective was to find out how different request loads may affect video stream QoE. We run the benchmarks with five different load scenarios using the following request rates: 1200, 1250, 1300, 1350 and 1400 requests per second. While generating each request rate, the load generator was instructed to create 500 concurrent HTTP connections to the server to better simulate concurrency. The test video, using segment duration of 4 seconds, was streamed twice for each load scenario, and average MOS of the two streams was recorded. We run this experiment against the streaming server deployed on a cloud VM flavor with 1 virtual CPU (vCPU). Fig. 2 presents the results of this experiment.

The results show that average MOS quickly drops to minimum (1) when the load exceeds the server capacity. This indicates that streaming servers may perform well in terms of QoE even when the load is very close but just under the server capacity. By utilizing that information, a server maintainer may consider adding more computing power only when the load seems to be very close to capacity. It is worth noting that the server capacity in terms of request rate depends on the

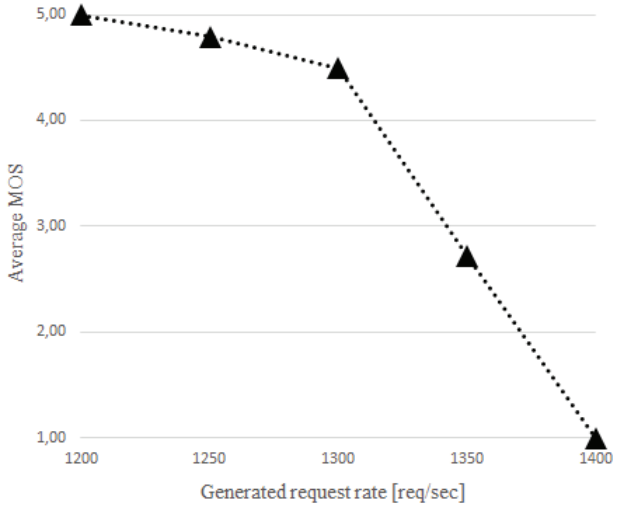

Fig. 2. Impact of server load on QoE.

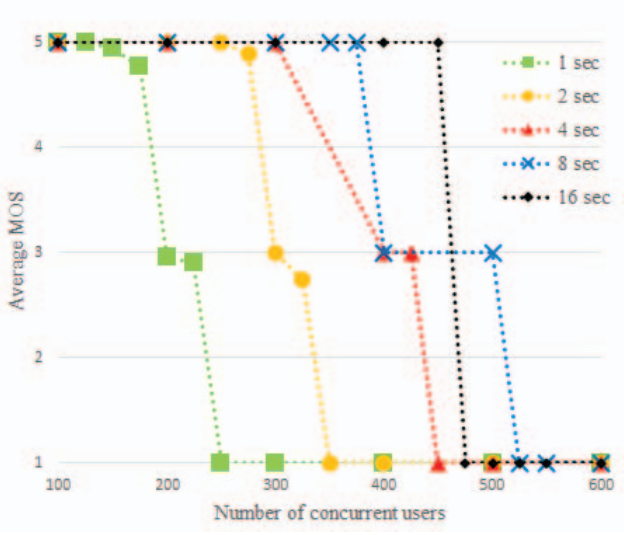

Fig. 3. Impact of server load and stream segment duration on QoE.

size of the requested file. In this test, we used a file size of 1000 bytes to be able to experiment with high request rates.

B. Impact of server load and stream segment duration on QoE

In the second experiment, we used the second load generation method described in Section III-D which also allowed us to find out how different stream segment durations may affect QoE. We run the benchmarks with load scenarios ranging from 100 to 600 concurrent users, and with five different segment durations, i.e., 1, 2, 4, 8 and 16 seconds. Benchmarks with load scenarios of 100, 200, 300, 400, 500 and 600 were run for each segment duration. In addition to that, we increased the load scenario density when average MOS values started to drop. We streamed the test video twice for each benchmark scenario, and recorded average MOS of the two streams. All the benchmark scenarios with corresponding average MOS values are presented in Fig. 3. Also in this experiment, we used a cloud VM flavor with 1 vCPU for the streaming server.

The presented results indicate that segment duration may have a remarkable effect on video stream QoE. In our experiment, the longest segment duration, i.e. 16 seconds, seemed to be the best choice in terms of resulting QoE. In that scenario, average MOS started to drop from its maximum value (5) after the load increased over 450 concurrent users. In the scenario of 1-second segments, the QoE degradation started already around 150 concurrent users. In all of the segment duration scenarios, average MOS dropped to its minimum value (1) relatively quickly after the load exceeded the server capacity. To sum up, these results can be seen as an example indicating how duration of stream segment can be optimized to increase the number of concurrent users a streaming server is likely to be able to serve, given a certain QoE requirement.

