Efficient transcoding and streaming mechanism in multiple cloud domains

Badr Eddine Mada, Miloud Bagaa and Tarik Taleb
Dept. of Communications and Networking, School of Electrical Engineering
Aalto University, Espoo, Finland
Emails: badr.mada@aalto.fi ; miloud.bagaa@aalto.fi ; tarik.taleb@aalto.fi

***Abstract*—Given the constantly growing demand for live streaming services, live transcoding has become compulsory and very challenging. So far, investigations have been confined to satisfy a huge number of users for ensuring the Quality of Experience (QoE). The aim of this paper is to propose a framework architecture following ESTI-NFV (Network Function Virtualization) model [1], whereby the transcoding and streaming Virtual Network Functions (VNFs) would be running on top of multiple cloud domains. By respecting ESTI-NFV model, we ensure the flexibility of our virtual delivery platform that scales up/down and in/out relative to the changing demands of the end-users in order to reduce cost. For this purpose, this paper presents a new framework for managing the virtual live transcoding and streaming VNFs on top of multiple cloud domains for ensuring the QoE while reducing the cost. In order to develop such a framework, we have done a set of experimental benchmarking of transcoding and streaming VNFs using variant flavors (i.e., in terms of CPU and Memory resources). The obtained results will be explored later for developing an intelligent algorithm that will be integrated with the proposed framework in managing different transcoding and streaming VNFs in an efficient manner.**

I. Introduction

In the last few years, live streaming has become extremely popular [2]–[5], and it is expected to be insensitively used in the near future. Real time entertainment services, such as video and audio streaming services, are currently accounting for more than 60% of the Internet traffic, e.g., in North America's fixed access networks during peak periods [6]. Until a couple of years ago, videos on the Internet were mostly recorded by devices, edited and then uploaded to big platforms such as Youtube and Dailymotion. Thanks to the processing capabilities of today's smart devices, live streaming has become possible for everyone. Nowadays, every major social network from Facebook to Twitter and Instagram allows users to broadcast live videos so that their followers can watch and interact with them in real time. Currently, the customers demand high quality videos and better performance in terms of bandwidth utilization from today's broadband applications. In order to meet subscribers' expectations in terms of high level of quality and performance, there is an obvious need for an online transcoding system for live transcoding and broadcasting.

Recent studies, given by CISCO, confirm that the data traffic utilization has increased more than twice within the last four years (from 2013 to 2017), whereby the online video traffic present more than 70% of the total data traffic in the Internet [7]. The same study expects that the online video traffic will be around 82% of the Internet data traffic by 2020. Every second, nearly a million minutes of video content will cross the network by 2020 [7]. However, transcoding videos is a challenging and time consuming process [8]–[10]. Variant techniques of video transcoding have been reviewed in [11], [12]. In order to improve the Quality of Experience (QoE) of live streaming services, above all in a cost-efficient manner, the following should be taken into account:

- Scalability; the solution must scale up/down automatically to meet the users' demands while optimizing resource utilization.
- Availability; the system should be available to respond to the clients' requests in a timely manner.
- Maintainability or the future proof technology; the system should be easy to upgrade for supporting new types of video resolutions, such as $4k$ with new codecs and formats.

In this paper, we propose a new framework for performing online transcoding and live streaming on top of multiple clouds. Indeed, both online transcoding and live streaming services will run as Virtual Network Functions (VNFs) on top of multiple cloud domains. The use of ETSI-NFV model for offering online transcoding and live streaming services have manifold benefits:

- The Capital Expenditures (CAPEX) would be reduced by eliminating the wasteful over-provisioning of hardwares by supporting pay-as-you-grow model.
- The Operations Expenditure (OPEX) would be reduced as per reducing the disk resource utilization and server maintenance requirements, and as per simplifying the management of network services.
- Agility and flexibility can be gained by scaling up/down and in/out variant VNFs based on the changing demands of the end-users. If there is an overwork on a VNF, the system will, automatically, scale up/out by creating new instance(s) for serving the overload. Otherwise, if the VNF is underloaded, the system scale down/in and stops few instances for reducing the cost.

Regarding the last benefit, in this paper, we have conducted a set of real experiments for benchmarking the capability of each flavor to perform the online transcoding and live streaming services. The obtained results from the experiments would help the orchestrator to make the right decisions, such as scaling up/down and in/out, at the right time.

