Benkacem, Ilias; Taleb, Tarik; Bagaa, Miloud; Flinck, Hannu

Optimal VNFs placement in CDN Slicing over Multi-Cloud Environment

Published in:
IEEE Journal on Selected Areas in Communications

DOI:
10.1109/JSAC.2018.2815441

Published: 12/03/2018

Document Version
Peer reviewed version

Please cite the original version:
Benkacem, I., Taleb, T., Bagaa, M., & Flinck, H. (2018). Optimal VNFs placement in CDN Slicing over Multi-Cloud Environment. IEEE Journal on Selected Areas in Communications, 36(3), 616-627. https://doi.org/10.1109/JSAC.2018.2815441

Optimal VNFs Placement in CDN Slicing Over Multi-Cloud Environment

Ilias Benkacem, Tarik Taleb, Miloud Bagaa, and Hannu Flinck

Abstract—**This paper introduces a content delivery network as a service (CDNaaS) platform that allows dynamic deployment and life-cycle management of virtual content delivery network (CDN) slices running across multiple administrative cloud domains. The CDN slice consists of four virtual network function (VNF) types, namely virtual transcoders, virtual streamers, virtual caches, and a CDN-slice-specific Coordinator for the management of the slice resources across the involved cloud domains. To create an efficient CDN slice, the optimal placement of its composing VNFs using adequate amount of virtual resources for each VNF is of vital importance. In this vein, this paper devises mechanisms for allocating an appropriate set of VNFs for each CDN slice to meet its performance requirements and minimize as much as possible the incurred cost in terms of allocated virtual resources. A mathematical model is developed to evaluate the performance of the proposed mechanisms. We first formulate the VNF placement problem as two Linear Integer problem models, aiming at minimizing the cost and maximizing the quality of experience (QoE) of the virtual streaming service. By applying the bargaining game theory, we ensure an optimal tradeoff solution between the cost efficiency and QoE. Extensive simulations are conducted to evaluate the effectiveness of the proposed models in achieving their design objectives and encouraging results are obtained.**

Index Terms—**Content delivery network, network function virtualization (NFV), slicing, network softwarization, edge cloud, optimization, bargaining game theory.**

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 [3]–[5]. Furthermore, the CDN concept has helped many renowned companies to develop and to expand their revenues. CDNs can improve the access to content by caching content nearby end-users, leveraging many distributed caches collaborating to deliver content across different network nodes. Hence, CDN providers have distributed topologies around the world. Moreover, these CDN providers have two types of users, namely *(i)* the customers – CDN administrators who must pay fees to the supplier; and *(ii)* CDN clients – the end users who download content through the CDNs.

Cloud providers own a number of globally distributed data centers which are expanding continuously. Their different services, including compute, storage, network, and virtualization, allow both elasticity and flexibility in service deployment. Most cloud providers use machine virtualization to provide flexible and cost-effective resources, and the price can vary depending on the computation resources demanded by the Virtual Network Functions (VNFs). 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. For example, the Amazon EC2 solution supports the instantiation of multiple VNF instances on a single physical server. However, CDN infrastructure can benefit from these virtualization techniques and gain geographically dispersed nodes in large-scale [6].

To deliver content to end users with QoE guarantees, a CDN administrator should ensure that his content is strategically placed across the globe [7], [8]. This can be done leveraging some algorithms [9]–[12] which specify the locations of VNFs running the applications, in order to achieve an improved performance with a low infrastructure cost. In this vein, some parameters are of crucial importance such as the number of VNFs, allocated virtual resources, where those VNFs are geographically located, and which VNF will serve end-users' requests. Oljira *et al.* [10] proposed a model for the placement of VNFs, guaranteeing the QoS and latency requirements of the service chains. The goal is to optimize resource utilization in order to reduce cost satisfying QoS such as end-to-end latency. A trade-off solution between the two conflicting objectives in terms of resource utilization and Service Level Agreement (SLA) requirements is proposed in [9] and [11]. Abu-Lebdeh *et al.* [12] focused on the VNF placement and aimed at minimizing the operational cost without violating

