Leveraging SDN to Provide an In-network QoE Measurement Framework

Arsham Farshad*, Panagiotis Georgopoulos†*, Matthew Broadbent*, Mu Mu* and Nicholas Race*
*School of Computing and Communications, Lancaster University, UK
†Communication Systems Group, ETH Zurich, Switzerland
a.farshad@lancaster.ac.uk, panos@tik.ee.ethz.ch, m.broadbent@lancaster.ac.uk, m.mu@lancaster.ac.uk, n.race@lancaster.ac.uk

***Abstract*—Online video streaming using HTTP Adaptive Streaming (HAS) is becoming the most popular content delivery mechanism for media services. Network and content providers would like to ensure a high degree of video Quality of Experience (QoE) for their end-users. However, traditional network-level metrics do not necessarily reflect the end-users' true perception of delivered content. We introduce an in-network QoE measurement framework (IQMF) that provides QoE monitoring for HAS streams as a service. The framework leverages Software Defined Networking for its control plane functionality to streamline non-intrusive quality monitoring and to offer a closed control loop for QoE-aware service management. IQMF adopts two specifically designed QoE metrics to capture the user experience of HAS streams with respect to video fidelity and switching impact. Finally, we used a pan-European SDN testbed to demonstrate how IQMF can be used as a foundation for in-network QoE measurement and service optimisation.**

I. Introduction

Online video streaming has seen a huge growth in popularity during recent years. In 2013, live and on-demand Internet video traffic represented 66% of all global Internet traffic. It is expected that this trend will continue further and video traffic will be 79% of all Internet traffic by 2018 [1]. At the same time, High Definition (HD) video traffic has already surpassed that of Standard Definition (SD), and with the introduction of Ultra-High Definition content, providers will continue to push user expectations in the availability of higher video quality and bitrates. Undoubtedly, high quality video streaming has become an essential part of many consumers' lives.

Network and content providers are thus immensely interested in ensuring a high degree of video Quality of Experience (QoE) for their end-users. Network level metrics (e.g. bandwidth, latency, packet loss etc.), traditionally used by network administrators, are not adequate to indicate how satisfied a user is with their video streaming experience. In addition, research shows that there is not always a direct or deterministic correlation of the impact of the network-level metrics to the users' QoE [2]. Thus, the evaluation of video streaming should be based on user-centric QoE metrics (e.g. startup time, average playback bitrate, etc. [2]) that provide a better indication of the satisfaction of the end-users.

Meanwhile, HTTP Adaptive Streaming (HAS) protocols, such as Microsoft Smooth Streaming, Apple HLS, Adobe HDS and MPEG-DASH [3], are becoming very popular for online video streaming. This popularity is due to their unique ability to adaptively select a bitrate to maximise video quality, minimise video pauses and reduce buffering times for better overall QoE. However, such bitrate adaptation operates solely on the end-user's device without any coordination with network management, leading to a sub-optimal network-wide QoE experience [4]. Overall, enabling network-wide QoE monitoring requires an in-network transparent measurement framework that does not require a user's involvement (such as installing plug-ins on their devices). Such a framework should effectively and accurately assess the impact of changes in the network to the video streaming experience of the user and provide a closed control loop that allows QoE-aware service optimisations to take place at run-time.

This paper introduces an in-network QoE Measurement Framework (IQMF) that provides live network-wide QoE measurements. The framework does not require the users' participation in the process, as it monitors relevant streams within the network itself. IQMF adopts two specifically designed QoE metrics to quantitatively measure the user experience related to video fidelity and representation switching impact over HAS streams. IQMF offers these measurements and the respective analysis as a service to the network provider or content distributor via an API. This service can be exploited for different monitoring and management purposes including as a feedback loop for QoE-aware service optimisation.

To achieve the required functionality within a network, we leverage Software Defined Networking (SDN) to provide the control plane interaction for the framework. This enables IQMF to measure and analyse the end-users' QoE in a flexible and deterministic way. In addition, IQMF uses the dynamic traffic management that SDN provides to offer performance and scalability when deploying additional measurement agents that can all be controlled by the same measurement controller. Our deployment of IQMF in a large-scale pan-European SDN testbed and the respective experiments, demonstrate how IQMF can be exploited in order to assess and optimise HAS video distribution.

The remainder of the paper is organised as follows. Section II provides use cases that motivate our work, whilst Section III presents related work in this problem domain. Section IV introduces IQMF, its design, implementation and QoE metrics. Evaluation is discussed in Section V and finally, Section VI concludes our work.

