On-the-fly QoE-Aware Transcoding in the Mobile Edge

Sunny Dutta[1], Tarik Taleb [1], Pantelis A. Frangoudis [2], and Adlen Ksentini [3]
[1] Aalto University, Espoo, Finland
[2] IRISA, Rennes, France
[3] Eurecom Institute, Nice, France
Emails: {firstname.lastname}@aalto.fi; pantelis.frangoudis@irisa.fr; adlen.ksentini@eurecom.fi}

***Abstract* – To enhance video streaming experience for mobile users, we propose an approach towards Quality-of-Experience (QoE) aware on-the-fly transcoding. The proposed approach relies on the concept of Mobile Edge Computing (MEC) as a key enabler in enhancing service quality. Our scheme involves an autonomic creation of a transcoding service as a Virtual Network Function (VNF) and ensures dynamic rate switching of the streamed video to maintain the desirable quality. This edge-assistive transcoding and adaptive streaming results in reduced computational loads and reduced core network traffic. The proposed solution represents a complete miniature content delivery network infrastructure on the edge, ensuring reduced latency and better quality of experience.**

I. INTRODUCTION

The telecom world has experienced a rapid development in the mobile communication technology over the last few years. With high-end devices (e.g., smartphones and tablets) and new interactive mobile applications in place, mobile data usage is on the rise. Services such as video, music, and social networking are gaining momentum. According to the Cisco Visual Networking Index [1], it is estimated that by 2020, 75% of the global mobile data traffic will be occupied by video streaming services. In the recent years, user demand has also shifted from traditional broadcasted video to dynamic on-demand live streaming and to mobile video viewing. As these services are becoming an integral part of the mobile users' entertainment and social life, user expectations towards high Quality of Experience (QoE) are also increasing. The soaring demands for video services with guaranteed quality are now becoming challenging for the service providers. High data rate connectivity to ever-growing data volumes and provisioning a reliable and scalable network to ensure better Quality-of-Service (QoS) are now the prime focus of content providers as well as of mobile network operators.

Traditional video streaming was designed considering a stable Internet link and a limited type of end-user devices. But with the typical link bandwidth variations in the present mobile networks, along with an increase in diverse types of end-user devices, the traditional technology has started to fall behind. This resulted in poor video viewing experience due to its non-scalable and maladaptive nature.

At the media encoding level, to address the scalability support for a diverse range of devices, the Scalable Video Coding (SVC) mechanism [2] is introduced. Media profiles with multiple subsets (e.g., comprising various formats, screen resolutions, and frame rates) are stacked as layers in a single media file. The *base* layer is necessary to decode the video with minimum quality, while *enhancement* layers add spatial and temporal information to increase the delivered video quality.

SVC can be combined with adaptive bitrate streaming techniques (i.e., MPEG DASH and Apple HLS) to adapt to varying network bandwidth. Media files are segmented in chunks, and a manifest file with information of the available profiles/video representations is made available to the end users. The latter can then decide on the desired representation to retrieve media, based on the profiles the end-user device supports. It can then dynamically switch among the available video bitrates (i.e., qualities) to better cope with network dynamics, aiming to achieve continuous playout. For example, a client can monitor the download time of chunks and if the required download time fails to meet the desired time, the client device can automatically switch to a lower profile. Maintaining a huge number of profiles remains a storage challenge for adaptive video delivery systems, and SVC technologies can assist to this end.

Numerous advances have recently also taken place in the content delivery infrastructure front. In a present day scenario, thousands of video content items are uploaded daily to the content provider's network. This content is stored in large volumes in the provider's centralized content database, and is then transcoded from source format to final delivery format. After this computationally intensive transcoding process, the prepared content is then transferred to multiple streaming servers (residing at the edge of the content provider's network) for further delivery to the users. To this extent, cloud computing has played a key role. The on-demand provisioning and auto-scaling features of the cloud supply service providers with scalable resources, such as huge computing power and storage, and facilitate service and infrastructure management.

Unlike some typical enterprise cloud applications, video services are highly time-sensitive. Therefore, apart from storage and transcoding functionality, they also require the delivery of content in real-time. But with the present 4G and cloud infrastructure support, users still experience buffering delays and intermittent playback interruptions. This is mainly due to the congestion that appears more often in the core network, which leads to higher end-to-end latency and jitter. Thus, extending cloud-based services to mobile environments requires a few more factors to be considered for quality enhancement, such as wireless link dynamics, user mobility, latency, and core network traffic.