The remainder of this paper is organized as follows. Section II summarizes the fundamental background topics of this work

978-1-5090-5019-2/17/$31.00 ©2017 IEEE

and discusses some related research work. An overview of the proposed framework along with its main components is presented in Section III. Section IV illustrates the experimental setup and discusses the obtained results. Finally, the paper concludes in Section V.

II. Related work

In this section, we present the literature research work most relevant to our research work. In [13], authors have proposed a hybrid video transcoding framework that enables the transcoding and live streaming of on-demand videos using cloud services by taking into account the Quality of Service (QoS) requirements. Due to the large disk resources taken by pre-transcoded videos, the framework reduces the required disk resources and optimizes the cloud resource cost by transcoding the videos in lazy manner. In [14], C. Wang et al. proposed a solution that receives one video as input and converts it to many qualities with more than 2x complexity reduction. Meanwhile, the authors in [15] have proposed a framework that is based on a load prediction model to dynamically allocate Virtual Machines (VM) in an Infrastructure as a Service (IaaS) environment to transcode variant videos. The work in [16] presents an efficient video coding encoder adaptation scheme that takes into account QoE of all users. X. Li et al. [17] presented a video live streaming framework that uses the cloud service architecture to handle the transcoding of the high video quality, using a scheduling method that meets the QoS demands of the live streaming viewers. Authors in [18] implemented a solution that uses the cloud infrastructure for allocating resources dynamically, taking into account that the users' demands change when streaming variant videos.

Building on research work conducted in the recent literature, the work presented in this paper explores how to maintain online transcoding and live streaming on cloud-based systems, based essentially on the ETSI-NFV model for improving QoE while reducing the incurred cost [19], [20]. We also present a testbed experiment to measure CPU utilization during online transcoding and live streaming of videos in multiple cloud domains.

III. Proposed framework

A. Framework overview

Fig. 1 depicts the architecture envisioned for managing the streaming of a high number of live videos leveraging multiple cloud domains. The proposed architecture enables the live streaming of videos hosted on multiple cloud domains while ensuring the network scalability and guarantying QoE. In the proposed architecture, live adaptive bitrate streaming technique is considered. It enables the streaming of the same video content to users using multiple resolutions according to their network bandwidth and device capabilities [21]. ETSI-NFV [1] defines a reference architecture in order to create standardized solutions and ensure network scalability. The NFV architecture also supports elasticity and flexibility for creating different VNFs, such as virtual streaming and transcoding servers, across multiple domains. As depicted in Fig. 1, the proposed architecture consists of four main components:

1) *Users*: A set of users that would be interested in watching a set of live streaming videos from a cloud network. These users are grouped into different sets according to: i) Videos of interest; ii) Network bandwidth; and iii) Features of their multimedia devices. Indeed, users who are interested in the same video content, and have similar network bandwidth and multimedia devices, would be considered in the same group, and practically will receive the same stream.
2) *Cloud networks*: Each cloud network is managed by a Virtual Infrastructure Manager VIM (e.g., OpenStack) [22]. In order to provide Virtualized Infrastructure Management functionality, the cloud networks run different virtualization technologies (e.g., KVM, XEN or Containers - LXC or Docker) that allow the management of different virtual resources on top of hardware resources (e.g., Computing, Storage and Network). VIM allows the instantiation of different VNF instances with different virtual resources using pre-stored VNF images (e.g. streaming or transcoding). Different resources in a cloud network are defined through a set of flavors, whereby each flavor represents the amount of virtual resources (i.e. number of Virtual cores CPU, memory and storage) that could be dedicated to a specific VNF instance. The cost of a VNF changes according to the size of its flavor.
3) *Streaming and Transcoding VNFs*: The transcoding VNFs enable the transforming of already-compressed (or encoded) content with a specified resolution to another compressed content with another resolution to satisfy the requirements of a group of users in terms of network bandwidth and multimedia device capabilities. Meanwhile, the streaming VNFs enable the live streaming of video contents to different users while satisfying their required QoE.
4) *Global transcoder*: In the proposed architecture, there can be one or multiple global transcoders, distributed over a specific geographical area. A global transcoder may run on top of a physical machine (i.e., dedicated server) or another VNF that has more powerful resources using a bigger flavor.
5) *Orchestrator*: The envisioned orchestrator offers a RESTful API that allows the administrator to specify different management rules and policies for the instantiation and auto-scaling of different VNFs (i.e., streaming or transcoding) instances. According to the policies and the service level agreement (SLA) received from the administrator, the orchestrator enforces these rules by ensuring the communication with the corresponding VIM. The orchestrator would receive requests from different users, then decide on how to manage the live streaming flows between the transcoder and streamer VNFs, from one side, and the streamer and users from the other side.

