# Study on the accuracy of QoE monitoring for HTTP adaptive video streaming using VNF

**Lam Dinh-Xuan, Michael Seufert, Florian Wamser, Phuoc Tran-Gia**

**Angaben zur Veröffentlichung / Publication details:**

Dinh-Xuan, Lam, Michael Seufert, Florian Wamser, and Phuoc Tran-Gia. 2017. "Study on the accuracy of QoE monitoring for HTTP adaptive video streaming using VNF." In *IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 8-12 May 2017, Lisbon, Portugal*, edited by Prosper Chemouil, Edmundo Monteiro, Marinos Charalambides, Edmundo Madeira, Paulo Simões, Stefano Secci, Luciano Paschoal Gaspary, and Carlos Raniery P. dos Santos, 999–1004. Piscataway, NJ: IEEE.
https://doi.org/10.23919/inm.2017.7987425.



# Study on the Accuracy of QoE Monitoring for HTTP Adaptive Video Streaming Using VNF

Lam Dinh-Xuan, Michael Seufert, Florian Wamser, Phuoc Tran-Gia
Chair of Communication Networks,
Institute of Computer Science, University of Würzburg
Am Hubland D-97074, Würzburg, Germany
Email: lam.dinh-xuan|seufert|florian.wamser|trangia@informatik.uni-wuerzburg.de

***Abstract*—The fast growth of video streaming offers a potential market for video providers, which can significantly increase their revenues. In order to provide users a good experience, HTTP adaptive video streaming has been introduced to adapt the video quality to the network conditions. Nevertheless, it is still difficult for the network operators to assess the actual video quality on the device of the users and therefore they may not react to improve the service on the network.**

**In this work, we propose a Virtual Network Function (VNF) to monitor the Quality of Experience (QoE) for online video service in the network. To conduct the study, on the one hand, we design a VNF monitoring to measure the video quality and estimate the QoE at the client machine. On the other hand, we place the function in two locations nearby and far away from the user to analyze the impact of geographical placement of the VNF on its performance. Our findings show that with respect to function placement, the VNF has high accuracy in estimating the QoE if it is deployed at the edge network close to the user. However, the VNF does not perform well when it operates far away from the users, e.g., at data centers. These insights help network vendors to more closely monitor the quality of the videos streamed to their customers.**

## I. Introduction

The variety of Internet video providers (e.g., YouTube, IPTV, Netflix) as well as user-generated video content are responsible for an impressive increase in video traffic in the data networks. In [1], Cisco predicts that nearly a million minutes of video content will cross the network in every second. The rapid growth of video streaming provides video providers with a great opportunity to increase their revenues, but also presents the challenges of managing the high volume of video traffic and the larger number of subscribers. As users expect a good service, they may stop watching the video if there are interruptions during playback. Consequently, to offer a high QoE perceived by the users, the network providers must be aware of how good the video is provided to the users. To tackle this problem, an in-network video monitoring mechanism is necessary. This not only provides the providers with information about the video quality that prevails on the user's device, but also gives them the ability to improve the service, for example by making decisions to migrate the content to the edge in order to reduce latency.

In order to monitor the video quality in the network, the video flows must be analyzed at packet level. This can be achieved by using the Network Function Virtualization (NFV) paradigm [2]. The basic idea of NFV is to separate software from its underlying physical hardware. By using virtualization technology, the Virtual Network Functions (VNF) can be consolidated on commodity servers. This helps the providers to quickly deploy a service on multiple hardware platforms [3], [4]. In addition, VNFs can be instantiated, operated, and/or migrated automatically on one or more commodity servers in an NFV architecture [5] without having to install new hardware. These benefits provide us with a solution for the development and implementation of a VNF for video monitoring at different locations on the network apart from using a dedicated physical device.