Manuscript received October 1, 2017; revised February 5, 2018; accepted February 27, 2018. Date of publication March 12, 2018; date of current version May 21, 2018. This work was supported in part by the Academy of Finland Project CSN under Grant 311654 and in part by the European Union's Horizon 2020 Research and Innovation Program through the 5G!Pagoda Project under Grant 723172. This paper was presented in part at the proceedings of the 2018 edition of the IEEE Wireless Communications and Networking Conference [1] and the 2017 edition of IEEE International Conference on Communications [2]. *(Corresponding author: Ilias Benkacem.)*

I. Benkacem and M. Bagaa are with the Department of Communications and Networking, School of Electrical Engineering, Aalto University, 02150 Espoo, Finland (e-mail: ilias.benkacem@aalto.fi; miloud.bagaa@aalto.fi).

T. Taleb is with the Department of Communications and Networking, School of Electrical Engineering, Aalto University, 02150 Espoo, Finland, and also with Sejong University, Seoul 143-747, South Korea (e-mail: tarik.taleb@aalto.fi).

H. Flinck is with Nokia Bell Labs, FI-02610 Espoo, Finland (e-mail: hannu.flinck@nokia-bell-labs.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2018.2815441

the performance requirements. They provided an integer linear programming formulation and proposed an algorithm to solve larger instances of placement problem with a significant reduction in the operational cost of large-scale distributed NFV deployments.

In this paper, we introduce a CDN as a Service (CDNaaS) platform whereby a user can create a CDN slice defined as a set of isolated distributed network of edge servers over multi-cloud domains where an edge server hosts a single VNF such as virtual cache, virtual transcoder, virtual streamer and a CDN-slice-specific coordinator for the lifecycle management of the slice resources and also for managing uploaded videos and subscribers. This platform is designed to have the maximum level of flexibility for scaling out and down a CDN slice on top of different public and private Infrastructure as a Service (IaaS) such as Amazon AWS service, Microsoft Azure, Rackspace, and OpenStack-managed cloud. Furthermore, the platform employs mechanisms and algorithms that create cost-efficient QoE-aware CDN slices, involving an optimal number of required VNFs and deciding on their optimal placement taking into account the desired QoE level. Hereby, the main challenge is to provide a delicate balance between cost and customer satisfaction (in terms of QoE). 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 cost. Three solutions are proposed for VNF placement across multiple cloud domains. While the first solution, dubbed Efficient Cost Solution (ECS), aims to minimize the cost as much as possible, the second solution, named Efficient QoE Solution (EQS), aims to increase as much as possible the QoE in the network. Meanwhile, the third solution, named Fair Trade-off between cost and QoE Solution (FTS), uses bargaining game theory to ensure a fair trade-off between cost and QoE.