The next generation of mobile systems, commercially known as 5G, aims at addressing these issues. Relying on technologies such as Network Function Virtualization (NFV), Software Defined Networking (SDN), and Mobile Edge Computing (MEC), 5G promises to attain system flexibility, elasticity and agility. The reformation of the cloud hierarchy towards a decentralized architecture and pushing computing resources close

to mobile users (i.e., at the mobile network edge) is expected to provide effective solutions to these limitations. The concept behind MEC is to provide storage and computation resources from the network edge, in the proximity of users. Accordingly, pushing data processing from remote cloud locations to the edge and processing data "locally" can reduce traffic bottlenecks in the core network. Besides, it can help in offloading important computational load from power-constrained user equipment to the edge while ensuring short end-to-end latency.

There are high expectations on how MEC can empower video streaming services. In this paper, we do not go deep in optimizing transcoding techniques but, rather, we focus on using transcoding as an enabling service at the mobile edge to enhance video quality. This paper proposes a scheme to achieve fine granularity in bitrate selection for adaptive streaming based on real-time perceived video quality monitoring, using QoE estimation techniques. Degradation in perceived quality triggers an on-the-fly transcoding (OTFT) service at the mobile edge to vary the bitrate of the stream, a process which continues until optimal quality is achieved. Our objective is to demonstrate how the QoE of a video service can be maintained by enforcing transcoding as a requirement-based Virtual Network Function (VNF). Additionally, our scheme reflects the reduction in queuing delay in transcoding and performing quality assisted video streaming.

The remainder of this paper is organized as follows. Section II presents some related work. Section III describes our proposed OTFT framework, along with its supporting mechanisms. For the sake of performance evaluation, Section IV portrays our experimental setup and discusses the obtained results. We conclude the paper by summarizing our main findings in Section V.

II. Related Work

MEC has been proposed as an enabler for novel, low-latency services in a mobile network. Considering its potential, both industry and the research community are working on maximizing the benefit and efficiency of the MEC technology. As discussed in [3], [4], such decentralized architecture with support for fault resiliency will enable new services and promising business models. Importantly, MEC has been considered to be the key player in mobile video applications.

In the context of video streaming, the proposed two-hop edge architecture in [5] reflects the data transfer rate and throughput of the edge in comparison with the remote cloud. The research work in [6] introduces a network-assisted adaptive streaming application to enhance QoE of the delivered multimedia content. An architecture with distributed parallel edges to increase QoE for content delivery has been proposed by Zhu et al. in [7]. Chang et al. [4] deploy independent small-scale datacenters at the network edges, which are capable of performing video caching and streaming on their own. Jararweh et al. [8] integrate caching with proxy functionality at the edge to store media content. They also enforce computation offloading to increase the lifetime of mobile devices.

Video transcoding in the cloud has recently received significant research attention. Utilizing virtual instances of the cloud to perform video transcoding upon request has been proposed in [9], [10] as the most simple and straightforward use case. The work in [11] and [12] propose cloud-assisted video transcoding. Utilization of cloud resources to assist mobile devices for customized transcoding services [13] and for energy conservation on mobile devices [14] has also been proposed. As an efficient way of video transcoding in the cloud, an approach to reduce the bitrate of the transcoded video by using a higher quantization parameter without reducing the frame size or the frame rate has been proposed in [15]. Transcoding only portion of a video to reduce the transcoding time [16], [17] and distributed video transcoding in the cloud to enhance efficiency have also been studied in [18], [19]. Amazon in its recent development [20] has introduced an elastic transcoder to reduce time of the intensive transcoding service.

In all the above research works, MEC is deemed to be a promising solution for handling video services, although mainly focusing on streaming, caching and compression techniques; the computationally intensive transcoding functionality has been generally proposed to be treated on the cloud-based infrastructure. Energy-efficient video transcoding as a network function at the edge has been proposed in [21] for a Voice over Long Term Evolution (VoLTE) service.

In our prior work [22], we focused on the effects of processing load on user experience, and proposed a QoE-driven mechanism for elastic *compute* resource allocation in a cloud-native 5G environment. In this work, we address issues that pertain to *network*-related resources and effects. To the best of the authors' knowledge, QoE-aware on-the-fly transcoding along-with bitrate variation in adaptive streaming at the mobile edge for content delivery has not been yet considered. In this paper, we describe and showcase an innovative transcoding and streaming scenario leveraging the potential of MEC.