II. Use Cases

The design of the QoE measurement framework is motivated by the challenges faced in a number of real-life use cases:

1) Video-on-Demand Caching Optimisation: Most ISPs or last mile networks employ caching services to optimise the delivery of content to end-users. A QoE framework could assist network administrators in monitoring the QoE of a Video-on-Demand (VoD) streaming service and use such intelligence to optimise the distribution of VoD content between cache servers or to use intelligent traffic engineering and routing techniques to improve QoE dynamically. OpenCache [5] is just one example of an OpenFlow-based in-network caching service, that could highly benefit from in-network QoE measurements to further optimise its caching behaviour. The framework could provide measurements as input to any caching service on the last mile or even, given appropriate SLAs, to Content Delivery Networks (CDN) to optimise their content placement and delivery [6], [7].

2) Live Streaming Optimisation: A QoE framework can provide real-time QoE statistics of live video streaming services in a network. Different run-time techniques could then be used to further optimise streaming based on the QoE measurement, such as interacting with the queueing disciplines on network switches (e.g. using the corresponding OpenFlow-based interface [8]) or invoking QoS mechanisms. The live QoE feedback provided by the QoE framework can also be exploited to avoid the over provisioning of network resources when a target QoE level is met.

3) Network Utilisation Optimisation: There are multiple routing schemes that are used in-network to optimise packet delivery and ensure appropriate load-balancing of network links. Quantitative QoE evaluations of different network reconfigurations must be in place to determine the optimal solution to allocate network resources. With a QoE framework, an administrator could preemptively estimate the impact of certain network control to the user experience of related media streams and deploy the best configurations accordingly.

III. Related Work

To enhance video content delivery as well as increase the QoE of the end-users, [9] proposed the Server and Network Assisted DASH (SAND) architecture. SAND is a control plane for video delivery that obtains QoE metrics from the users (clients) and returns network-based measurements to help the clients enhance their overall QoE. The third-party measurement server in SAND, known as DANE (DASH-assisting network element), provides measurement information to the different parties in the delivery chains including CDNs, ISPs and content providers. However, the heterogeneity of the end-user devices (e.g. smartphones, tablets, laptops, IPTVs etc.) and the diversity in video playback applications make retrieving QoE measurements challenging. Realising architectures that are reliant on end-user device feedback, such as SAND, can therefore be particularly difficult to implement, and hence only a few works in the literature have tackled this problem this way.

MintMos is an application-independent client-side QoE measurement tool [10]. MintMos is a Linux kernel module loaded by video applications on the client's device to infer the QoE of a video stream as it passes up the protocol stack to the application. To subjectively estimate the QoE, MintMos uses the constructed k-dimensional QoE space. The authors of [11] propose a client-independent and network-side measurement approach for YouTube video traffic, using passive traffic measurements. QMON [12] enhances previous work by supporting more video codecs and by having a more advanced measurement approach. Specifically, it estimates the playout buffer levels and the number of stalls and their duration, and then calculates the MOS (Mean Opinion Score) for the played YouTube video. To identify the stalling, the authors of QMON use the TCP segment timestamp and the timestamp on the video and then estimate the buffer fill up levels or its depletion. This measurement is obtained by conducting deep packet inspection on the path the YouTube traffic is routed via. QMON therefore only measures the stalling effect and can only be used for YouTube traffic.

Our proposed framework is different from related work (e.g. QMON, MintMos etc.) in a number of ways: it is designed to address an additional number of QoE measurement metrics; it only captures URL addresses and manifest files from video traffic (hence requiring significantly less processing resources comparing to those frameworks reliant on deep packet inspection); and can easily be deployed and controlled by leveraging SDN features of the network. Last but not least, it provides the QoE measurements as a service that can be used by third party applications and network elements.

IV. QoE Measurement Framework

In this work, we look to build a QoE measurement framework *within* the network. As such, a number of functional building blocks (illustrated in Figure 1) are being brought together to form the IQMF framework. At the core of this framework is the measurement layer. This contains a number of measurement agents that provide the actual QoE monitoring (the implementation of which is discussed in Section IV-A). Importantly, the measurement layer utilises functionality in the underlying network layer, which uses a Software Defined Network (SDN) to duplicate specific flows of interest to the measurement agent for analysis.