V. Future work: User-friendly benchmarking and enhanced load simulation

In order to make the benchmarking system more user-friendly, our future work will focus on exposing the benchmarking system through a web application which can be implemented in the orchestrator. In this scenario, the orchestrator would have a central role. It would handle benchmark request coming from a user by deploying a requested cloud flavor, forwarding load request to load generator, measuring QoE, and finally delivering QoE reports to the user. Designed architecture and operation logic of the system is presented in Fig. 4. The presented benchmarking system would be an important part of our envisioned virtual video delivery platform, components of which are outlined, e.g., in [14], [18], and [24].

Moreover, if utilized in conjunction with a virtual infrastructure management scheme proposed in [25], the benchmarking system would offer a user a possibility to take samples of resulting QoE of a video streamed from different sized flavors, i.e., with different number of virtual CPU cores, memory and storage.

Also, our future work includes enhancing the load generation method to simulate streaming clients in a more realistic way. We plan to create simulation workloads where all stream segments are requested sequentially one by one, instead of requesting a single file repeatedly. It would be interesting to find out if a more realistic user simulation loads the server differently compared to the methods we used in this work.

VI. Conclusion

In this paper, we presented a QoE estimation–based server benchmarking system targeted to be part of our envisioned virtual video delivery platform. In the system, video stream QoE is estimated in terms of MOS which is obtained automatically without any human intervention using a selected PSQA-based method. We showed that the system can be used to find out how different factors, such as server load, affect QoE of a video stream. Stream segment duration affects the load characteristics, and, according to the performed experiments, therefore also affects QoE. It is thus an important factor to be taken into account during resource optimizations. For example,

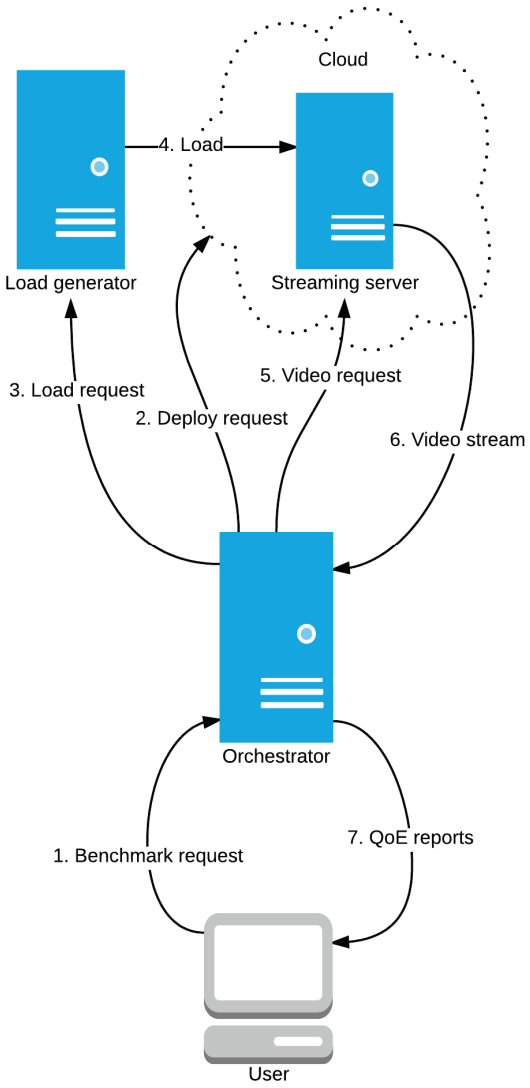

Fig. 4. Architecture and operation logic of the envisioned user-friendly benchmarking system.

adjusting segment duration based on estimated QoE instead of increasing flavor size can yield more efficient resource usage and cost savings.

Acknowledgement

This work was partially supported by the European Unions Horizon 2020 research and innovation programme under the 5G!Pagoda project with grant agreement No. 723172.