III. Proposed OTFT Scheme

A. Use cases

In a real-life scenario, many use cases can be considered involving video streaming from the edge. We here consider two use cases where fine-grained on-the-fly transcoding will enhance users' video experience:

- Use case 1: Bob and Alice are two friends residing in the same city in the vicinity of the same mobile edge node. Bob has one high-end smart phone capable of taking High Definition (HD) videos whereas Alice's device supports only Standard Definition (SD) quality. We consider a situation where Bob takes a HD video of some funny moments and wants to immediately share it with Alice. He uploads the video to the nearest edge tagging Alice. Alice, upon receiving the notification, clicks to play the video. Due to unsupported resolution, Alice will experience buffering when playing the content. The edge, being smarter in this case, will make use of its on-the-fly transcoding capabilities to convert the high-bitrate source format to a lower bitrate one. This will help Alice to view the video more smoothly. The relevant operations, including uploading, transcoding, and streaming, will take

place in the edge without involving the content provider's backend cloud network.

- Use case 2: Bob is waiting for his next transit flight in a busy airport and wants to watch some music video for leisure. He connects to the nearby edge and starts receiving the video stream. The edge server has adaptive streaming enabled with contents of only HD quality. However, due to limited network bandwidth and the unavailability of other low-bitrate profiles of the media content, Bob will experience buffering delays and, thus, reduced QoE. In such a situation, the edge will spin up the transcoding service to create media content with a lower bitrate than HD. Upon completion, the newly prepared content will be streamed. Bob's QoE will be monitored, and this process of fine tuning will continue in a recurring manner, until a smooth viewing experience is achieved. This will both balance user experience and improve network resource utilization and, thus, network availability.

In this paper, we consider a lightweight, container-based transcoding service to meet the requirements of the aforementioned use cases, with more focus on fine bitrate granularity and the on-the-fly transcoding aspect of Use case 2.

B. Proposed OTFT Architecture

The proposed OTFT architecture, represented in Fig. 1, is based on a two-tier principle. The content provider's cloud-based network consists of a centralized content database (CCD), a transcoder, and a segmenter. The cloud has its own orchestrator to manage its infrastructure and resources. The uploaded content from the content producers are initially stored in the CCD. The content is then sent to the transcoder to perform H.264/AVC transcoding. Note that the use of SVC technologies is an option that can offer storage advantages, since all available video representations can be stored in a single file. However, our design considers H.264/AVC encoding, which is also more widely supported. The prepared content is finally segmented in chunks and is prepared for adaptive streaming by the segmenter. Streaming-ready contents are then transferred to the streaming server (SS). The SS is hosted in the MNO's edge network node and is responsible to further deliver the video stream to the end user's device. An additional component, the cloud controller (CC), is considered. The CC is responsible for business-related functionality and maintains the Service Level Agreement (SLA) between the content provider and the mobile network operator (MNO). This deals with the content provider's access rights over the MNO's network to manage the SS. The dotted line between CC and the Edge Orchestrator (EO) represents agreement level connectivity.

The Edge Node (EN) is hosted on virtual machines on top of existing hardware in the MNO's edge network. The EN runs its own compute and storage services. The compute one is responsible for hosting container-based applications on the edge, and the storage one is used to host the container image templates. The EN and all its services are managed by the EO. These services are hosted in containers inside the edge, and the EO controls their deployment and management.

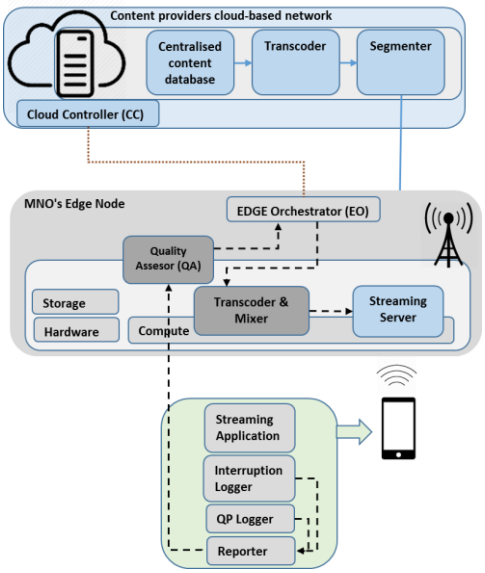

Fig. 1. Proposed OTFT Architecture.

The primary service components considered here are:

a. **Streaming Server:** Part of the content provider's network, and a container-based application to perform adaptive streaming to the client. In this scenario, we have considered HTTP live streaming (HLS), so the pre-transcoded chunks of media segments (sent from the segmenter) reside inside this server. Upon a client request, the manifest file with the media description and structure is first served, and sequentially the chunks are delivered.

b. **Quality Assessor:** It is responsible for assessing the quality of the served video using the Pseudo-Subjective Quality Assessment (PSQA) methodology. PSQA uses machine learning techniques to train a Random Neural Network (RNN) classifier on data from subjective tests with human subjects, where specific parameters that affect QoE are monitored while the viewer assesses the quality of test video sequences in the scale from 1 (poor quality) to 5 excellent quality). The resulting RNN can then be used in real time, given that its input parameters are measurable. In our case, we used the PSQA tool of Singh et al. [23], which is trained to estimate QoE for an adaptive streaming service of H.264/AVC-encoded video. The input to this tool is the number, frequency and duration of playout interruptions in a 16-second video window, as well as the average value of the Quantization Parameter (QP) (the input values have to be appropriately normalized; for details see [23]). Its output is an estimate of the Mean Opinion Score (MOS), i.e., the expected quality rating in the 1-5 scale that a panel of humans would give for a video under the specific QP and interruption conditions.