In this study, we examine the accuracy of the monitoring function, depending on its placement in the network. We then use a predefined mapping function to estimate the video QoE based on the number of stalling events calculated from our function. To this end, we carry out a study in several steps. First, we design a VNF as a plain software that exploits a Python library, namely Scapy[1] to capture the video flows at network interface. We then design an algorithm to estimate the video buffer and detect the stalling events based on timestamps of streamed packets. In the second step, we set up a local testbed with two scenarios to assess the impact of the function placement on its accuracy. In both scenarios, we validate the accuracy of the function by comparing its estimate to the actual video quality that is monitored at the client web browser. The result is the accuracy of the estimate depending on the placement of the VNF in the network, which is our main contribution of the paper.

The remainder of the paper is structured as follows. In Section II, we present the background of the study and highlight related works. Our estimation algorithm, research methodology and measurement setup are described in Section III. The main outcomes of the study are presented in Section IV, and Section V concludes this work.

## II. Background and Related Work

In this section, we briefly introduce HTTP adaptive video streaming and we then highlight an overview of related works.

[1] http://www.secdev.org/projects/scapy/

### A. Background

HTTP Adaptive Video Streaming (HAS) was developed to overcome the traditional inefficient streaming technology such as Real-Time Streaming Protocol or progressive download. The main idea of HAS is breaking a video into one or more consecutive non-overlapping periods that are seamlessly streamed to client web browser over HTTP [6]. These periods are so-called video segments or chunks that are described in a manifest file named Media Presentation Description (MPD). By streaming separated video chunks to the client, the streaming server can adapt the quality of each video chunk to the network condition.

The MPD is transmitted to the client after a video request via HTTP. This file contains information about all video segments, such as segment length, resolutions, frame bit rates, etc. Before the client playing back the video on a player, at least the first segment is fully downloaded to the client, while downloads of subsequent segments can still be ongoing. All the downloaded segments are stored in the client's video buffer. These buffered data are consumed during the video playback and filled up when a new segment is fully downloaded. If the video buffer is empty during the playback, an interruption of the video playback occurs, i.e., stalling. In HAS systems, the video bit rate may be reduced to avoid stalling events when the network condition degrades.

### B. Related Work

The deployment location is one of the most important VNF configurations. Where to place the VNF to meet the requirements of resources, high performance, network efficiency is an emerging topic in research. In [7], Clayman et al. contribute a Placement Engines software installed in the orchestration layer of an NFV architecture. This algorithm decides where to place the virtual routers to meet an adaptive resource utilization. The placement problem of virtual Deep Packet Inspection (vDPI) functions is presented in [8] for SDN and in [9] for NFV. By solving the placement problem using genetic algorithms for SDN and Integer Linear Program for NFV with cost constraints, respectively. The authors conclude that an appropriate placement of vDPI depends on functional targets, operation cost, and the number of instances as well.

Concerning to video monitoring, in [10], Wamser et al. model the YouTube stack at three levels of transport, application, and user. These models may help service providers to understand the functionality of YouTube from controlling video data flows to user perceived Quality of Experience. In [11] and [12], the authors introduce the YoMo application for YouTube monitoring with respect to user perceived QoE. A QoE model for video streaming based on stalling frequency and length is reported in [13]. The approach of monitoring YouTube video based on DPI is presented in [14], [15], these studies are similar with ours in the method of using DPI to parse video flows. Nevertheless, the authors focus more on QoE monitoring rather than deployment location. In our study, we however concern to the accuracy of the VNF for video monitoring depending on different placements.

## III. Methodology and Measurement Setup

In this section, we first present the description of our estimation algorithm and two scenarios. We then highlight our research methodology and describe the measurement setup for both scenarios.

### A. Video Estimation Algorithm

In [16], Seufert et al. describe the QoE influencing factors of HAS, which are initial delay, stalling, and quality adaptation. To achieve the goal of monitoring QoE for the video, we design an algorithm to estimate these parameters based on the extracted HTTP payload of video flows. Figure 1 shows the concept of our algorithm.

