Content Delivery Network Slicing: QoE and Cost Awareness

Sara Retal*†, Miloud Bagaa*, Tarik Taleb* and Hannu Flinck ‡
*Dep. of Communications and Networking, School of Elect Eng, Aalto University, Espoo, Finland
†Computer Science Laboratory, Faculty of Sciences, Mohammed V University in Rabat
‡Nokia Bell Labs, Finland
Emails: *firstname.lastname@aalto.fi, ‡hannu.flinck@nokia-bell-labs.com

***Abstract*—Content Delivery Networks (CDNs) emerged to manage the great amount of content, as well as the transmissions over long distances. In recent years, this concept proves to be a promising solution for emergent enterprises. In this paper, we present a Content Delivery Network as a Service (CDNaaS) platform which can create virtual machines (VMs) through a network of data centers and provide a customized slice of CDN to users. CDNaaS manages a great number of videos by means of caches, transcoders, and streamers hosted in different VMs. However, an optimal placement of VMs with adequate flavors for the different images is required to obtain an efficient slice of CDN. In this work, we argue the need to find a convenient slice for the CDN owner while respecting his performance requirements and minimizing as much as possible the incurred cost. We first formulate the VMs placement problem as two Linear Integer problem solutions, aiming at minimizing the cost and maximizing the quality of experience of streaming. Then, extensive simulation results are presented to illustrate the effectiveness of the proposed models.**

I. Introduction

Over the last decade, content delivery networks (CDNs) have played a valuable role in hosting and distributing content to users. Thanks to its architecture that consists of multiple servers distributed geographically, content is replicated across a wide area, and has accordingly become highly available. Several studies have demonstrated the effectiveness of CDNs in improving the quality of experience (QoE) by making applications and services faster and more reliable [1]–[3]. One thing is certain, this concept has helped many renowned companies to develop and to expand their revenues. CDNs can improve the access by caching and streaming content, with many distributed components collaborating to deliver content across different network nodes. Hence, CDN providers have in general distributed topologies around the world . Moreover, those providers have two types of users: *(i)* the customers are CDN administrators who must pay fees to the supplier; *(ii)* CDN clients are the end users who download content through the CDNs.

Cloud providers own a number of globally distributed data centers which is growing continuously. Their different services including compute, storage, network, and virtualization allow elasticity and provide customers more choice than ever before. Most cloud providers use machine virtualization to provide flexible and cost effective resources, and the price can vary depending on the performance of the Virtual Machines (VMs). Recently, a great number of companies such as Amazon, Google, and Microsoft, have launched their cloud service businesses. Nowadays, users rent machine instances with different capabilities as needed and pay at a certain per machine hour billing rate. Thanks to machine virtualization techniques, flexible and cost-effective resources are provided for users. For example, Amazon EC2 [4] solution supports multiple VM instances on a single physical server. However, CDN infrastructure can benefit from these virtualization techniques and gain geographically dispersed nodes in large scale [5].

To deliver content to end users with QoE guarantees, a CDN administrator should ensure that his content is strategically placed across the Web [6], [7]. This can be done thanks to some algorithms which specify the location of VMs running the applications, in order to achieve an improved performance with a low infrastructure cost. Some parameters are of crucial importance such as the number of VMs, where those VMs are geographically located, and which server will serve end users' requests. In the context of CDNs, several studies have been conducted proposing different algorithms to place strategically servers. In [8], the authors propose a dynamic programming algorithm for cache placement aiming at improving the performance of CDNs. Other algorithms have been proposed for this issue, e.g., the Greedy algorithm [9], [10]. The VM placement problem is of vital importance and differs from one case to another. Generally speaking, the problem is to place a number of servers in different locations in a way that yields the lowest cost. For example in Carrier Cloud [11], [12], the placement of VMs was the subject of several kinds of research [13]–[16]. The goal here is the placement of VMs in specific data centers for a given user respecting architecture constraints. This problem can be studied in two ways [16]: *(i)* in the same data center, the research has been done with the goal of reducing costs using Bin packing, Simulated Annealing, Ant Colony, Transient cooling effects, N-dimensional set, etc. *(ii)* through a group of data centers. However, as shown in the following, it is necessary to consider an intelligent VMs placement through a group of data centers for the CDN owner to provide better services and satisfy his performance requirements. In this paper, we propose a CDN as a service (CDNaaS) platform where the user can create a CDN slice including caches, transcoders, and streamers, in order to manage a number of videos for a number of subscribers. This platform is designed to have the maximum level of flexibility for integrating with different public and private infrastructure as a service (IaaS) providers such as Amazon AWS service [4], Microsoft Azure [17], Rackspace [18], and OpenStack-managed cloud [19]. And thus, cache, transcoder, and streamer images could be hosted. Furthermore, CDNaaS