Coupling these measurement and network layers together is the control layer. This contains controllers for both the network layer (SDN controller) and the measurement layer (measurement controller). Through communication between these two controllers, the measurement controller renders the changes necessary to redirect flows in the network layer. The measurement controller can also control the measurement taking place on the agent. This is done through the API, which is described in Section IV-B.

The measurement agents will report information back to the measurement controller. This information can then be requested by the application layer, which may contain a number of such applications. Furthermore, these applications may

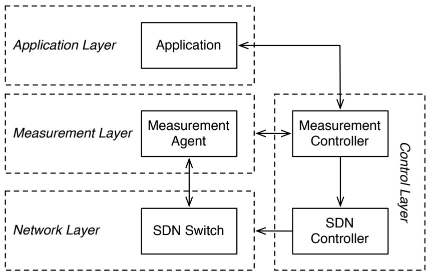


Fig. 1. In-network QoE Measurement Framework (IQMF)

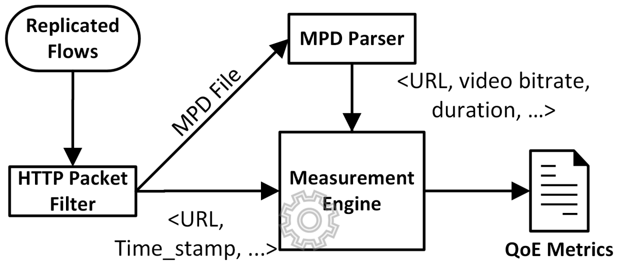


Fig. 2. Prototype Measurement Agent

then make modifications to the network, via the measurement controller, to the SDN controller. This pass-through is done to ensure a level of consistency between the operation of the framework and any requested changes.

A. Implementation

As mentioned previously, IQMF leverages an underlying SDN to provide the flow duplication necessary for QoE monitoring. More specifically, IQMF interacts with an existing OpenFlow controller responsible for maintaining the forwarding behaviour of the network. Although the initial implementation is limited to one SDN-technology (OpenFlow [8]), providing the necessary functionality is available with other technologies; there is little reason why they could not be used as an alternative.

Once the necessary video streaming related traffic has been replicated toward the measurement agent, it will be captured and processed in two distinct stages, as illustrated in Figure 2. Initially, traffic is passed through an HTTP packet filter in order to identify HTTP GET requests and responses. This separation prevents the management agent from processing unnecessary information. Once this is complete, the management agent will examine the GET requests in order to identify any manifest files present (e.g. the Media Presentation Description file (MPD)). These are then passed to the specific parser for further inspection. These manifest files are particularly important, as their presence typically represents a client about to start, and in most cases their presence is required before playback can commence.

The MPD parser will extract information from the MPD file, including a description of the different representations. This includes references to a multitude of qualities, resolutions and playback codecs. The manifest file also contains other useful information, such as the total duration of playback. Once parsing is complete, the parser will pass this information to the measurement engine.

With this information, the measurement engine can track the behaviour of a client during playback. This is carried out by combining this information with additional details from the HTTP packet filter. More specifically, the measurement engine will observe the content that a client is requesting, match this with the information in the equivalent manifest file, and thus ascertain various metrics related to their QoE. For example, from a client request IQMF will obtain the URL of the content the client wishes to retrieve. Without the information from the manifest file, it is impossible to contextualise this information from just the URL. In particular, it is hard to accurately determine the playback quality of the request without additional information. With the use of IQMF, this information is available, thanks to preemptive parsing of the manifest, and as such, we can identify a particular request with its associated quality level.

By periodically tracking this information, and intentionally linking it back to the context described in the manifest file, we can provide a number of metrics on a per-client and per-session basis. These can then be used to form a feedback loop. Using the QoE metrics ascertained through the agents, the measurement controller can modify the underlying network to remedy some of the deficiencies in QoE. As the underlying network is software defined, the measurement controller can modify the forwarding plane, through the SDN controller, to better serve a client's requests. This is carried out to intentionally improve a client's QoE. The process is cyclical: as the measurement controller redirects flows, it will also be able to observe, through its own measurement agents, the effect of these changes. This allows the measurement controller to determine the success of a particular action.