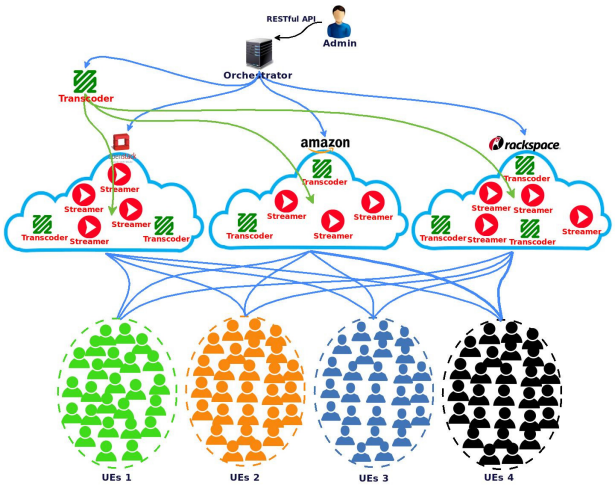


Fig. 1. Main overview of the proposed architecture.

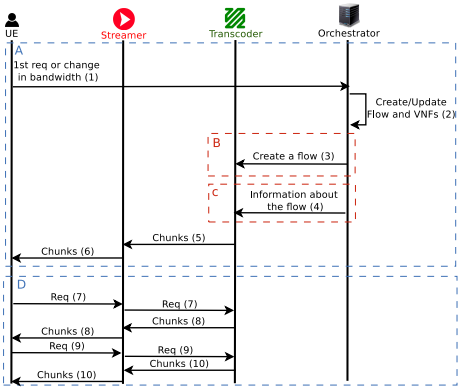


Fig. 2. Sequence diagram of the proposed solution.

B. Framework description

The proposed architecture adopts two protocols, namely the Dynamic Adaptive Streaming over HTTP (DASH) and HTTP Live Streaming (HLS) for serving variant clients [23], [24]. Both protocols ensure that the same video content will be streamed to the end user at a variety of resolutions. Each resolution is subdivided into small chunks of fixed time duration. The proposed framework leverages both DASH and HLS protocols for serving large number of users over the Internet using the HTTP protocol. Nowadays, most of cloud providers adopt the pay-as-you-go model [25], [26], whereby a user will pay only for the resources that he has indeed used, including computing, memory, storage and traffic communication [27]. Therefore, for reducing the cost, the proposed solution should use carefully the available resources in terms of network storage, CPU and memory. The use of network storage should be minimized as much as possible, as well as the different VNFs should be created/launched only when they are needed for reducing the cost. In the previous approaches, for serving users with variant resolutions, each video is pre-transcoded to those resolutions before the streaming process, which demands high amounts of storage, and consequently increases the cost. In order to overcome this problem, the proposed framework adopts the online transcoding and live streaming of variant videos throught the Real time Message Protocol (RTMP). The RTMP module, integrated with FFMPEG at the transcoding VNF, allows the generation of multiple live streaming flows from the same video content, whereby each stream flow refers to a specific resolution. These streaming flows would be received by the streaming servers that are responsible for streaming each flow to an intended group of users by taking into account the network bandwidth and their devices' features.

In order to reduce the cost while ensuring the QoE for different users, an intelligent algorithm would be proposed that will be executed at the orchestrator level. Mainly, the algorithm would decide on the number and the location (i.e. on which cloud network) of streaming and transcoding VNFs that should be created in the network. Moreover, the algorithm should select the streaming and transcoding VNFs for each video content and resolution flow, such that the cost is minimized and the QoE is maximized. According to the popularity of the video, CPU, memory and network storage costs, the algorithm decides for each video content the resolutions that should be pre-transcoded. Fig. 2 shows the sequence diagram of the proposed architecture. When a user is interested in a specific live video, a request would be sent to the orchestrator. According to the existing flow of that video, the resolution needed by that user and the expected QoE, the orchestrator performs the following test: If a streaming flow that meets the user's requirements and ensures the QoE already exists, the orchestrator informs the concerned streaming and transcoding VNFs, as well as the user (Fig. 2: Arrow 3). Then, the user starts getting different chunks from the streaming and transcoding VNFs (Fig. 2: Arrows $7-10$). Otherwise, if the respective streaming flow does not exist, one of the following options would be executed (Fig. 2: Arrow 4):

1) *Available CPU and memory resources*: A new live streaming flow would be created in existing streaming and transcoding VNFs that meet the user's requirements and ensure QoE. Then, the orchestrator informs the

concerned streaming and transcoding VNFs, as well as the user about the new streaming flow.