References

[1] Cisco Systems. (2016, Jun.) Cisco Visual Networking Index: Forecast and Methodology, 2015-2020. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf

[2] Sandvine. (2016, Jun.) Global Internet Phenomena Report. [Online]. Available: https://www.sandvine.com/downloads/general/global-internet-phenomena/2016/global-internet-phenomena-report-latin-america-and-north-america.pdf

[3] ——. (2015, Dec.) Global internet phenomena report. [Online]. Available: https://www.sandvine.com/downloads/general/global-internet-phenomena/2015/global-internet-phenomena-africa-middle-east-and-north-america.pdf

[4] Clearleap. (2015, Nov.) Study: Streaming Penetration 'On Par' With Cable. [Online]. Available: http://www.multichannel.com/news/content/study-streaming-penetration-par-cable/395332

[5] The Broadcast Bridge. (2015, Aug.) The Future of Optical Recording Media. [Online]. Available: https://www.thebroadcastbridge.com/content/entry/3377/the-future-of-optical-recording-media

[6] K. D. Singh, Y. Hadjadj-Aoul, and G. Rubino, "Quality of experience estimation for adaptive HTTP/TCP video streaming using H.264/AVC," *2012 IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 127–131, Jan. 2012.

[7] H. G. Msakni and H. Youssef, "Is QoE estimation based on QoS parameters sufficient for video quality assessment?" *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 538–544, Jul. 2013.

[8] I. Paudel, J. Pokhrel, B. Wehbi, A. Cavalli, and B. Jouaber, "Estimation of video QoE from MAC parameters in wireless network: A Random Neural Network approach," *2014 14th International Symposium on Communications and Information Technologies (ISCIT)*, pp. 51–55, Sep. 2014.

[9] J. Summers, T. Brecht, D. Eager, and B. Wong, "Methodologies for Generating HTTP Streaming Video Workloads to Evaluate Web Server Performance," *Proceedings of the 5th Annual International Systems and Storage Conference*, pp. 2:1–2:12, 2012.

[10] T. Palit, Y. Shen, and M. Ferdman, "Demystifying cloud benchmarking," *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 122–132, Apr. 2016.

[11] Apple Inc. (2016) HLS Authoring Specification for Apple Devices. [Online]. Available: https://developer.apple.com/library/content/documentation/General/ Reference/HLSAuthoringSpec/Requirements.html

[12] D. M. Nguyen, L. B. Tran, H. T. Le, N. P. Ngoc, and T. C. Thang, "An evaluation of segment duration effects in HTTP adaptive streaming over mobile networks," *2015 2nd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS)*, pp. 248–253, Sep. 2015.

[13] T. Yeh and Z. Yang, "Using dynamic segmentation adjustment to improve the performance of streaming proxy servers," *IEEE international Symposium on Broadband Multimedia Systems and Broadcasting*, pp. 1–5, Jun. 2012.

[14] S. Retal, M. Bagaa, T. Taleb, and H. Flinck, "Content Delivery Network Slicing: QoE and Cost Awareness," *Proc. IEEE ICC 2017*, May 2017.

[15] S. Dutta, T. Taleb, and A. Ksentini, "QoE-aware elasticity support in cloud-native 5G systems," *2016 IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2016.

[16] A. Ksentini, T. Taleb, and K. B. Letaif, "QoE-Based Flow Admission Control in Small Cell Networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2474–2483, Apr. 2016.

[17] T. Taleb and A. Ksentini, "QoS/QoE predictions-based admission control for femto communications," *2012 IEEE International Conference on Communications (ICC)*, pp. 5146–5150, Jun. 2012.

[18] P. A. Frangoudis, L. Yala, A. Ksentini, and T. Taleb, "An architecture for on-demand service deployment over a telco CDN," *2016 IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2016.

[19] S. Dutta, T. Taleb, P. A. Frangoudis, and A. Ksentini, "On-the-Fly QoE-Aware Transcoding in the Mobile Edge," *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Dec. 2016.

[20] wrk2. [Online]. Available: https://github.com/giltene/wrk2

[21] NGINX. [Online]. Available: https://www.nginx.com

[22] FFmpeg. [Online]. Available: https://ffmpeg.org

[23] iPerf. [Online]. Available: https://iperf.fr

[24] A. Nakao, P. Du, Y. Kiriha, F. Granelli, A. A. Gebremariam, T. Taleb, and M. Bagaa, "End-to-End Network Slicing for 5G Mobile Networks," *Journal of Information Processing*, vol. 25, no. 1, pp. 153–163, Jan. 2017.

[25] F. Z. Yousaf and T. Taleb, "Fine-Grained Resource-Aware Virtual Network Function Management for 5G Carrier Cloud," *IEEE Network Magazine*, vol. 30, no. 2, pp. 110–115, Mar. 2016.