B. API

The measurement framework's controller exposes an API which allows other applications to access information derived from the operation of the framework. This API contains a number of methods, including `start`, `stop` and `stat`. These are used to control the measurement framework and explicitly define the content that is of interest. This is done by specifying a manifest file in a call to the API, using the `start` method. This manifest will contain the URLs of the content in question. This allows an application or user to define specifically the content that they want to monitor the QoE for. Similarly, the `stop` method is used to halt the monitoring of QoE for a given manifest file. The `stat` method exposes the core reporting functionality of the measurement framework. Using this method, an application can retrieve any of the previously mentioned metrics for a given client or set of content.

C. Performance & Scalability

IQMF takes into account a number of performance and scalability considerations. As the replication of flows is done at the network layer, there is no associated performance impact on this process. As such, the user will notice no change in their experience, and the framework itself will not adversely effect QoE. In addition to this, the flow replication process can also be to control the granularity of information; the flows redirected to the measurement agents can include all traffic, or can be limited to specific flows of interest. By determining the volume of information a measurement agent receives, the measurement controller can dynamically manipulate the load on an agent. This is important if the measurement agent is in any way resource constrained, or has an upper threshold for the amount of flow information it can parse simultaneously.

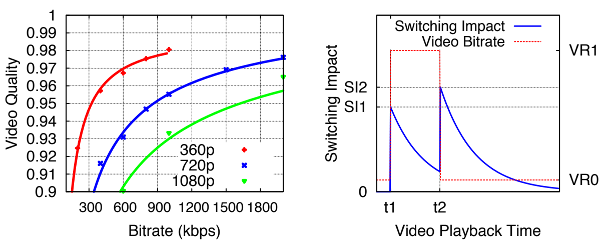
The metrics observed by IQMF can also be updated in near run-time as the analysis is a constantly ongoing process, with the measurement controller being updated regularly. The freshness of a metric is key to providing a suitable response to any potential issues that may be significantly reducing the QoE for one or more clients. If an action is deemed necessary, but informed using stale metrics, modifications may be made on a network that is not necessarily in the same state, potentially compromising the situation further.

Due to its controller-node architecture, it would be simple to add additional measurement agents to a deployment of the IQMF framework. They simply need to connect to the same measurement controller to be included. This architectural arrangement is especially interesting when you consider the availability and utilisation of an environment with rich computing resources, such as a data centre. In this case, provision can be scaled according to the load on the framework.

D. *QoE Measurement Metrics*

While the network performance and primary QoS metrics can be monitored using conventional network measurements, understanding the user experience of media streams requires tailored-made QoE functions to capture key QoE metrics of reference user applications. For HAS applications, the most recognised QoE metrics are: *quality switches*, *video quality* (picture fidelity), *startup delay*, and *stalling* [2], [13]. Accurately measuring these QoE metrics as part of a transparent in-network measurement framework, which does not require real-time feedback from user devices, is a non-trivial task. We focus on two QoE metrics *video quality* and *quality switches*, which are good examples of how QoE metrics are designed and incorporated. The modelling of *startup delay* and *stalling* is planned for future work.

1) Video quality: Using HAS-based streaming services, video content is encoded using specific frame resolution and bitrate definitions (known as representations) for both generic and specific end-user requirements (such as a mobile device). With a pre-defined encoding scheme (e.g., GoP structure), a higher encoding bitrate results in less compression loss and therefore leads to higher video quality in term of picture fidelity. Research has shown that the mapping between bitrate and video quality is non-linear [4], hence the framework exploits QoE utility functions to assess the quality of a video stream from its bitrate. Figure 3(a) shows the measurement results and the derived utility curves (for three representative



(a) Video Quality (b) Switching Impact

Fig. 3. Video Quality and Switching Impact Metrics

resolutions) of perceptual quality evaluation experiments reported in our previous work [4].

Equation 1 is the generic QoE utility function. VR denotes the video bitrate and VQ denotes the video quality within the data range of $(0, 1]$. A VQ of 1 is the maximum possible video quality (when no compression or lossless compression is applied to the content). a, b, and c are the coefficients that instantiate the utility function for certain video resolutions. For example, Equation 2 is an instance of utility function for 720p videos. The utility function is of low complexity (suitable for real-time quality assessment) and yet offers high performance. Equation 2 shows significant correlation (R^2 of 0.9988 and $RMSE$ of 0.002923) to the observed experimental results [4].