2) *CPU and memory resources are unavailable*: If the CPU and memory resources are unavailable, one of the following options will be carried out according to the video's popularity, CPU, memory and network storage costs:

   a) *Streaming the video without live transcoding*: If the available resources, in terms of CPU and memory, at the streaming VNFs is sufficient, as well as the available network storage, then the orchestrator informs the global transcoder to transcode the video content to the desired resolution. Then, the orchestrator will inform the concerned streaming VNF, as well as the user about the new streaming flow. Note that the new streaming flow needs only packaging (no live transcoding), which consumes too much CPU resources as will be shown in Section IV.

   b) *Scaling up the streaming and/or transcoding VNFs*: The orchestrator will communicate with the VIM in order to scale-up (add more virtual resources in terms of CPU and memory) to the existing streaming and transcoding VNFs in order to meet the user's requirements and ensure the QoE. Then, the new streaming flow will be created in these VNFs. Later, the orchestrator will inform the concerned streaming and transcoding VNFs, as well as the user about the new streaming flow. Note that when a set of live streaming flows are finished, the orchestrator can scale-down some VNFs in order to release their resources, thereby reducing their otherwise incurred cost.

   c) *Scaling out the streaming and/or transcoding VNFs*: Assigning a new set of users to the streaming and transcoding VNFs that are located in far away geographical areas can affect the QoE experienced by the users. For this reason, if the existing streaming and transcoding VNFs cannot serve the incoming users with the required QoE, then new instances, that ensure QoE, should be created for serving those users. If so, the orchestrator would request the creation of new streaming and transcoding VNFs from the VIMs of variant clouds. Then, the orchestrator would establish communication with the new streaming and transcoding VNFs in order to generate the needed streaming flows to serve those users. Moreover, the orchestrator informs the users about their streaming flows. Note that when the live streaming flows, created in these VNFs, are finished, the orchestrator can delete (scale-in) these VNFs to release the resources, thereby reducing the cost.

In order to find the set of appropriate thresholds for the aforementioned policies, it is essential to find the maximum number of: i) Online transcoding flows that a transcoding VNF can process using variant flavors; ii) Live streaming without transcoding (i.e., packaging) that a transcoding VNF can process using different flavors. This will help us get the appropriate aforementioned policies in a proactive manner, thus ensuring the continuity of services.

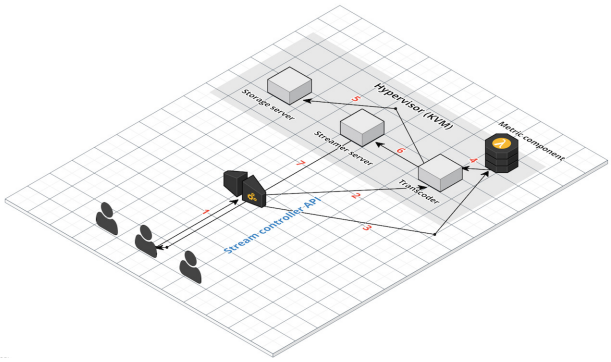


Fig. 3. Testbed experiment used for performing the performance benchmarking.

IV. Testbed and Experimental Evaluation

We benchmarked the streaming and transcoding VNFs by varying both the CPU and memory resources. Our virtualized environment is set up on a KVM Hypervisor in a dual Intel $E3-1231$ computer node. In this setup, three flavors were considered for benchmarking the streaming and transcoding VNFs as detailed in Table I.

TABLE I
Deployment flavors.