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

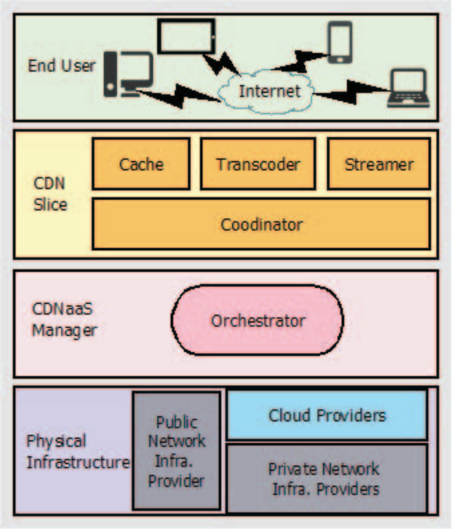


Fig. 1. Key stakeholders of a CDNaaS architecture.

proposes an efficient cost of CDN slice with a number of VMs and its locations for each image and also with an efficient QoE. Consequently, the challenge is to provide a delicate balance between costs and customers satisfaction. Therefore, the objective of this paper is to find an efficient cost of CDN slice respecting, on one hand, the CDN owner requirements in terms of QoE, and on the other hand, the cloud infrastructure and its costs. Two solutions are proposed for image placement over the cloud.

The rest of this paper is organized in the following way. In Section II, an overview of the proposed CDNaaS platform is presented. In Section III, system model and problem formulation for VM allocation problem are given. Section IV presents the optimal solutions and illustrates the simulation results. Finally, we conclude this study in Section V.

II. CDN As A Service Platform

The envisioned CDNaaS platform is designed to offer to users an easy way to manage a great number of videos for plenty of subscribers, providing different VM instances from different cloud suppliers [7], [20]. This platform consists of the following components: an orchestrator, a coordinator server, a cache server, a transcoding server, and a streaming server. The orchestrator enables users to connect via a web interface and create, modify or delete their respected CDN slices over the available IaaS providers. Fig. 1 shows the most important stakeholders in the envisioned CDNaaS architecture. It consists of a public network connecting different data centers across several geographical areas. Server racks within a data center are connected through a private network. The virtual infrastructure is created using different infrastructure providers, e.g., Amazone AWS service [4], Microsoft Azure [17], and Rackspace [18]. The CDNaaS manager architecture is depicted in Fig. 2. The orchestrator acts as the main management component of the CDNaaS platform and is responsible for running core front and back-end services. Front-end services help to meet users' preferences and set up VMs for coordinator, cache, transcoder, and streamer images. Regarding the back-end related functions used in the orchestrator, they are as follows:

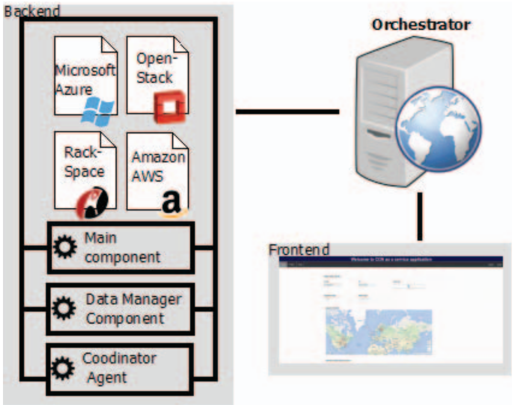


Fig. 2. CDNaaS orchestrator's architecture.

- Main component: The CDNaaS main component is responsible for checking the database and VMs creation and deletion.
- Data Manager Component: This component contains all database-related methods required for data management.
- Coordinator Agent: This agent contains required methods used for communications among orchestrator and coordinator servers in different CDN slices.
- Amazon AWS Agent: This agent contains methods required for handling Amazon AWS IaaS provider.
- Microsoft Azure Agent: This agent contains methods required for handling Microsoft Azure IaaS provider.
- OpenStack Agent: This agent contains methods required for handling an Openstack-based IaaS provider.
- RackSpace Agent: This agent contains methods required for handling Rackspace IaaS provider.