The Quality Assessor receives real-time information of the service status from the client. The status includes the ID of the last downloaded segment and of the playing segment, the playback interruption count and duration, and the QP value of the video playing at the end-user

device. Considering these parameters, the PSQA model generates a QoE estimate in real-time in an automatic manner and without any human intervention. It also performs a check on the calculated MOS value. It is under the QA's scope to trigger the EO for initiating transcoding service.

c. **Transcoder and mixer:** It is a container-based virtual transcoding service deployed by the EO. It is used to transcode the source media format to another deliverable format with a different bitrate. Once the newly prepared content is ready, the mixer replaces the old content in the streaming server with the new one. The triggering process to deploy the transcoder and mixer service is assisted by the EO. Once the required operation is done and the target QoE level is achieved, the service is terminated and removed from the EN.

Referring to Use case 2, the stepwise operational flow is depicted in Fig. 2. For the purpose of demonstration, let consider that the streaming server stores HLS content with bitrate of R1 (high bitrate). Also, we consider here that the application used to view the stream can provide the information necessary for QoE estimation, i.e., the ability to measure playout interruption and QP information and the reporter functionality. The interruption logger maintains a log of occurred interruptions count and interruption duration of the played video segment. Similarly, the QP logger tracks the QP value per picture macroblock of the played video. The log values are finally handed over to the reporter in real-time. The reporter also keeps information of the downloaded segment ID, playing segment ID, and playout start time.

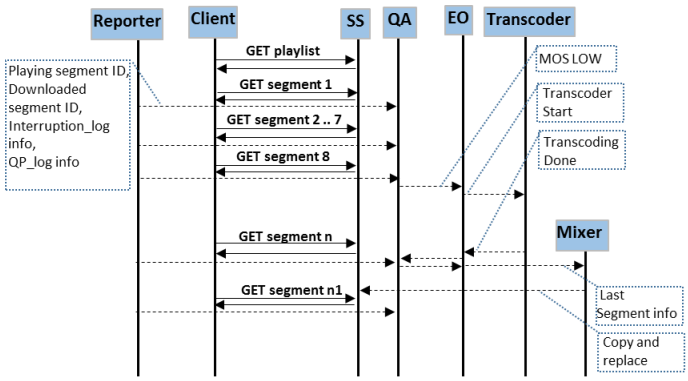

Fig. 2. Signaling diagram.

To start the service, Bob connects to the edge and sends an HTTP GET request to the server to fetch the playlist/manifest file. Upon receiving the file, the end-user device starts sending HTTP GET requests to fetch the media segments sequentially one by one as detailed in the playlist/manifest. Once the initial necessary playout buffer level is achieved the player starts displaying the first segment and automatically moves on to the second one, as soon as the first segment is over. At the end of every segment, the reporter sends the whole set of information to the QA. The QA performs a real-time calculation of the MOS value at fixed intervals. If the desired value is above the optimal, it reports that the QoE of the video is acceptable and no further action is required until the next calculation time instance. On the other hand, if the value decreases, the QA triggers the EO to spin up the transcoder and mixer service.

The container is started (using template image from the EN) and the transcoding of the media content is performed. The bitrate is reduced to R2 (R2<R1), one step lower than the current bitrate. Once the newly-built media content is ready, the EO performs a check to get information on the last downloaded segment ID. Upon confirmation, the EO then triggers the mixer to replace the old segments (which are not yet fetched by the client) with the new ones. Once the operation is finished, the EO waits for the next QA information. If no information is received within a certain interval, the EO considers that the target MOS is achieved, and discards the service and stops the container. However, if the QoE still remains low, the same operation is performed again reducing the bitrate to R3 (R3<R2). This stepwise recurring operation continues until a target optimal QoE is achieved at the client end.

It is worth noting that the proposed solution not only ensures QoE, but also transforms the edge into a complete video delivery solution. Performing transcoding on-the-fly incurs compute-intensive load only for a limited time. Furthermore, storage overhead can be reduced, as pre-transcoded multiple versions are not required from the beginning and can be generated on-demand. Moreover, by serving the content locally (i.e., from the edge), the solution ensures reduction in core network traffic and reduced end-to-end latency.