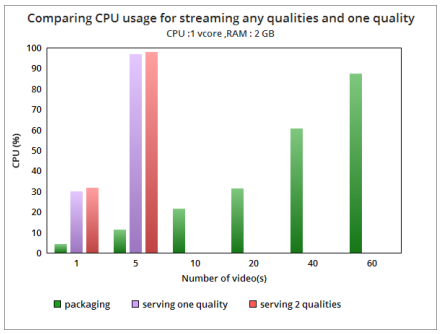
Deployment Flavor	Mini	Small	Medium
CPU	1 core	2 cores	4 cores
RAM (GB)	2	4	8

We have conducted three sets of experiments: i) video streaming only without transcoding (i.e. Packaging); ii) video transcoding and streaming in only one resolution; and iii) video transcoding and streaming in two different resolutions. The impact of the streaming and transcoding process is evaluated in terms of the following metrics:

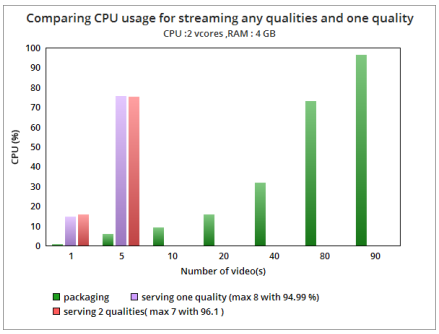
- Percentage of CPU usage that shows the ability of each flavor in handling the streaming and transcoding processes. This metric gives the main overview on the cost that would be incurred by the live streaming and online transcoding in the proposed framework;
- Response time that represents the impact of streaming and transcoding processes, using different flavors, on the QoE.

In order to perform the benchmarking, we use the testbed depicted in Fig. 3. The testbed consists of four VMs:

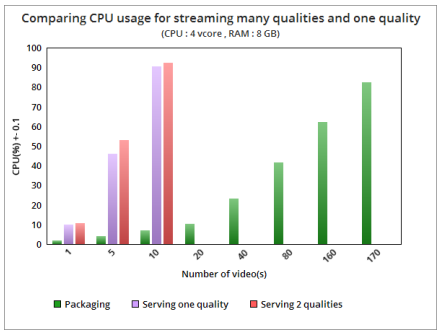
- Storage server that contains different videos that would be streamed. The NGINX, with mod.264 enabled, software is installed on that server for enabling live transcoding over HTTP;



(a) 1-Core CPU and $2GB$ as memory.



(b) 2-Core CPU and $4GB$ as memory.



(c) 4-Core CPU and $8GB$ as memory.

Fig. 4. The percentage of CPU usage measured when different flavors are used for instantiating VNFs.

- Transcoder server that is responsible for the live transcoding of the different stored videos with different qualities. It is also responsible for creating different streaming flows using the RTMP module;
- Streaming server that is responsible for streaming the different flows to end-users;
- The streaming controller API, which is a restful web service, that is responsible for receiving users' requests and forwarding them to the transcoding server. In addition, we have developed a metric component that is installed at the transcoding server, which is responsible for measuring the CPU usage and the response time.

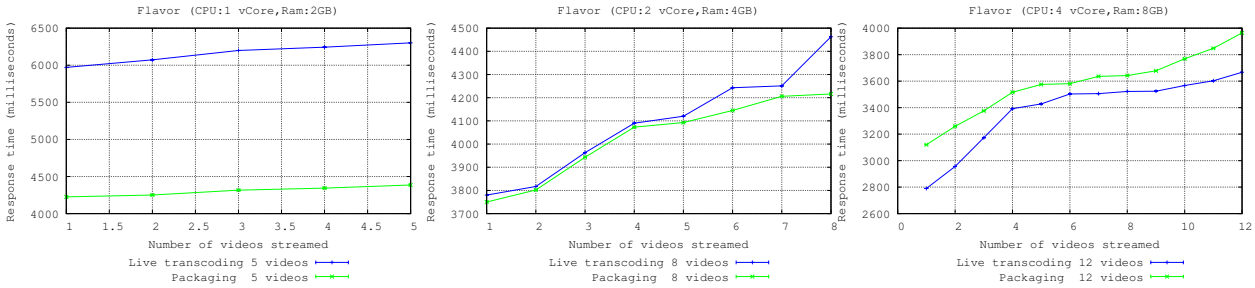
As depicted in Fig. 3, for launching the streaming and/or transcoding process, a request is sent to the streaming controller API component. The latter would then send two different messages: i) the first message is sent to the transcoder server and includes the following information: a) the video path in the storage server; b) the desired resolutions; and c) the IP address of the streaming server (step 2); ii) The second message launches the metric component at the transcoding server in order to start measuring the CPU usage and the response time (Step 3). For starting the streaming and transcoding processes, the transcoder starts getting the required videos from the storage server using HTTP (Steps 4 and 5). Then, the transcoder starts forwarding the chunks to the clients through the streaming server, using the RTMP module [28], while the metric component simultaneously measures the CPU usage and the response time (Steps 6 and 7).