The creation of a CDN slice is depicted in Fig. 3. The process follows these steps:

- R1: A user requests the creation of a CDN slice specifying its requirements.
- R2: The orchestrator sends requests to each IaaS provider indicating the VM instances to be instantiated and specifying the images (i.e., Virtual Network Functions - VNFs - cache, transcoder, streamer, and coordinator) to be run on each of them.
- R3: VMs for all images are created and information on these VMs are communicated to the orchestrator.
- R4: The user (i.e., CDN owner) can manage the VMs of his CDN slice via the orchestrator.

The coordinator server is responsible for getting information about available machines in each CDN slice from the orchestrator and manages communications among the different nodes such as the transcode request (i.e., sent by the coordinator to the transcoder) or the transcode reply (i.e., sent by the transcoder to the coordinator) as depicted in Fig. 4. A CDN slice consists of one coordinator, one or more transcoders, streamers, and caches. Once a CDN slice is created, the CDN owner can manage his videos through the coordinator, which is a mandatory component used to manage the entire

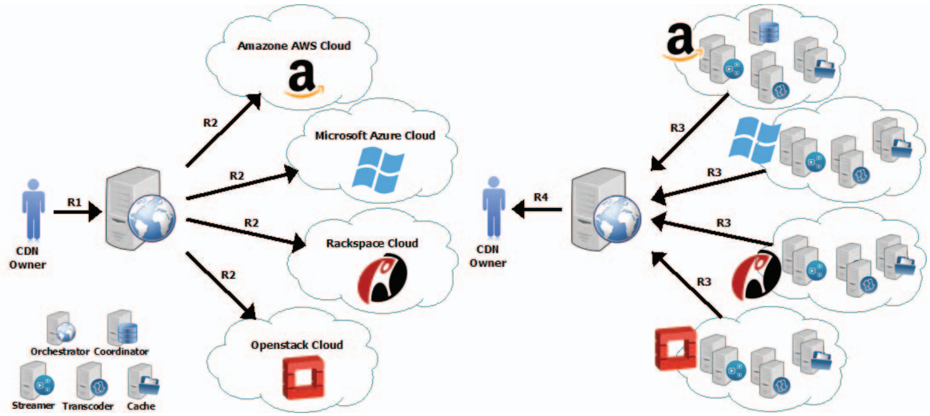


Fig. 3. Creation of a CDN slice.

CDN slice including caches, transcoders, and streamers. It enables users to upload, modify or delete videos, select the preferred transcoder, cache, and streamer among the available ones to transcode the video to desired resolutions, store and stream the transcoded videos. Caches are mainly in charge of storing videos after being uploaded by users and after being transcoded by the selected transcoder server. Transcoders get a request from the coordinator and transcode videos at rates specified by the CDN owner. The role of streamers is load balancing and receiving end users'requests for playing a specific video and redirecting the request to a proper cache server to show the video content using available resolutions (See Fig. 4). For the creation of cost-efficient and QoE-aware CDN slices, a smart placement of VNF images on adequate VM flavors must be ensured by the system. This placement concerns the geographical locations of VMs and their respected flavors (e.g., CPU, memory and storage) offered by providers [21]. Indeed, the placement of caches, transcoders, and streamers has a great impact on the QoE. Moreover, it affects the cost paid by the CDN owner (i.e., similar in spirit to the general VNF placement problem in case of cloud-based Telco) [13], [16]. In the following section, the problem formulation is given and two Linear Integer problem solutions are proposed for the CDNaaS VNF placement problem.

III. System model and problem formulation

Once a user successfully logs into the orchestrator domain, he is able to create a CDN slice. He defines a validity period for the CDN slice and specifies a number of videos to be stored and a number of subscribers to be serviced across a specific geographical area, with the service areas of a number of IaaS providers. Each IaaS provider can offer a number of VMs with specific flavors. A VM flavor defines the category of the VM instance and is characterized by a vCPU, a RAM, and a storage capacity, in addition to other feature [21]. Each flavor has a cost which should be paid by the CDN owner. Each VM shall run an image that can be either for a coordinator, a transcoder, a cache or a streamer. The objective is to create a cost-efficient CDN slice minimizing the incurred cost while meeting the requirements specified by the user.