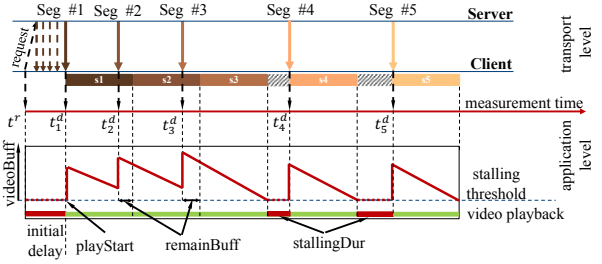



Fig. 1. Graphical View of Estimating Video Quality Algorithm

The diagram can be divided into two levels. At transport level, packets are captured and parsed to feed all necessary information for the algorithm such as IP address, TCP header and the payload of application layer protocols. The video flows are then detected by filtering a set of pre-defined keywords contained in the payloads of HTTP. The video resolutions and segments length are recorded by matching the requested segments with the MPD file. Thereafter, all downstream packets composing a segment are grouped and analyzed based on their common TCP acknowledgment number. From the figure, the solid vertical arrows represent the timestamps $t_i^d$ of downloaded segment $i$, the solid rectangles depict the corresponding segment length $s(i)$ extracted from MPD.

At application level, `playStart` is timestamps when the video starts to play out, `playStart` $\approx t_1^d$. In fact, through measurements on client, we observed that the video starts playback almost immediately after the first segment is fully downloaded. `videoBuff` is the amount of video stored in buffer memory for playing out which is illustrated by a solid zig-zag line. It decreases during video playback and is filled up when a new segment is fully downloaded. When the `videoBuff` is empty then it reaches stalling threshold, the video playback is interrupted and stalling occurs. `remainBuff` is amount of video remaining in the memory before it is filled up by a new segment. It is a trigger to determine whether a stalling event occurs or there remains video buffer in the memory. `stallingDur` is duration of a stalling event.

In the following we present a simplified algorithm for estimating video buffer and stalling events.

**Algorithm** Estimating Video Buffer and Stalling Events

Estimate video starting time
$\texttt{playStart} = t_1^d$
Calculate first available playback time
$\texttt{avlPlayBackTime} = s(1)$
**for** Next segment **do**
  Estimate the amount of played out video
  $\texttt{playOut} = t_i^d - \texttt{playStart}$
  Estimate the remaining video buffer
  $\texttt{remainBuff} = \texttt{avlPlayBackTime} - \texttt{playOut}$
  **if** $\texttt{remainBuff} <= 0$ **then**
    Record one stalling event
    $\texttt{stalingDur}(i) = t_i^d - t_{s(i-1)}^{end}$
  **else**
    Record estimated video buffer at segment $i$
    $\texttt{videoBuff}(i) = \texttt{remainBuff}$
  **end if**
  $\texttt{avlPlayBackTime} = \texttt{avlPlayBackTime} + s(i)$
  $t_{s(i)}^{end} = t_i^d + s(i)$
**end for**

We first estimate the video starting time `playStart` when the first segment is fully downloaded. The available playback time `avlPlayBackTime` is cumulatively summed by segment length $s(i)$ when a new segment is downloaded. When a new segment comes, we assume that `playOut` is the amount of played out video calculated by the download timestamps of the segment and the starting timestamps `playStart` of the video. With `playOut` value we can estimate the remaining video buffer `remainBuff` by its difference with the available playback time. The number of stalling events and length are then recorded for QoE estimation which is presented in Section IV-B. Finally, `videoBuff` is saved in log files that are used to visualize the behavior of video buffer under the impact of network condition and function placement.

### B. Scenarios

The Edge Server (ES) scenario and Data Center (DC) scenario are depicted in Fig. 2.