$$VQ = aVR^b + c \tag{1}$$

$$VQ_{720p} = -4.85VR^{-0.647} + 1.011 \tag{2}$$

2) Switching impact: HAS media streams are able to switch between representations as the means to adapt to the available network resource. The purpose of switches can be to increase the bitrate and hence improve video quality or to reduce the bitrate to avoid stalling. However, the switching process itself may cause disturbance to the end-user. The impact of quality switches is influenced by the *amplitude* and the *distribution* of switch events [14]. The amplitude is determined by the perception of video quality changes between representations. We define such quality change as $\Delta_{VQ} = |VQ - VQ'|$ with VQ' denoting the video quality after the representation switch. A higher change of video quality leads to more severe perceptual impact at the time of switch. In a related work, Liu et al. observed that the impact caused by an "increasing switch" is much smaller than a "decreasing switch" of the same scale [13]. The modelling of this advanced feature requires further subjective experiments, which will be carried out in our future work. A crucial aspect when modelling the HAS switching impact is the *forgiveness effect*, which captures the psychological observations that, when followed by intact content, the impact of quality distortion degrades over time. The *forgiveness effect* related to video quality degradation was first studied and modelled by Seferidis et al. [15] and Hands [16]. One of the key findings from the user ratings is that the impact of quality distortion is reduced to nearly 70% after 20 seconds. We incorporate the *forgiveness effect* (Equation 3) in our framework based on the generalised model introduced by Liu et al. [17]. Equation 3 is a function of intensity of quality changes (Δ_{VQ}) and the duration of time since a switch event $(t - t_i)$.

SI= Switching Impact

$$SI_i(t) = (\Delta_{VQ})\mathrm{e}^{-0.015(t-t_i)}, \quad t_i \textit{ is the time of quality switch } i \tag{3}$$

Figure 3(b) illustrates how switching impact is captured. The switching impact is 0 prior to any quality switch. At t_1, a first switch of representation from VR_0 to VR_1 causes the impact of SI_1. At t_2 the impact from the first switch has been greatly reduced. However, the second switch (from VR_1 back to VR_0) is added to the overall switching impact. The aggregated impact from both switches gradually diminishes as playback continues.

Depending on the quality assessment scheme, it is also possible to derive the accumulative impact over time on a media session, as depicted by Equation 4.

$$CI = \sum_{i=1,t=0}^{i=N,t=t_e} SI_i(t) \tag{4}$$

Using the modelling of video quality and switching impact as the QoE measurement metrics, we can capture the user experience of HAS media streams influenced by the absolute video quality as well as the dynamics of quality levels.

V. Experiments and Evaluation

To demonstrate the exploitation of the proposed framework for improved QoE-aware network services, we realised IQMF within a large-scale pan-European SDN testbed and evaluated a number of the use cases mentioned in Section II. Initially, we evaluated the QoE measurement information that IQMF provides in video streaming experiments and secondly, how the output of the IQMF can be used to optimise VoD content caching and distribution.

A. Experimentation Testbed & Setup

Experiments were carried out over a large-scale SDN testbed, namely GOFF, provided by GÉANT. GOFF is composed of a number of OpenFlow-capable software-based switches (i.e. Open vSwitches) and virtualised computing resources. These resources are located in five different countries across Europe. The GOFF testbed offers an ideal platform to evaluate our framework on, as it resembles operational networks. The implementation components of IQMF related to OpenFlow and the measurement agents are also directly transferable to production environments. For our experimentation, we also developed Scootplayer[1], an open-source DASH player that facilitates large-scale unattended experiments and is equipped with comprehensive logging modules.

Figure 4 shows our experimentation setup. We deployed VoD servers, caches and video clients on virtual machines across different sites. Scootplayer is deployed on the video client at the Amsterdam site. With respect to network monitoring and control, a Measurement Controller (part of IQMF)

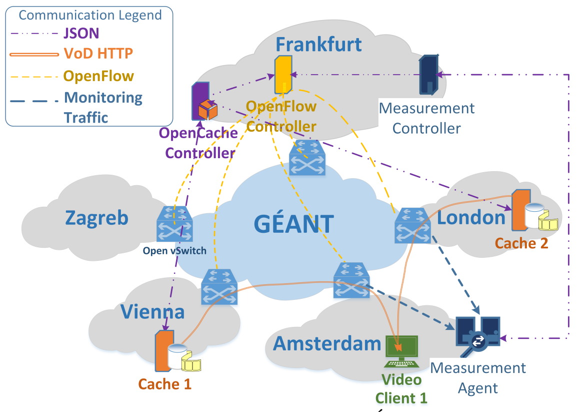


Fig. 4. The Experimentation Setup in the GÉANT OpenFlow Facility

and an OpenFlow Controller (i.e. Floodlight) are deployed in the Frankfurt site. Such a cross-site deployment allows us to truly evaluate the functionality of IQMF to provide QoE measurements assisted by SDN. To further evaluate IQMF's performance and how it captures QoE degradation caused by bad network conditions, we used a network emulator (i.e. dummynet) to introduce packet loss, delay and delay variation on the testbed's links during our experimentation.