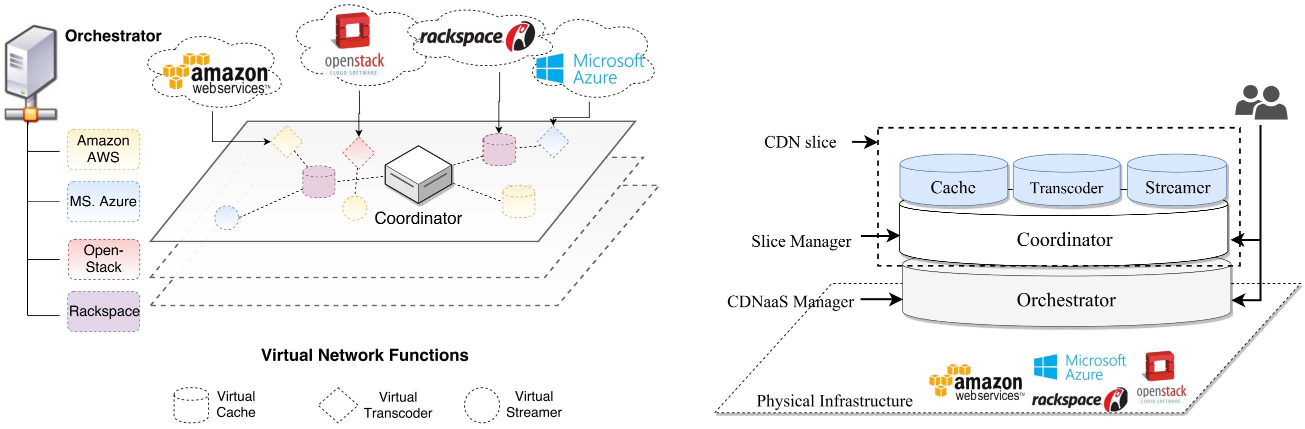
The rest of this paper is organized as follows. Some related work is summarized in Section II. In Section III, the proposed CDNaaS platform is introduced. In Section IV, the system model and problem formulation for VNF allocation problem are given. Section V presents the solutions proposed for placing different streaming VNFs, that reduce both cost and QoE. Section VI describes the simulation setup and discusses the obtained results. Finally, the paper concludes in Section VII.

II. Related Work

Regarding network slicing, network management and orchestration (MANO), there have been an important amount of research work conducted recently, summarized in [13] and [14]. Network slicing enables the deployment of multiple logical, self-contained networks on a common physical infrastructure, allowing resource isolation and a customized network operation as detailed in [15]. In other words, network slicing introduces a multi-tenant environment supporting flexible provisioning of network resources as well as dynamic instantiation and placement of virtual network functions [16]. Ordonez-Lucena *et al.* [17] provide a comprehensive study of the architectural frameworks of both Software Defined Networking (SDN) and Network Function Virtualization (NFV) as key enablers to achieve the realization of network slices. Peter and Christian [18] provide the necessary flexibility and scalability associated with future network implementations. The authors propose 5G based on network slicing with the coexistence of dedicated as well as shared slices in the network. Bin *et al.* [19] present a network slice design for the multicast/broadcast of ultra high definition (UHD) video to achieve higher network efficiency and improved QoE. Nakao *et al.* [20], Taleb *et al.* [21], and Zhang *et al.* [22] improve the flexibility of network resource allocation and the capacity of 5G networks based on network slicing and discuss the potential of network slicing to provide the appropriate customization and highlight the relevant technology challenges.

In the context of CDN, several studies have been conducted proposing different algorithms to place strategically servers. Li *et al.* [23] 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 [24], [25]. The VNF placement problem is of vital importance and differs from one case to another. Generally speaking, the problem is to place a number of VNFs in different locations in a way that yields the lowest cost. For example, in Carrier Cloud [26], [27], the placement of VNFs was the subject of several kinds of research [28]–[31]. The goal here is the placement of VNFs in specific data centers for a given user or a group of users, respecting architecture constraints and target service requirements. This problem can be studied in two ways [31]: *(i)* placement within the same data center – the research has been done with the goal of reducing cost using Bin packing, Simulated Annealing, Ant Colony, Transient cooling effects, N-dimensional set, etc., and *(ii)* placement across a federation of data centers. However, as discussed below, it is important to employ an intelligent VNF placement strategy across a federation of data centers for the CDN owner to provide better services, maintain efficient cost, and meet the performance requirements.

Zhang *et al.* [32] use a fair resource allocation to ensure fairness between cloud and big data applications while virtual machine migration is used to make each virtual machine in cloud application reach the desired level of performance. The authors used Nash Bargaining game to model the situation whereby virtual machines compete for more resources while their minimal demand is ensured. Iyer *et al.* [33] consider addressing the resource allocation and pricing strategies in a Compute Cloud. They introduce the concept of asymmetric pricing scheme wherein a user can specify his budget constraints and the cloud service providers can attempt to maximize the revenue without compromising the performance. The authors employ two axiomatic bargaining approaches, namely Nash Bargaining Solution (NBS) and Raiffa Bargaining Solution (RBS), to formulate the problem and derive an optimal solution for allocating virtual CPU instances in a Compute Cloud. In the context of mobile relay networks, Baharlouei and Jabbari *et al.* [34], Zheng *et al.* [35], and Zhang *et al.* [36] proposed a Nash bargaining approach to balance the information transmission efficiency of source-to-destination pairs and the residual harvested energy of relays.