Fig. 2. Overview of Scenarios

From the figure, video content is assumed to be stored in a data center. In ES scenario, our function is deployed on an edge server to monitor all the video flows passing through from the streaming server to the client. The edge server is an essential instance of the Edge Computing paradigm, which refers to the enabling technologies allowing computation to be performed at the edge of the network close to the users [17]. In the DC scenario, we deploy the monitoring function nearby the data center. In this scenario, the function can utilize a high data rate traffic from data center network. However, it is placed far from the client, which may cause some performance problems due to the degradation of network conditions on the path to the client. In both scenarios, we deploy the same VNF monitoring.

### C. Methodology

Our study copes with the need to monitor video streaming in the network. To implement the VNF monitoring, we use a dedicated testbed including a network emulator and a middle-box to compose the two scenarios. The VNF is installed on the middle-box to sniff the video flows and outputs necessary video information to feed our algorithm. We assume that in between the server and the edge is a long distance with various network segments and routers. The degradation of the network can be considered as the combination of high round trip time and congestion. We therefore shape the video traffic by using NetEm [18]. This Linux-based software can adjust different network parameters to evaluate the impact of network QoS on the service or application in general. To validate the video quality estimated by the function, we compare it to actual video quality obtained from the client. This actual video quality is measured by using a Javascript-based web API.

### D. Measurement Setup

The measurement setup for the ES scenario is schematically depicted in Fig. 3, and consists of one *NetEm* server and three PCs. One PC is used to install our VNF monitoring named *MoniV*, another PC is used for the *Client* that browses the videos via a testbed network, and we use a *Control PC* that is connected remotely to the testbed via a dedicated control network to avoid interference with the experiments.

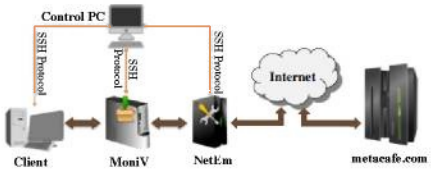



Fig. 3. Overview of the Testbed in Edge Server Scenario

The *NetEm* is running on a SUN FIRE X4150 server. This server uses Ubuntu 12.04 LTS while other PCs use Ubuntu 14.04 LTS as operation systems. The testbed is connected to the Internet via a research network. To achieve high reliability in timestamps calculation, in both scenarios we synchronized the clocks of *Client* and *MoniV* machines to the same NTP server. For the DC scenario, we use all the same devices and only change the positions of the *MoniV* and the *NetEm*. Thus, the *MoniV* is placed near the data center in this scenario. At the implementation, a Python measurement script is used to start the monitoring function at the *MoniV*, then we use the Selenium Webdriver[2] to automatically browse the video on the *Client* after adjusting the pre-defined link capacity parameters on the *NetEm*. This ensures the network to be configured before the client starting to play the video.

[2]http://www.seleniumhq.org

We choose *metacafe.com*[3] as the video streaming source, as they provide unencrypted HAS streaming. This provides us an easy way to extract the video information. We implement measurements on 10 different videos which have various types of content and lengths, but offer the same three levels of resolution, i.e, 240 p (428 × 240), 360 p (570 × 320), and 720 p (1280 × 720). Since we want to investigate the accuracy of the function in all possible behaviors of video playback. We shape traffic on the *NetEm* which may cause some possible influences on the video quality as shown in Tab. I.

TABLE I
SHAPED LINK CAPACITIES ON NETEM

| Link Capacity | Possible impacts on video playback |
|---|---|
| 512 kbit/s | Stalling occurs sporadically |
| 1 mbit/s | No stalling with small video buffer |
| 10 mbit/s | No stalling with large video buffer |

## IV. RESULTS

Based on the methodology and the testbed setup presented in Section III, we have implemented several measurements between September and October 2016 at the University of Würzburg. To secure the stability of the monitoring function, we tested on 10 different videos with 5 replications each.