We model the physical network representing the cloud infrastructure and subscribers as a weighted bipartite complete graph and denote it as $\mathcal{G} = (\mathcal{V}; \mathcal{E})$. The set of vertices is $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$ where $\mathcal{V}_1$ is the set of physical nodes constituting locations of data centers and $\mathcal{V}_2$ is the set of subscribers' locations. $\mathcal{E}$ is the set of physical links. We assume that the QoE of each physical link is different because of the distance between nodes. We note that the distance between subscribers' locations and the VM hosting a streamer has an impact on the QoE of streaming. Hence, $\omega(k, l) = \lambda_{kl}$ denotes the QoE of a physical link between two locations where $k \in \mathcal{V}_1$ and $l \in \mathcal{V}_2$.

A CDN owner defines a location $l \in \mathcal{V}_2$ of his subscribers, the minimum value of QoE (i.e., in terms of Mean Opinion Score) the consumers of his CDN slice shall experience, the capacities of the caches and the transcoders which are denoted as μ, ρ and σ, respectively. Subscribers in a location l and videos are denoted by the sets M_l and N, respectively. The CDN owner sets an estimated average duration of videos to be cached which we represent by t_v whereby $v \in N$ and desired resolutions $R = \{r_1, r_2 \ldots\}$ where $r_m = (w_m, l_m)$. R is the set of desired resolutions while w_m and l_m are, respectively, the width and the length of the frame.

In what follows, we consider the placement of caches, transcoders, and streamers in different locations and on different flavors. Let $F = \{f_1, f_2 \ldots\}$ denote the set of available flavors and c_i denote the cost of a flavor i. The relationship of flavors to a location $k \in \mathcal{V}_1$ is represented through the matrix $\mathcal{P}(F, \mathcal{V}_1)$. If and only if a flavor i is associated with a location k, then $\mathcal{P}(i, k) = 1$, otherwise $\mathcal{P}(i, k) = 0$.

Knowing that the available disk memory of a flavor has a great impact on the cache image, while the CPU and the RAM have a great impact on transcoder and streamer images, we assume that $h(i, Y)$ and $g(i, Y)$ are the capacities of a flavor i running a cache and a transcoder image, respectively, handling a size Y of videos. $q(i, M_l, \lambda_{kl})$ represents the QoE of a streamer image with a flavor i in a location $k \in \mathcal{V}_1$ serving a set of subscribers M_l in a location $l \in \mathcal{V}_2$. We assume that the QoE of a link between a streamer and a cache is efficient and does not affect q. $h^{-1}(i, \rho), g^{-1}(i, \sigma)$ and $q^{-1}(i, \mu, \lambda_{kl})$ are respectively the inverse functions of the functions $h(i, Y)$, $g(i, Y)$ and $q(i, M_l, \lambda_{kl})$. In other words, $h^{-1}(i, \rho)$ and $g^{-1}(i, \sigma)$ are the possible size of videos handled by a flavor i with the capacity ρ for the cache image and the

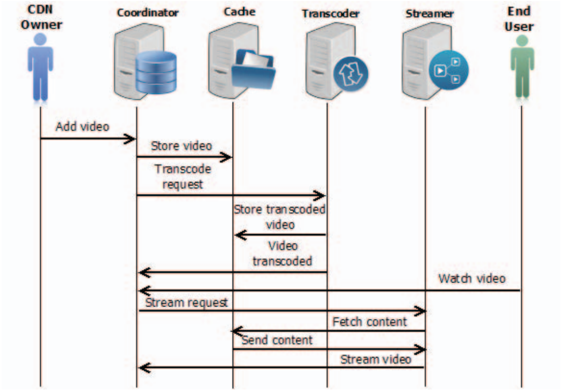


Fig. 4. Communication between CDN slice nodes.

capacity σ for the transcoder image. Similarly, $q^{-1}(i, \mu, \lambda_{kl})$ denotes the possible number of subscribers in a location $l \in \mathcal{V}_2$ that can be handled by a flavor i in a location $k \in \mathcal{V}_1$ with a QoE μ for the streamer image.