C. Potential enhancements

Considering a mobile network, where network conditions are dynamically changing due to user mobility, high sensitivity towards transient events may lead to a "ping pong" effect, where transcoding will be initiated every now and then with the varying conditions of the network. Instead, the triggering should be done only if the network conditions have actually degraded. However, still the tradeoff between responsiveness and avoiding the ping pong effect exists. Maintaining a sliding window of QoE scores to decide on whether to initiate transcoding or not based on a running average of the MOS (within that window) can help address this issue.

Various further performance enhancements are also possible. For example, the cross-layer mobility, bandwidth [24], and QoE [25] prediction mechanisms that we have studied in our prior work in similar contexts can be applied to this end. Such an ability to predict the conditions at the client end can assist in identifying the optimal time to initiate transcoding.

Depending on the resource availability in the EN, the transcoding service can be initiated in parallel to support multiple end-user requests. Considering it as a compute-intensive task, and in case of limited resources in the serving Edge, it is the EO's responsibility to select another nearby edge (taking into account response time and resource availability) to perform the transcoding-only operation, leveraging the shared infrastructure concept of MEC.

The concept can be further enhanced by introducing smart algorithms in the selection of the steps for bitrate variation. Also, introducing advanced transcoding techniques may reduce

the delay incurred from triggering till completion. Moreover, if the video is almost towards the end (i.e., the remaining video time is less than the transcoding time), the service initiation may simply be omitted.

IV. Performance Evaluation

Fig. 3 portrays the testbed environment which we have built to simulate the proposed edge-based transcoding and streaming service. The content provider's cloud-based network is not simulated, as we have considered that the media content for adaptive streaming is ready and is already stored in the streaming server. The simulation environment is mostly focused towards assessing the performance of the edge. Our testbed is simulated with two laptops (with Ubuntu 14.04.3 LTS desktop OS), where one is the edge node and the other is the client.

The client was simulated using a version of the VLC player which we modified to implement QP and interruption monitoring. The reporter functionality was implemented in a python script, which retrieves the required information, filters it, and passes it on to the QA database residing in the EN.

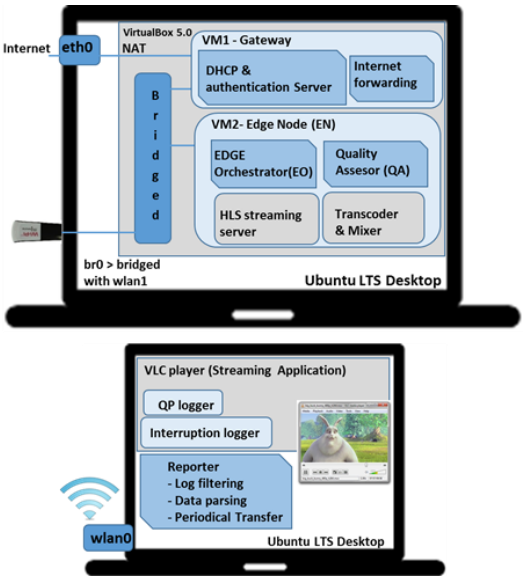

Fig. 3. Test-bed setup.

On the edge side, two VirtualBox VMs were used. VM1 was used as a gateway for the entire network to access the Internet. DHCP with authentication was also set up inside VM1 to configure the whole network using a single subnet (for ease of the simulation). VM2 was configured using the Proxmox Virtual Environment to act as the EN. We use OpenVZ containers to host the services of our architecture in our testbed. The streaming server functionality was achieved using an Ubuntu cloud minimal image using Nginx as the webserver inside an Openvz container. Nginx was also configured for HLS streaming. The content for initial streaming was pre-transcoded and prepared using ffmpeg, and the video codec used was H.264/AVC. The same container is used as storage for the media files. The QA was configured to receive interruption and QP information periodically from the database and appropriately normalize/transform them to be used as input to the PSQA tool to calculate the expected MOS values. The same QA script was responsible for the evaluation of the MOS values and triggering the EO. The automated orchestration was performed with a script serving the functionality of EO for spinning up the container which provides the transcoding service. The transcoding container template was built with an Ubuntu minimal image and had ffmpeg installed. This service was configured to start on boot. The mixer functionality was created with a script residing inside the same container. Finally, to ensure that the laptop acted as an edge access point, its wireless LAN interface was configured using hostapd in IEEE.802.11 master mode. Also, Netem and Wondershaper tools were used to simulate a cellular environment.