B. IQMF; Providing QoE Measurements

We measured the QoE metrics defined in Section IV-D by deploying IQMF as depicted in Figure 4. We carried out video streaming experiments by using Scootplayer in a VM in Amsterdam that requests an MPEG-DASH version of the reference Big Buck Bunny video (approximately 10 minutes in length) from Cache 1 in Vienna. The test was repeated 20 times. The VoD traffic was monitored by the Measurement Agent hosted in the same site as Scootplayer. Measurement statistics are reported to the Measurement Controller and are accessible through the designed API.

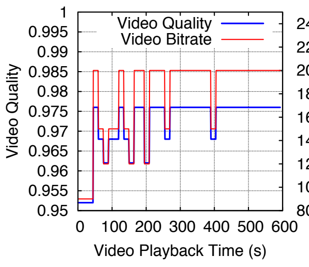
Table I gives an overview of the indicative QoE measurements that the IQMF reports during this experiment. Furthermore, IQMF derives the two bespoke QoE metrics, Video quality and Switching impact, which better characterise the QoE of the end-users based on the particular media streams.

TABLE I
QoE Measurements

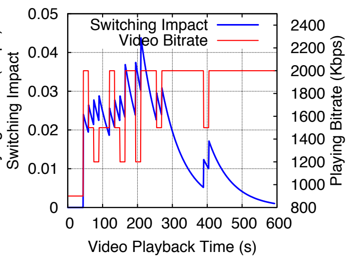
Metric	Average Measurement
Weighted average bitrate	2152 Kbps
Minimum bitrate	50 Kbps
Video bitrate changes	23.52
Minimum Video resolution	240p

Figure 5(a) shows the streaming video quality (VQ) of the reference video with options to switch between different video bitrates in 720p video resolution. The figure demonstrates the non-linear mapping between the video bitrate and the video quality. For example, a switch between two very high bitrates shows less impact on the video quality compared with the same amount of bitrate changes between representations of lower bitrates. Such QoE measurements are valuable for both single-stream quality optimisation and QoE fairness between media streams. Switching impact accounts for the frequency

[1]http://github.com/broadbent/scootplayer

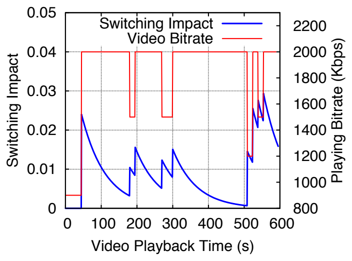


(a) Video Quality

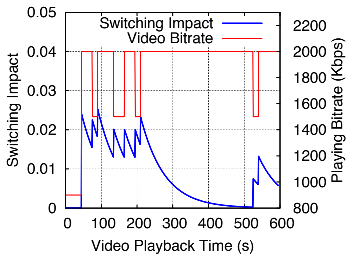


(b) Switching Impact

Fig. 5. Video Quality and Switching Impact of video streaming from a server in Vienna by a video client in Amsterdam.



(a) Cache 1, $\sum SI = 5.35$



(b) Cache 2, $\sum SI = 5.12$

Fig. 6. Switching Impact for playing the same video from caches in different sites

and distribution of changes over the playback time. As is demonstrated in Figure 5(b), high switching impact can be caused by high video quality variation or small, but temporally close, changes.

C. IQMF Evaluation for VoD Caching and Distribution

Following up the VoD caching optimisation use case presented in Section II, we carried out another experiment to demonstrate how the QoE measurements that IQMF provides can be used as input for VoD content placement and distribution. In particular, we wanted to evaluate how a third party caching service can use IQMF's measurements as input for the selection of the most appropriate geographically distributed cache. To enable the experiment, OpenCache [5], a caching service, has been deployed on the GOFF testbed. OpenCache makes use of OpenFlow to provide transparent, flexible and highly configurable video content caching. It controls cache nodes distributed over the network and communicates with the OpenFlow controller to redirect and modify the traffic flows to the most suitable cache. Scootplayer was instructed to request a video and OpenCache worked to transparently redirect the client's requests to a cache (e.g. in Vienna or London).