A. Percentage of CPU usage

Fig. 4 shows the impact of streaming and transcoding processes on the CPU usage. Figs. $4(a)$, $4(b)$, and $4(c)$ depict the CPU utilization when the Mini, Small, and Medium flavors are used, respectively. The first observation that can be drawn from these figures is that the live transcoding and streaming process requires too much resources (i.e., CPU and memory) in comparison to when performing only online live streaming (i.e., packaging). For example, it is possible to stream up to 60 videos using only one CPU-core and 2 GB as memory as depicted in Fig. $4(a)$. Using the same flavor, it is not possible to transcode live more than 5 videos with one or two resolutions. As shown in Figs. $4(b)$ and $4(c)$, 90 and 170 stream flows could be successfully created using the 1^{st} and 2^{nd} VM flavor, respectively. In contrast, it was not possible to transcode live more than 5 and 10 videos for one or two resolutions using the 1^{st} flavor and the 2^{nd} flavor, respectively. The figure also reveals the fact that the transcoding operation of the same video to one or two resolutions requires almost the same amount of resources in terms of CPU usage. This is mainly attributable to the fact that for performing live transcoding, FFMPEG needs to decode and encode videos again. The decoding process requires more CPU resources than the encoding. For transcoding the same video into two different resolutions, FFMEG needs only one decoding and two encoding processes. For this reason, transcoding to one or two resolutions exhibit similar performance.

B. QoE and response time

Fig. 5 shows the impact of the streaming and transcoding processes on QoE and the response time. Figs. $5(a)$, $5(b)$, and $5(c)$ show the response time when using one CPU-core and 2 GB as memory, 2 CPU-cores and 4 GB as memory, and 4 CPU-cores and 8 GB as memory, respectively. From this figure, we observe that the response time when performing only the live streaming (packaging) process is intuitively better than when performing both live transcoding and streaming processes. For example, as depicted in Fig. 5(a), for one CPU-core and 2 GB as memory, while the response time does not exceed 4400 milliseconds when the live streaming process is performed, it exceeds 6400 milliseconds when both the live transcoding and streaming processes are performed. We also observe that the VNF's resource has a positive impact on the response time. Clearly, the higher a VNF's resources are, the better the response time becomes. Moreover, from these figures, we observe that the gaps between performing only the live streaming process and performing both live transcoding and streaming process is reduced by increasing the amount of CPU and memory resources used by VNFs. As depicted in Fig. 5(a) and Fig. 5(c), the gap is reduced from 1850 milliseconds to 350 milliseconds when using the 3^{rd} configuration (i.e., 4 CPU-cores and $8GB$ as memory) instead of the 2^{nd} configuration (i.e., One CPU-core and $2GB$ as memory).



(a) 1-Core CPU and $2GB$ as memory. (b) 2-Cores CPU and $4GB$ as memory. (c) 4-Cores CPU and $8GB$ as memory.

Fig. 5. The response time measured when different flavors are used to instantiate VNFs.

V. Conclusion

In this paper, we proposed a framework that allows live transcoding and streaming on top of multiple cloud domains for ensuring high QoE while reducing the cost. We have also benchmarked the streaming and transcoding VNFs by varying both CPU and memory resources. The benchmarking results will help for making the right decisions about scaling up/out and down/in different streaming and transcoding VNFs to reduce the cost and to ensure QoE. In the near future, We plan to use the obtained results to devise an intelligent algorithm for delivering different policies at the orchestrator. The algorithm will leverage different optimization techniques, including game theory, linear integer programing and convex optimization, for making the right decisions.

Acknowledgement

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

References

[1] Network functions virtualisation (NFV); management and orchestration. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[2] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale p2p iptv system," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1672–1687, Dec 2007.

[3] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer internet video broadcast," *Proceedings of the IEEE*, vol. 96, no. 1, pp. 11–24, Jan 2008.

[4] S. Retal, M. Bagaa, T. Taleb, and H. Flinck, "Content delivery network slicing: QoE and cost awareness," in *Proc. IEEE ICC 2017, Paris, France, May 21-25*, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICC.2017.7996499

[5] Z. Nadir, M. Bagaa, and T. Taleb, "Cost aware caching and streaming scheduling for efficient cloud based TV," in *Proc. IEEE ICC, Paris, France*, May 2017, pp. 1–6.