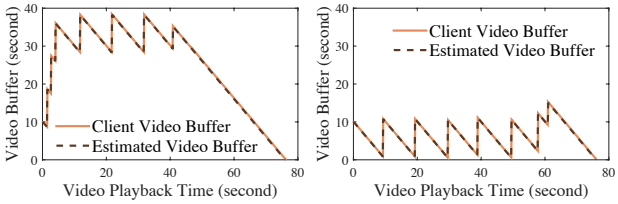
For the sake of intuitively comparing video buffer estimated by the function and obtained from client in two scenarios, we present in following the measurement results of one typical video which has total length of 75 s. To increase statistical significance of the measurements for this video, we produced 30 replications at every link capacity configuration as shown in Tab. I. After the experiment, we collected 180 log files given by the sniffing task of the function in both scenarios and a corresponding number of video buffer sampling logs on the client. To minimize missing the state of video buffer, we did sampling on the client every 100 ms. Using the algorithm detailed in Section III-A, we conducted the estimation task with all extracted log files. To calculate and compare the estimated and actual video buffer on the client, we use the timestamps of the first downloaded segment extracted from sniffing logs as the time reference for video playback start.

### *A. Accuracy of Video Buffer and Stalling Estimation under Stable Network Conditions*

Figure 4 and Figure 5 show the behavior of video buffer at different link capacities in the ES and DC scenarios. In all sub-figures, the $x$ axes indicate the video playback time while the $y$ axes show the video buffer. The dashed lines depict the estimated video buffer provided by the function and the solid lines show the actual video buffer extracted from the client browser. From the figures, it is obvious that the video buffer is smaller at lower bandwidth.

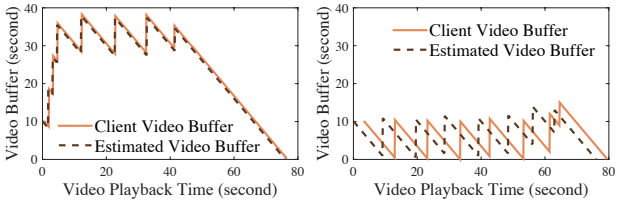
Considering our estimation function in comparison with the actual video buffer extracted from the client browser, it can be seen from Fig. 4 that our function has high accuracy in estimating the video buffer as well as detecting the stalling

[3]http://metacafe.com




(a) Link capacity of 10 mbit/s (b) Link capacity of 512 kbit/s

Fig. 4. Video Buffer over Playback Time in ES Scenario




(a) Link capacity of 10 mbit/s (b) Link capacity of 512 kbit/s

Fig. 5. Video Buffer over Playback Time in DC Scenario

events in the ES scenario. Due to the placement of the function near the client, the difference in arrival time of video flows is negligible at both machines, even at a low bandwidth of 512 kbit/s. Similarly, Figure 5a also shows a good estimation at high bandwidth of 10 mbit/s in the DC scenario. Although the function is placed far from the client, it can still estimate approximately the buffer of video playing on the client. This is because the bandwidth in this scenario is high, such that the packet arrival times at both machines are almost the same, which provides us a good estimation with small error. However, Figure 5b shows a bad fit in the DC scenario for a bandwidth of 512 kbit/s. As the function is located near the streaming server, it can utilize a high data rate and receive packets of the segments shortly after the client requests them. Due to the bandwidth limitation of the link after the VNF, the reception of these packets at the client is delayed. This induces the estimation error since the function calculates the video buffer based on timestamps of downloaded segments.

Figure 6 shows the cumulative distribution function (CDF) of root mean squared error (RMSE) between the estimated and actual video buffer at different bandwidth in two scenarios. The figure points out that 90 % of estimated samples in ES scenario have errors less than 1 s compared to baseline values.