The function f estimates the cost of storing videos by calculating an approximate video size (in Megabytes), given the frame rate denoted by fr_v and the color depth denoted by d_v where $v \in N$. Knowing that $s_v = (t_v, fr_v, d_v) \in \mathbb{N}^3$ and $r_m = (w_m, l_m) \in \mathbb{N}^2$, this function is defined as follows:

$$\begin{array}{rcccc} f & : & \mathbb{N}^3 \times \mathbb{N}^2 & \rightarrow & \mathbb{R} \\ & & (s_v, r_m) & \mapsto & \frac{d_v \times fr_v \times t_v \times w_m \times l_m}{8 \times 1024 \times 1024} \end{array} \qquad (1)$$

The total size of videos is calculated using (1) as follows:

$$Y_{TOTAL} = \sum_{v \in N} \sum_{m \in R} f(s_v, r_m) \qquad (2)$$

Let $\mathcal{C}$, $\mathcal{T}$ and $\mathcal{S}$ denote cache, transcoder and streamer images, respectively. $E = \{\mathcal{C}, \mathcal{T}, \mathcal{S}\}$ is the set of all images. The number of VMs with a flavor i in a location $k \in \mathcal{V}_1$ hosting an image j is represented by S^k_{ij}. $S(F, E, \mathcal{V}_1) \in \Omega$ is a solution to the problem where Ω denotes the set of all possible solutions. The cost of a solution S is calculated as follows:

$$C_{TOTAL}(S) = \sum_{i \in F} \sum_{j \in E} \sum_{k \in \mathcal{V}_1} S^k_{ij} \mathcal{P}_{ik} c_i \qquad (3)$$

The aggregate utility minimization problem is shown as follows:

$$\begin{cases} \textbf{min} \quad C_{TOTAL}(S) \\ \textbf{s. t.} \\ \quad j = \mathcal{C} : \sum_{k \in \mathcal{V}_1} \sum_{i \in F} h^{-1}(i, \rho) S^k_{ij} \mathcal{P}_{ik} \geq Y_{TOTAL} \\ \quad j = \mathcal{T} : \sum_{k \in \mathcal{V}_1} \sum_{i \in F} g^{-1}(i, \sigma) S^k_{ij} \mathcal{P}_{ik} \geq Y_{TOTAL} \\ \quad j = \mathcal{S}, l \in \mathcal{V}_2 : \sum_{k \in \mathcal{V}_1} \sum_{i \in F} q^{-1}(i, \mu, \lambda_{kl}) S^k_{ij} \mathcal{P}_{ik} \geq M_l \\ \quad \forall i \in F, \forall j \in E, \forall k \in \mathcal{V}_1 : S^k_{ij} \in \mathbb{N} \\ \quad \forall i \in F, \forall k \in \mathcal{V}_1 : \mathcal{P}_{ik} \in \{0, 1\} \\ \quad Y_{TOTAL}, M_l, \rho, \mu, \sigma > 0 \end{cases} \qquad (4)$$

The objective aims at minimizing as much as possible the incurred cost and then find a cost-efficient slice of CDN. Meanwhile, the constraints in linear programming (4) are used to ensure the following conditions:

- Constraints 1 and 2 ensure that capacities of the cache and the transcoder desired by the user (i.e., CDN slice owner) must be respected. The size of videos handled by all flavors for a cache or a transcoder image must be higher than or equal to the total size of user's videos.
- Constraint 3 ensures that the QoE of streaming desired by the user must be respected. Hence, the number of subscribers handled by all flavors for a streamer image and by all physical links must be higher than or equal to the number of subscribers defined by the user.
- Constraint 4 ensures that the number of VMs is valid.
- Constraint 5 ensures that the matrix $\mathcal{P}$ is binary.
- Constraint 6 ensures that the total size of videos, the number of subscribers, the QoE of the streaming service, and the capacities of the transcoder and the cache are valid.

We define the matrix $\mathcal{N}(F, M_l)$. If and only if a flavor i handles a number of subscribers n in a location l, then $\mathcal{N}(i, n) = 1$, otherwise $\mathcal{N}(i, n) = 0$. Knowing that $j = \mathcal{S}$ and $l \in \mathcal{V}_2$, the total QoE of all flavors hosting a streamer image is calculated as follows:

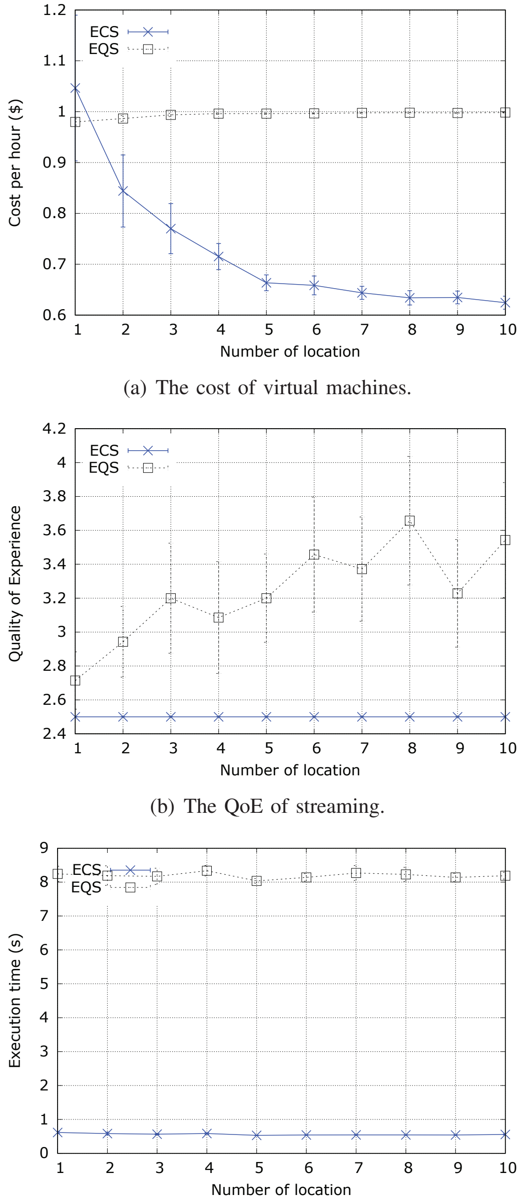
$$Q_{TOTAL}(S, \mathcal{N}) = \sum_{i \in F} \sum_{k \in \mathcal{V}_1} \sum_{n \in M_l} q(i, n, \lambda_{kl}) S^k_{ij} \mathcal{P}_{ik} \mathcal{N}_{in} \qquad (5)$$

Assuming that the CDN owner can define a maximum total cost denoted as $Cost_{Max}$, the aggregate utility maximization problem is shown as follows:

$$\begin{cases} \textbf{max} \, \underset{l \in \mathcal{V}_2}{\textbf{min}} \; Q_{TOTAL}(S, \mathcal{N}) \\ \textbf{s. t.} \\ \quad \sum_{k \in \mathcal{V}_1} \sum_{i \in F} \sum_{j \in K} S^k_{ij} \mathcal{P}_{ik} c_i \leq Cost_{Max} \\ \quad j = \mathcal{C} : \sum_{k \in \mathcal{V}_1} \sum_{i \in F} h^{-1}(i, \rho) S^k_{ij} \mathcal{P}_{ik} \geq Y_{TOTAL} \\ \quad j = \mathcal{T} : \sum_{k \in \mathcal{V}_1} \sum_{i \in F} g^{-1}(i, \sigma) S^k_{ij} \mathcal{P}_{ik} \geq Y_{TOTAL} \\ \quad j = \mathcal{S}, l \in \mathcal{V}_2 : \sum_{k \in \mathcal{V}_1} \sum_{i \in F} \sum_{n \in M_l} n \, S^k_{ij} \mathcal{P}_{ik} \mathcal{N}_{in} \geq M_l \\ \quad \forall i \in F, \forall j \in K, \forall k \in \mathcal{V}_1 : S^k_{ij} \in \mathbb{N} \\ \quad \forall i \in F, \forall k \in \mathcal{V}_1 : \mathcal{P}_{ik} \in \{0, 1\} \\ \quad \forall i \in F, \forall n \in M_l : \mathcal{N}_{in} \in \{0, 1\} \\ \quad Y_{TOTAL}, M_l, Cost_{Max}, \rho, \sigma > 0 \end{cases} \qquad (6)$$

The objective in linear programming (6) aims at maximizing as much as possible QoE of the streaming service while respecting the total cost paid by the user. Constraints in (6) are explained as follows:

- Constraint 1 ensures that the total cost desired by the user must be respected. The total cost of all flavors for caches, transcoders and streamers must be less than or equal to the total cost defined by the user.
- Constraints 2, 3, 5 and 6 are the same as in (4)
- Constraint 4 ensures that the total number of subscribers must be greater than or equal to the total number of subscribers defined by the user.
- Constraint 7 ensures that the matrix $\mathcal{N}$ is binary.
- Constraint 8 ensures that the total size of videos, the number of subscribers, the maximum cost, and the capacities of the transcoder and the cache are valid.



(a) The cost of virtual machines.

(b) The QoE of streaming.

(c) The operation time.

Fig. 5. Performance evaluation by increasing the number of locations.