It is worth recalling that the objective of these tests is to advocate the use of MEC to ensure on-the-fly transcoding and achieve results on its responsiveness. Responsiveness is the measured delay from the time of triggering the service to the actual QoE enhancement time. This responsiveness check was performed in two ways:

a. The container is already active with pre-transcoded media files. Only the mixer functionality is used, and, thus, taken into account in the delay measurement.
b. Using the full functionality of on-the-fly transcoding by booting a container, initiating the service, transcoding the media file and then performing the mixing.

The media file used for this test purpose was 9 minute 56 seconds long with 298 segments in total, each comprising of 2 s of video. The reporter information was sent exactly after 2 s of video has been played (considering video play-time). To perform QA operation, the tool requires the information of 16 s of played video. Therefore, the MOS calculation was performed only after information of a total of 8 segments were received. The MOS values are represented in a scale of 1 to 5. The value below 3.5 was considered low and was used to trigger the EO.

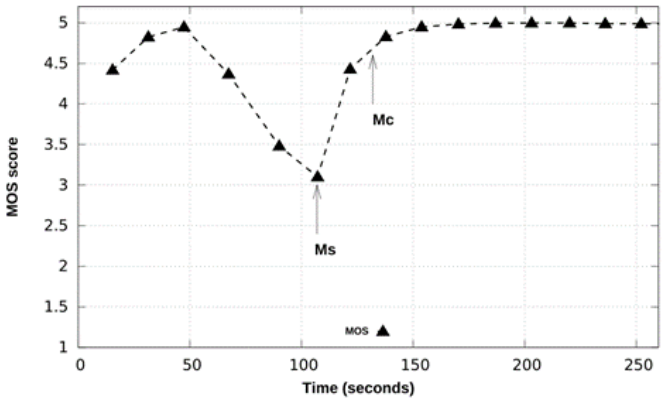

Fig. 4. MOS vs Time (with pre-transcoded media files). (Ms: mixing start time; Mc: mixing completion time).

Case a. is represented in Fig. 4, where initially the MOS value was high. With time, it degraded and as soon it reached below the predefined threshold, mixing was initiated. In this scenario, pre-transcoded low bitrate files were copied to the streaming server's desired location and replaced the existing ones. To save

time, only the segments which were yet to be downloaded by the client were replaced. The replacement occurs in a sequential manner. As a result, although the mixing time was approximately 26 s, the QoE started enhancing as soon as few segments were delivered.

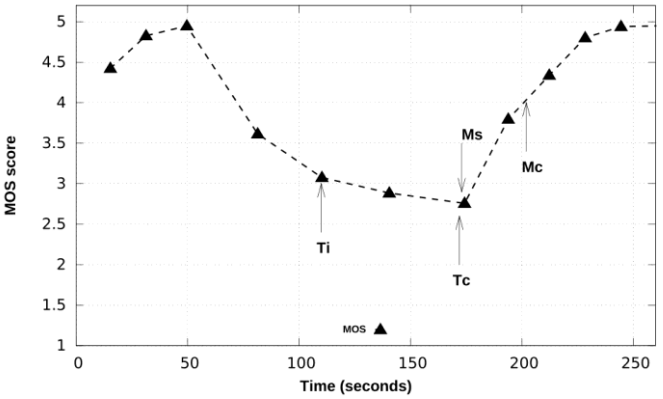

Fig. 5. MOS vs Time (full functionality). (Ti: transcoding container initiated; Tc: transcoding completed; Ms: mixing started; Mc: mixing completed).

Fig.5 depicts the full functionality as mentioned in case b. 'Ti' represents the time when container initiation started. The time difference between the QA triggering the EO and the EO starting the creation of the container is in the milliseconds range. The container boots up with the already prepared transcoder image template (within 3 secs). Once it is ready, the EO signals to start the transcoding with the mentioned rate. Having limited resources (2 vCPU & 1024MB RAM) the container performs this total operation in approximately 1 minute 10 seconds. The mixer operation starts as soon as the transcoding is over. In-between, the EO sends the information of the last downloaded file to indicate from which segment the mixing will start. The mixer operation takes almost similar time as mentioned before (in case a.) and depends on the number of segments to be transferred and replaced.