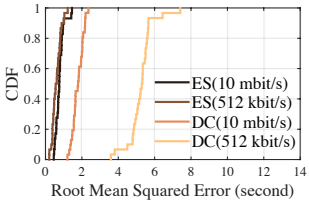



Fig. 6. RMSE Between Estimated and Actual Video Buffer

In the DC scenario, the accuracy of the function is acceptable as only $20\,\%$ of estimated samples have error more than $2\,\mathrm{s}$ at bandwidth of $10\,\mathrm{mbit/s}$. This demonstrates that at high link capacity, our function can still estimate approximately the buffer of the video playing on the client. Conversely, at lower link capacity of $512\,\mathrm{kbit/s}$, the function estimates the video buffer with high error where most of the estimated values have error larger than $4\,\mathrm{s}$.

Table II shows the mean errors with $95\,\%$ of confidence interval between the estimated and actual buffer of the other videos with various lengths. However, we only present in this table the results of measurements at the bandwidth of $512\,\mathrm{kbit/s}$ due to at higher bandwidth, the mean errors are negligible similar to our above conclusion. The table indicates that through different videos, our function can still estimate the videos buffer with high accuracy in ES scenario. In DC scenario, the mean errors are higher and various over different videos due to the differences in the number of segments and frame bit rates among themselves.

TABLE II
MEAN ERROR OF VIDEO BUFFER AT BANDWIDTH OF $512\,\mathrm{kbit/s}$

| Video ID | Length | ES Scenario | DC Scenario |
|---|---|---|---|
| 11419867 | 201 s | $0.84\,\mathrm{s} \pm 0.033$ | $2.81\,\mathrm{s} \pm 0.051$ |
| 11420085 | 126 s | $0.65\,\mathrm{s} \pm 0.039$ | $2.94\,\mathrm{s} \pm 0.059$ |
| 11419885 | 95 s | $0.46\,\mathrm{s} \pm 0.037$ | $2.54\,\mathrm{s} \pm 0.071$ |
| 11419888 | 71 s | $0.57\,\mathrm{s} \pm 0.041$ | $2.21\,\mathrm{s} \pm 0.085$ |
| 11419938 | 126 s | $0.52\,\mathrm{s} \pm 0.035$ | $2.78\,\mathrm{s} \pm 0.065$ |
| 11420011 | 67 s | $1.13\,\mathrm{s} \pm 0.068$ | $2.57\,\mathrm{s} \pm 0.078$ |
| 11420017 | 158 s | $0.54\,\mathrm{s} \pm 0.029$ | $4.47\,\mathrm{s} \pm 0.047$ |
| 11420019 | 305 s | $0.99\,\mathrm{s} \pm 0.036$ | $2.94\,\mathrm{s} \pm 0.058$ |
| 11420073 | 202 s | $0.52\,\mathrm{s} \pm 0.029$ | $3.12\,\mathrm{s} \pm 0.051$ |

In [13], Hoßfeld et al. contribute a study of QoE for YouTube video using subjective crowd test. They investigated the impact of stalling parameters, i.e., frequency and length on the users' perceived QoE. The authors proposed exponential fitting functions to quantify the QoE impact of stalling as

$$f_1(N) = 3.26 \cdot e^{-0.37 \cdot N} + 1.65 \quad (1)$$

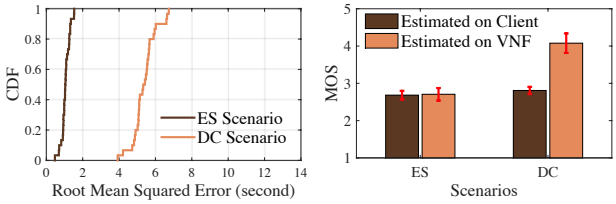
$$f_3(N) = 2.99 \cdot e^{-0.96 \cdot N} + 2.01 \quad (2)$$

where $f_1(N)$ and $f_3(N)$ are the functions of MOS given by the number of stalling events $N$ with stalling length of $1\,\mathrm{s}$ and $3\,\mathrm{s}$, respectively. It can be seen from the equations that the MOS only depends on the number of stalling events and length. In previous measurements, we observe that there is only constant time shifting in estimated video buffer and our monitoring function can estimate exactly the number of stalling events and length. Thus, our VNF is able to accurately estimate the QoE in both scenarios.