[6] Global internet phenomena report 2h 2014: Sandvine intelligent broadband networks, 2014.

[7] Cisco, "White paper: Cisco vni forecast and methodology, 2015-2020," in *This forecast is part of the Cisco Visual Networking Index (Cisco VNI)*, 2015.

[8] S. Wang and S. Dey, "Adaptive mobile cloud computing to enable rich mobile multimedia applications," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 870–883, June 2013.

[9] S. Dutta, T. Taleb, P. A. Frangoudis, and A. Ksentini, "On-the-fly qoe-aware transcoding in the mobile edge," in *Proc. IEEE GLOBECOM, Washington, DC, USA*, Dec 2016, pp. 1–6.

[10] P. A. Frangoudis, L. Yala, A. Ksentini, and T. Taleb, "An architecture for on-demand service deployment over a telco cdn," in *Proc. IEEE ICC, Kuala Lumpur, Malaysia*, May 2016, pp. 1–6.

[11] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang, "Video transcoding: an overview of various techniques and research issues," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 793–804, Oct 2005.

[12] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 18–29, Mar 2003.

[13] X. Li, M. A. Salehi, and M. Bayoumi, "High performance on-demand video transcoding using cloud services," in *Proc. IEEE/ACM CCGrid, Cartagena, Colombia*, May 2017, pp. 600–603.

[14] C. Wang, B. Li, J. Wang, H. Zhang, H. Chen, Y. Xu, and Z. Ma, "Single-input-multiple-ouput transcoding for video streaming," in *Proc. IEEE MMSP*, Sept 2016, pp. 1–5.

[15] S. M. Farhad, M. S. I. Bappi, and A. Ghosh, "Dynamic resource provisioning for video transcoding in iaas cloud," in *Proc. IEEE HPCC/SmartCity/DSS, Sydney, NSW, Australia*, Dec 2016, pp. 380–384.

[16] L. Qian, Z. Cheng, Z. Fang, L. Ding, F. Yang, and W. Huang, "A qoe-driven encoder adaptation scheme for multi-user video streaming in wireless networks," *IEEE Transactions on Broadcasting*, vol. 63, no. 1, pp. 20–31, March 2017.

[17] X. Li, M. A. Salehi, and M. Bayoumi, "VLSC: Video live streaming using cloud services," in *Proc. IEEE BDCloud-SocialCom-SustainCom, Atlanta, GA, USA*, Oct 2016, pp. 595–600.

[18] D. Karakasilis, F. Georgatos, L. Lambrinos, and T. Alexopoulos, "Application of live video streaming over grid and cloud infrastructures," in *Proc. IEEE 11th International Conference on Computer and Information Technology, Pafos, Cyprus*, Aug 2011, pp. 379–383.

[19] L. Koskimies, T. Taleb, and M. Bagaa, "QoE estimation-based server benchmarking for virtual video delivery platform," in *Proc. IEEE ICC, Paris, France*, May 2017, pp. 1–6.

[20] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *Proc. IEEE WCNC , Istanbul, Turkey*, April 2014, pp. 2402–2407.

[21] O. ElMarai, T. Taleb, M. Menacer, and M. Koudil, "On improving video streaming efficiency, fairness, stability & convergence time through client-server cooperation," *In Press in IEEE Trans. on Broadcasting.*

[22] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, March 2017.

[23] C. Concolato, J. L. Feuvre, F. Denoual, F. Maze, N. Ouedraogo, and J. Taquet, "Adaptive streaming of hevc tiled videos using mpeg-dash," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2017.

[24] J. Kua, G. Armitage, and P. Branch, "A survey of rate adaptation techniques for dynamic adaptive streaming over http," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2017.

[25] F. Z. Yousaf and T. Taleb, "Fine-grained resource-aware virtual network function management for 5g carrier cloud," *IEEE Network*, vol. 30, no. 2, pp. 110–115, March 2016.

[26] T. Taleb, "Toward carrier cloud: Potential, challenges, and solutions," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 80–91, June 2014.

[27] F. Z. Yousaf and T. Taleb, "Fine-grained resource-aware virtual network function management for 5g carrier cloud," *IEEE Network*, vol. 30, no. 2, pp. 110–115, March 2016.

[28] X. Lei, X. Jiang, and C. Wang, "Design and implementation of streaming media processing software based on rtmp," in *Proc. ICISP, Chongqing, China*, Oct 2012, pp. 192–196.