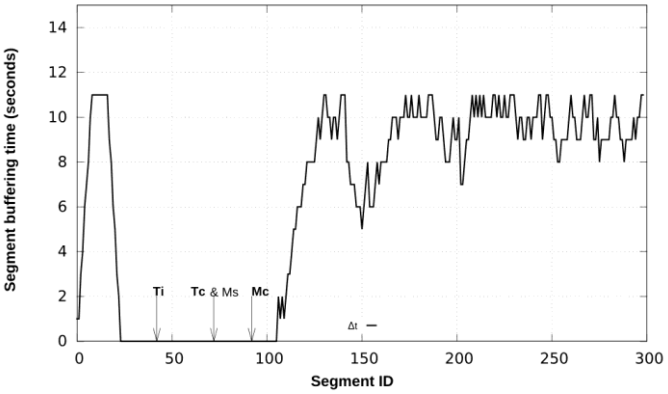

Fig. 6. Segment buffering time (Δt) *vs Segment ID. (Ti: transcoding container initiated; Tc: transcoding completed; Ms: mixing started; Mc: mixing completed; when Segment buffering time = '0', corresponding segments are not downloaded in advance).*

In Fig. 6, the segment buffering time (Δt) is the time a downloaded segment spends in the buffer waiting for playout. In other words, it is the time difference between the instant its playout started and the instant it was fully downloaded. It is clear that initially the segment buffering time was high, which indicates that the segments were downloaded in advance. As a result, the immediate next few segments were already downloaded and ready before it was being played. Therefore, there was no buffering delay, and thus no playout interruptions, hence the MOS was high (if compared with Fig. 5). When we introduced a degradation in network conditions, emulating congestion due to significant background traffic, the downloading segment time increased because of a reduction in the available bandwidth. At a point when the difference (Δt) was almost zero, the video experienced buffering delays as it had to wait till the downloading of a segment is complete. However, after transcoding to a lower-bitrate video, the segment size, as well as the segment download time, reduced. Consequently, adapting video bandwidth demands to the current network traffic conditions lead to timely video segment downloads, and minimized playback buffering time. This positively impacted the overall MOS.

To this extent the PSQA tool helped in somehow avoiding that ping pong effect of the mobile network, as it outputs one "average" value every 16s. Moreover, the simulation was performed using only two types of media files. Furthermore, finer granularity can be achieved by properly setting the QoE threshold, by increasing the intelligence of the EO in terms of variety of bitrates and finally by advanced lightweight transcoding techniques with less startup latency.

V. CONCLUSION

In this paper, we proposed a scheme that reflects QoE in deciding, in an autonomic manner, when to enforce transcoding in an edge environment to increase the service quality. The proposed scheme features a cognitive way in selecting the best suitable media profile by utilizing the concept of on-the-fly-transcoding, as one of the future applications of MEC. Instead of accepting the pre-selected bitrate of the streamed video from the content provider's end, this framework enforces the edge to customize the content based on the user's expectation. The framework was validated using a real life testbed, and interesting results were obtained on response times. Based on the obtained result, it can be concluded that MEC awaits a lightweight transcoding functionality, which can convert this proof-of-concept to reality. This defines one of the authors' future research directions in this area.

Acknowledgment

This work was partially supported by the TAKE 5 project funded by the Finnish Funding Agency for Technology and Innovation (TEKES) and in part by the Finnish Ministry of Employment and the Economy.

REFERENCES

[1] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020," [Online] http://www.cisco.com/c/en/us/solutions/collateral/service-provider/vis