### B. QoE Estimation under Unstable Network Conditions

In the following, we consider another scenario where network condition is unstable, i.e., packet reordering. In this scenario, packets belonging to one video segment may arrive at the client out of order. Packet reordering is also reported in several studies, e.g., in [19], Leung et al. describe five major causes of packet reordering. In [20], Gao et al. argue that the packet reordering occurs with probabilities usually more than $30\,\%$ in concurrent multipath transfer system. The network heterogeneity and the use of multiple links in wireless networks also causes packet reordering which is described in [21].

To investigate the impact of packet reordering on the VNF monitoring for QoE, we have done several measurements with the same testbed. Specifically, in both scenarios we configured the NetEm to immediately dequeue $25\,\%$ of the packets, the others are delayed by $500\,\mathrm{ms}$ and the link was set to $512\,\mathrm{kbit/s}$. Figure 7a shows the CDF of RMSE between the estimated and actual video buffer. It can be seen that the error is similar to our previous measurements shown in Fig. 6.




(a) CDF of RMSE (b) QoE Estimation

Fig. 7. The Accuracy of VNF at 512 kbit/s with Packet Reordering

However, in these measurements, the function additionally fails to correctly estimate the number of stalling events. Figure 7b depicts the QoE estimation based on number of stalling events. The $x$ axis shows the measurements in two scenarios, the $y$ axis indicates the estimated MOS values calculated by Equation (1). The MOS can take the following values: (1) bad; (2) poor; (3) fair; (4) good; (5) excellent. The darker bars describe MOS values calculated from the actual number of stalling events counted at the client, while the other MOS values are estimated by the function. The MOS values are averaged over 30 replications with $95\,\%$ of confidence interval. The figure shows that in the DC scenario, the function estimates a smaller number of stalling events, which is represented by the high average MOS value of 4.07. Meanwhile the MOS calculated in ES scenario and on the client are smaller than 3. It can be concluded that, in the case of packet reordering, the function does not work properly if it is placed far away from the client.

## V. CONCLUSION

The rapid growth of video streaming offers video providers with a great opportunity to increase their revenues, but also presents challenges for the network providers. Network providers are typically not aware of the actual quality experienced by users of video streaming services. As a result, it is difficult for providers to adapt in the network when the QoE is low due to network impairments. To solve this problem, a monitoring mechanism in the network is needed to estimate the QoE perceived by the users. NFV has become an emerging network technology, which promises to quickly deploy a service at several hardware platforms and can also be utilized for QoE monitoring in the network. Nevertheless, NFV

is still on early stage and its performance and reliability need more research efforts to be commercialized. In particular, the placement of a VNF is one of the most important influencing factors for its performance.

In this paper, we proposed a VNF monitoring for QoE of video streaming in the network and evaluated its accuracy depending on different placements. We developed a function using a DPI-like method to estimate the video buffer based on download timestamps of video segments. Our function can also detect stalling events and estimate their duration, which serves as input parameters for QoE models. We then set up a dedicated testbed with two scenarios to evaluate the influence of different geographical placements of the VNF on its accuracy. The measurement results are validated by extracting and comparing the actual video quality from the client's web browser. Our results for stable network conditions show that in the ES scenario, where the function is placed near the client device, we can estimate the video buffer and stalling events with high accuracy. In contrast, when the function is placed near the streaming server in the DC scenario, we can accurately estimate the video buffer only at high link capacity of $10\,\mathrm{mbit/s}$ in our measurements. At lower link capacities, the buffer estimation has a constant time shift. In case of unstable networks with packet reordering, additional errors in estimating the stalling occur in the DC scenario and lead to an inaccurate QoE estimation. Based on these findings, we believe that in an NFV architecture, the VNF for monitoring video QoE can only achieve high accuracy if it is placed near the clients. For future work, we will extend our study to be able to monitor encrypted video traffic, which is nowadays widely used, e.g., by YouTube.