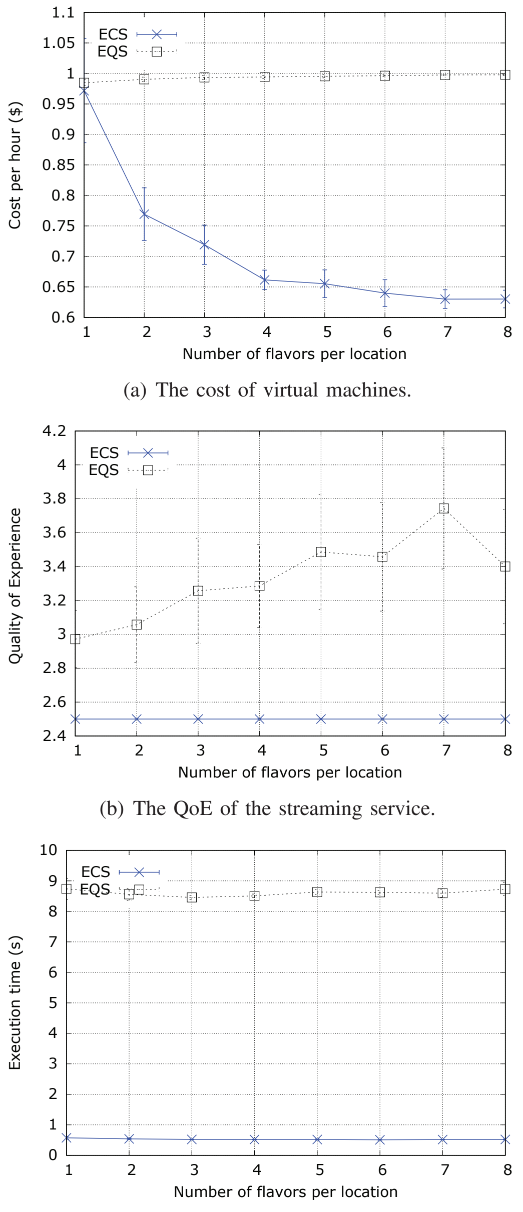
IV. Simulation results

To simulate our proposed solutions, a simulator was developed using the Python programming language. The two linear integer problem solutions are implemented using Gurobi optimization tool [22] and are evaluated through the following metrics: *(i)* the paid cost of virtual machines; *(ii)* the QoE of the streaming service; and *(iii)* the operation time. In the simulations, the linear programming (4) is dubbed Efficient Cost Solution (ECS), whereas the second one (6) is named Efficient QoE Solution (EQS). The optimization problems are solved by varying: *(i)* the number of data centers' locations; and *(ii)* the number of flavors per location. In the first scenario, we vary the number of locations of data centers and fix the number of flavors to 8 in each location. While in the second scenario, we vary the number of flavors in each location and fix the number of locations of data centers to 10. Flavors and their respective costs are obtained after examining the prices of 87 flavors offered by Amazone AWS service [4], Microsoft Azure [17], and Rackspace [18]. For both cases, the total number of subscribers, the total size of videos, and the capacities of the cache and the transcoder remain unchanged. For the sake of simplicity, we consider only one location of subscribers in $\mathcal{V}_2$. In the simulation results, each plotted point represents the average of 35 times of executions. The plots are presented with 95 % confidence interval.

Figs. 5 and 6 show the performance of the two proposed solutions when the number of flavors and the number of data centers' locations increase. In the ECS algorithm, the minimum acceptable QoE of the streaming service μ is set to 2.5, while in the EQS algorithm, $Cost_{Max}$ is set to 1$. In the two figures, QoE in the EQS algorithm is presented through the average of the QoE values of all virtual machines hosting a streamer image but not by Q_{TOTAL}. Fig. 5 (a) and 5 (b) show the total cost of virtual machines by varying the number of locations and by varying the number of flavors per location, respectively. In both figures, it is apparent that regardless the number of data centers' locations and the number of flavors per location, the ECS algorithm exhibits the best performance in terms of minimizing the total cost. When just one or two locations are considered, the cost in case of the ECS algorithm is high, and this also applies when the number of flavors per location is small in Fig. 6 (a). This is attributable to the fact that there is not much choice of flavors, then the cost could be high. Hence, the total cost decreases when the number of locations and the number of flavors increase. It can be noticed that the total cost in case of the EQS algorithm can be lower than that in case of the ECS solution and this is due to the fixed value of $Cost_{Max}$. Furthermore, a difference in the maximum cost is noticed between the two figures: when the number of data centers' locations is low, the cost is higher. Figs. 5 (b) and 6 (b) show the QoE of the streaming service while varying the number of data centers' locations and the number of flavors per location, respectively. As depicted in these two figures, the QoE of the streaming service in case of the EQS algorithm increases when the number of data centers' locations and the number of flavors per location become higher. That is tied to the great number of choices of flavors. The EQS algorithm exhibits the best performance in terms of maximizing QoE. In Figs. 5 (c) and 6 (c), the execution time of the two solutions is presented. As observed from the figures, the ECS algorithm exhibits better performance than the EQS algorithm in terms of time, regardless the number of data centers and the number of flavors per location.