ual-networking-index-vni/ mobile-white-paper-c11-520862.pdf, February 2016.
[2] H. Schwarz, D. Marpe and T. Wiegand, “Overview of the Scalable Video Coding Extension of the H.264/AVC Standard,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 17, No. 9, Sept. 2007, pp. 1103-1120.
[3] European Telecommunications Standards Institute (ETSI), “Executive Briefing – Mobile Edge Computing (MEC) Initiative – Issue 1”, [Online] https://portal.etsi.org/portals/0/tbpages/mec/docs/mec%20ex-ecutive%20brief%20v1%2028-09-14.pdf, Sep. 2014.
[4] H. Chang, A. Hari, S. Mukherjee, and T. Lakshman,"Bringing the cloud to the edge," in Proc. IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, May 2014.
[5] D. Fesehaye, Y. Gao, K. Nahrstedt, and G. Wang, “Impact of Cloudlets on Interactive Mobile Cloud Applications,” in Proc. IEEE 16th Int’l Conf. on Enterprise Distributed Object Computing Conference (EDOC), Beijing, China, Sep. 2012.
[6] J. Fajardo, I. Taboada, and F. Liberal, “Improving content delivery ef-ficiency through multi-layer mobile edge adaptation”, in IEEE Net-work Mag., Vol. 29, No. 6, Dec. 2015, pp. 40-46.
[7] W. Zhu, C. Luo, J. Wang, and S. Li, “Multimedia Cloud Compu-ting ,“ in IEEE Signal Processing Mag., Vol. 28 , No. 3, May 2011, pp. 59-69.
[8] Y. Jararweh, L.Tawalbeh, F. Ababneh, and F.Dosari, “Resource Efficient Mobile Computing Using Cloudlet Infrastructure,” in Proc. IEEE 9th Int’l. Conf. on Mobile Ad-hoc and Sensor Networks (MSN), Dalian, China, Dec. 2013.
[9] F. Wang, J. Liu, and M. Chen, “CALMS: Cloud-Assisted Live Media Streaming for Globalized Demands with Time/Region Diversities,” in Proc. IEEE Conf. on Computer Communications (INFOCOM), Orlando, Florida, Mar. 2012
[10] Y. Zhao, H. Jiang, K. Zhou, Z. Huang, and P. Huang, “Meeting Service Level Agreement Cost-Effectively for Video-on-Demand Applications in the Cloud,” in Proc. of IEEE Conf. on Computer Communications (INFOCOM), Toronto, Canada, May 2014.
[11] R. Cheng, W. Wu, Y. Lou, and Y. Chen, “A Cloud-Based Transcoding Framework for Real-Time Mobile Video Conferencing System,” in Proc. of 2nd IEEE Intl. Conf.on Mobile Cloud Computing, Services, and Engineering (MobileCloud), London, UK, April 2014.
[12] Y. Wu, C. Wu, B. Li, and F. C. Lau, “vSkyConf: Cloud-assisted Multiparty Mobile Video Conferencing,” in Proc. of the 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing, Hong Kong, China, Aug. 2013.
[13] M. Chen, “AMVSC: A Framework of Adaptive Mobile Video Streaming in the Cloud,” in Proc. of IEEE Global Communications Conference (GLOBECOM), Anaheim, California, Dec. 2012.
[14] W. Zhang, Y. Wen, and H.-H. Chen, “Toward Transcoding as a Service: Energy-Efficient Offloading Policy for Green Mobile Cloud,” IEEE Network, Vol. 28, No. 6, Nov. 2014, pp. 67–73.
[15] F. Jokhio, T. Deneke, S. Lafond, and J. Lilius, “Bit Rate Reduction Video Transcoding with Distributed Computing,” in Proc. of 20th Intl. Conf. on Parallel, Distributed and Network-Based Processing (PDP), Garching, Germany, Feb. 2012.
[16] F. Lao, X. Zhang, and Z. Guo, “Parallelizing Video Transcoding Using Map-Reduce-Based Cloud Computing,” in Proc. of IEEE Intl. Symposium on Circuits and Systems (ISCAS), Seoul, Korea, May 2012.
[17] G. Gao, W. Zhang, Y. Wen, Z. Wang, W. Zhu, and Y. P. Tan, “Cost Optimal Video Transcoding in Media Cloud: Insights from User Viewing Pattern,” in Proc. of IEEE Intl. Conf. on Multimedia and Expo (ICME), Chengdu, China, July 2014.
[18] A. Heikkinen, J. Sarvanko, M. Rautiainen, and M. Ylianttila, “Distributed Multimedia Content Analysis with MapReduce,” in Proc. of 24th IEEE Intl. Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), London, UK, Sept. 2013.
[19] M. Kim, S. Han, Y. Cui, H. Lee, H. Cho, and S. Hwang, “CloudDMSS: Robust Hadoop-Based Multimedia Streaming Service Architecture for a Cloud Computing Environment,” Cluster Computing, Vol. 17, No. 3, Sept. 2014, pp. 605–628.
[20] Amazon, “Amazon Elastic Transcoder,” [Online] https://aws.amazon.com/elastictranscoder/
[21] M. T. Beck, S. Feld, A. Fichtner, C. L. Popien and T. Schimper, “ME-VoLTE: Network functions for energy-efficient video transcoding at the mobile edge ,” in Proc. of 18th Intl. Conf. on Intelligence in Next Generation Networks (ICIN), Paris, France, Feb. 2015.
[22] S. Dutta, T. Taleb, and A. Ksentini, “QoE-aware Elasticity Support in Cloud-Native 5G Systems,” in Proc. IEEE ICC’16, Kuala Lumpur, Malaysia, May 2016.
[23] K. D. Singh, Y. Hadjadj-Aoul, and G. Rubino, “Quality of experience estimation for adaptive HTTP/TCP video streaming using H.264/AVC,” in Proc. IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV,USA, Jan. 2012.
[24] A. Nadembega, A. Hafid, and T. Taleb, “An Integrated Predictive Mobile-Oriented Bandwidth-Reservation Framework to Support Mobile Multimedia Streaming,” IEEE Trans. on Wireless Communications, Vol. 13, No. 12, Dec. 2014, pp. 6863 – 6875.
[25] A. Ksentini, T. Taleb, and K.B. Letaif, "QoE-Based Flow Admission Control in Small Cell Networks," IEEE Trans. Wireless Communications, Vol. 15, No. (4), 2016, pp. 2474-2483.