## ACKNOWLEDGMENT

This work was partly funded in the framework of the EU ICT project INPUT (H2020-2014-ICT-644672) and the DFG grant QoE-DZ (TR257/41). The authors alone are responsible for the content.

## REFERENCES

[1] C. Systems, "Cisco visual networking index: Forecast and methodology, 2015-2020," White Paper, 2016.

[2] M. Chios, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, and H. Deng, "Network functions virtualisation: an introduction, benefits, enablers, challenges and call for action," White Paper available at https://portal.etsi.org/nfv/nfv_white_paper.pdf, 2012.

[3] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, 2015.

[4] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, 2015.

[5] E. G. N. 002, "Network functions virtualisation (nfv); architectural framework," Group Specification available at http://www.etsi.org, 2014.

[6] T. Stockhammer, "Dynamic adaptive streaming over http: Standards and design principles," in *Proceedings of the Second annual ACM Conference on Multimedia Systems*. Santa Clara, CA, USA: ACM, Feb 2011.

[7] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *2014 Network Operations and Management Symposium (NOMS)*. Krakow, Poland: IEEE, May 2014.

[8] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized deep packet inspection functions in sdn," in *MILCOM 2013 IEEE Military Communications Conference*. San Diego, CA, USA: IEEE, Nov 2013.

[9] M. Bouet, J. Leguay, T. Combe, and V. Conan, "Cost-based placement of vdpi functions in nfv infrastructures," *International Journal of Network Management*, vol. 25, no. 6, 2015.

[10] F. Wamser, P. Casas, M. Seufert, C. Moldovan, P. Tran-Gia, and T. Hoßfeld, "Modeling the youtube stack: from packets to quality of experience," *Computer Networks*, 12 2016.

[11] B.Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle, "Yomo: A youtube application comfort monitoring tool," in *New Dimensions in the Assessment and Support of Quality of Experience for Multimedia Applications*, Tampere, Finland, 6 2010.

[12] B. Staehle, M. Hirth, F. Wamser, R. Pries, and D. Staehle, "Yomo: A youtube application comfort monitoring tool," University of Wuerzburg, Tech. Rep. 467, 3 2010.

[13] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of youtube qoe via crowdsourcing," in *IEEE International Symposium on Multimedia (ISM2011)*. California, USA: IEEE, Dec 2011.

[14] R. Schatz, T. Hoßfeld, and P. Casas, "Passive youtube qoe monitoring for isps," in *The Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2012)*. Palermo, Italy: IEEE, Jul 2012.

[15] P. Casas, R. Schatz, and T. Hoßfeld, "Monitoring youtube qoe: Is your mobile network delivering the right experience to your customers?" in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. Shanghai, China: IEEE, Apr 2013.

[16] 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 and Tutorials*, vol. 17, Mar 2015.

[17] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[18] S. Hemminger, "Network emulation with netem," in *Australia's National Linux conference*, Canberra, Australia, apr 2005.

[19] K. C. Leung and D. Yang, "An overview of packet reordering in transmission control protocol (tcp): Problems, solutions, and challenges," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 4, 2007.

[20] C. Gao, Z. Ling, and Y. Yuan, "Packet reordering analysis for concurrent multipath transfer," *International Journal of Communication Systems*, vol. 27, no. 12, 2014.

[21] D. Kaspar, K. Evensen, A. F. Hansen, P. Engelstad, P. Halvorsen, and C. Griwodz, "An analysis of the heterogeneity and ip packet reordering over multiple wireless networks," in *IEEE Symposium on Computers and Communications(ISCC)*. IEEE, 2009.