V. Conclusion

In this paper, the study focused on virtual machine placement and flavors selection for different images in a CDNaaS platform. The platform manages a high number of videos deploying virtualized caches, transcoders, and streamers. A CDN slice owner can add videos specifying their resolutions



(a) The cost of virtual machines.

(b) The QoE of the streaming service.

(c) The operation time.

Fig. 6. Performance evaluation by increasing the number of flavors per location.

and these videos are streamed to the end-users, consumers of the CDN slice. To create an efficient CDN slice, virtual machines hosting cache functions, transcoder functions, and streamer functions must be assigned adequate flavors and must instantiated at adequate locations. Similarly, the total cost to be paid by the CDN slice owner must be efficient and fair. For this purpose, two solutions were proposed. The first one aims at minimizing the incurred total cost, whereas the second one aims at maximizing QoE. The results obtained from the conducted simulations demonstrate the efficiency of each proposed solution in achieving its key design goals.

Acknowledgment

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

References

[1] S. Gadde, J. Chase, and M. Rabinovich, "Web caching and content distribution: A view from the interior," *Computer Communications*, vol. 24, no. 2, pp. 222–231, 2001.

[2] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, Burlingame, CA, USA, 2001, pp. 169–182.

[3] A. Vakali and G. Pallis, "Content delivery networks: Status and trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, 2003.

[4] Amazon, "Amazon ec2 - virtual server hosting," 2016. [Online]. Available: https://aws.amazon.com/ec2/

[5] S. Dutta, T. Taleb, and A. Ksentini, "Qoe-aware elasticity support in cloud-native 5g systems," in *IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, 2016, pp. 1–6.

[6] 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, 2016.

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

[8] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby, "On the optimal placement of web proxies in the internet," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, New York, USA, 1999, pp. 1282–1290.

[9] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, Anchorage, USA, 2001, pp. 1587–1596.

[10] Y. Chen, R. H. Katz, and J. D. Kubiatowicz, "Dynamic replica placement for scalable content delivery," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 306–318.

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

[12] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis, and T. Magedanz, "Ease: Epc as a service to ease mobile core network deployment over cloud," *IEEE Network*, vol. 29, no. 2, pp. 78–88, 2015.

[13] T. Taleb and A. Ksentini, "Gateway relocation avoidance-aware network function placement in carrier cloud," in *Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems*, Barcelona, Spain, 2013, pp. 341–346.

[14] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Istanbul, Turkey, 2014, pp. 2402–2407.

[15] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *IEEE network operations and management symposium (NOMS)*, Krakow, Poland, 2014, pp. 1–9.

[16] T. Taleb, M. Bagaa, and A. Ksentini, "User mobility-aware virtual network function placement for virtual 5g network infrastructure," in *IEEE International Conference on Communications (ICC)*, London, UK, 2015, pp. 3879–3884.

[17] Microsoft, "Create linux and windows virtual machines in minutes," 2016. [Online]. Available: https://azure.microsoft.com/en-us/services/virtual-machines/

[18] Rackspace, "The industry leading open source technology," 2016. [Online]. Available: https://www.rackspace.com/cloud

[19] Openstack, "Open source software for creating private and public clouds." 2016. [Online]. Available: https://www.openstack.org/

[20] A. Nakao, P. Du, Y. Kiriha, F. Granelli, A. A. Gebremariam, T. Taleb, and M. Bagaa, "End-to-end network slicing for 5g mobile networks," *Journal of Information Processing*, vol. 25, pp. 153–163, 2017.

[21] 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, 2016.

[22] Gurobi, "Gurobi optimization." 2016. [Online]. Available: http://www.gurobi.com/