IQMF measurement results (Figure 6(a) and Figure 6(b)) show that the streaming session from the two different caches encountered almost the same number of oscillations in the video bitrate. With the QoE metrics provided by IQMF, we are able to quantitatively measure the accumulative switching impact (Cache 1: $\sum SI = 5.35$, Cache 2: $\sum SI = 5.12$). Hence, with this input OpenCache could potentially adjust its behaviour and chose the appropriate cache based on the QoE of the end-user. In practice, such measurements on either probing traffic or production services provide insights into how network traffic can be managed for better user QoE.

VI. Conclusion

This paper proposed IQMF; a new in-network measurement framework that provides transparent QoE monitoring as a service for HAS video streams. Without any user involvement or application level configurations, IQMF offers the invaluable quality monitoring closed control loop and novel QoE metrics for QoE-aware media service assessment and optimisation. IQMF leverages SDN to provide in-network functionalities so that crucial service quality metrics can be measured, exchanged, stored and accessed efficiently. These measurements can assist content distribution services or network management to improve the overall experience of end-users. The framework has also been implemented and deployed in a large-scale SDN testbed to demonstrate its QoE measurement capabilities. Using IQMF as a foundation, additional QoE metrics and real-world deployment will be part of our future work.

VII. Acknowledgments

This work has been funded by the EU FP7 GN3plus project (FP7-INFRASTRUCTURES-605243).

References

[1] Cisco Visual Networking Index: Forecast and Methodology, 2013-2018. Technical report, CISCO, June 2014.
[2] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the Impact of Video Quality on User Engagement. In *Proc. of ACM SIGCOMM*, 2011.
[3] ISO-IEC 23009-1:2012 Information Technology. Dynamic Adaptive Streaming over HTTP (DASH).
[4] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race. Towards Network-wide QoE Fairness Using Openflow-assisted Adaptive Video Streaming. In *Proc. of ACM SIGCOMM, Workshop on Future Human-centric Multimedia Networking (FhMN)*, 2013.
[5] P. Georgopoulos, M. Broadbent, B. Plattner, and N. Race. Cache as a Service: Leveraging SDN to Efficiently and Transparently Support Video-on-Demand on the Last Mile. In *Proc. of IEEE ICCCN*, 2014.
[6] E. Nygren, R. K. Sitaraman, and J. Sun. The Akamai Network: A Platform for High-performance Internet Applications. *SIGOPS Oper. Syst. Rev.*, 44, August 2010.
[7] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. In *Proc. of ACM SIGCOMM IMC*, 2007.
[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *ACM CCR*, 38, Mar 2008.
[9] ISO/IEC JTC1/SC29/WG11 (MPEG) Report. Technical report, ISO, November 2013.
[10] M. Venkataraman and M. Chatterjee. Inferring Video QoE in Real Time. *IEEE Network*, 25, 2011.
[11] R. Schatz, T. Hossfeld, and P. Casas. Passive youtube qoe monitoring for isps. In *Proc. of IMIS*, 2012.
[12] M. Eckert, T. M. Knoll, and F. Schlegel. Advanced MOS calculation for Network Based QoE Estimation of TCP Streamed Video Services. In *Proc. of ICSPCS*, 2013.
[13] Y. Liu, S. Dey, D. Gillies, F. Ulupinar, and M. Luby. User Experience Modeling for DASH Video. In *Proc. of IEEE Packet Video Workshop*, 2013.
[14] M.-N. Garcia, F. De Simone, S. Tavakoli, N. Staelens, S. Egger, K. Brunnstrm, and A. Raake. Quality of experience and http adaptive streaming: A review of subjective studies. In *Proc. of QoMEX*, 2014.
[15] V. Seferidis, M. Ghanbari, and DE. Pearson. Forgiveness effect in subjective assessment of packet video. *Electronics Letters*, 28(21), 1992.
[16] DS Hands. Temporal characterisation of forgiveness effect. *Electronics Letters*, 37(12), 2001.
[17] T. Liu, Y. Wang, J.M. Boyce, H. Yang, and Z. Wu. A novel video quality metric for low bit-rate video considering both coding and packet-loss artifacts. *IEEE Journal of Selected Topics in Signal Processing*, 3, 2009.