(a) Multiple Fully-Fledged CDN Slices over Multiple Cloud Domains.

(b) Key Stakeholders of a CDNaaS Architecture.

Fig. 1. CDN as a Service Architecture.

III. CDN as a Service Platform

A. Architecture and Components

Recently, performance and reliability have become the major factors that directly impact the user experience. In our highly connected world and to efficiently serve an ever-growing community of mobile users demanding high-bandwidth services, it has become very important to reach clients whenever and wherever they are. In case of CDNs, the content is delivered to end-users based on their geographical locations and availability of resources using the geographically dispersed servers of CDNs. To make efficient usage of CDNs, it is important to cache video contents in an intelligent fashion, leveraging smart caching strategies that are based on content popularity and geographical distribution of end-viewers. Although video contents can be transcoded into various formats, this operation requires greedy processes that need a large amount of computation for decoding and encoding. Moreover, efficient media delivery requires high performance transcoding and that is in case of both Video On Demand (VOD) and live broadcast to various types of user devices.

For an efficient delivery of video services, our envisioned CDNaaS platform aims for improving service responsiveness, not only by replicating the contents in several caches to ensure their availability but also by running on-demand other virtual services, such as virtual transcoder to transcode videos on the fly into optimal resolutions before streaming them to the interested viewers [37]. In the following, we introduce the architecture of our CDNaaS platform along with its components.

Our envisioned CDNaaS platform allows the creation and life-cycle management of multiple slices of virtual CDNs running across multiple cloud domains. The CDN slices include virtual transcoders, virtual streamers, virtual caches, and a CDN-slice-specific Coordinator for the management of the uploaded videos and the subscribers as well as the slice resources across different private and public Infrastructure as a Service (IaaS) providers such as OpenStack-managed data-centers, Amazon AWS service, Microsoft Azure and Rackspace. Fig. 1.a shows an overview of the CDNaaS architecture. As stated earlier, a CDN slice consists of a number of virtual services and is administrated by only one server, called Coordinator, that manages the communication among VNFs of the same slice. As depicted in Fig.1.a, our envisioned CDNaaS platform consists of five main components:

- *The Orchestrator server:* The orchestrator server allows consumers to create new CDN slices or modify/delete their existing CDN slices over the available IaaS providers. The owner of a slice also logs into the orchestrator to manage the VM instances of the slice. This includes instantiation of new VMs specifying their flavors, termination of existing ones, and definition or update of policies for the management of slice resources (e.g., for scaling out and down). For every slice, the orchestrator updates its respective coordinator whenever new VMs are created for the slice or existing VMs of the slice are updated.

- *The Coordinator server:* Each CDN slice has only one NFV manager, called the coordinator, that ensures the communication among virtual cache servers, virtual transcoder servers and virtual streaming servers associated to the CDN slice. The owner of a slice can manage through its respective coordinator its videos and administers its subscribers. The coordinator runs concrete methods including the smart selection of the closest and least loaded virtual transcoder. It also defines the service function chaining (SFC) between the slice VNFs (i.e., the triplet Cache, Transcoder, and Streamer) and then publishes the jobs in the queues of the concerned servers.

- *The Cache server:* Basically, a CDN slice essentially consists of a network of geographically dispersed cache servers. Each node caches static content and stores videos uploaded by the end-users and also the transcoding output. When a user requests a video with a specific resolution and quality, the closest cache in proximity to that user will deliver the content, ensuring the shortest distance (i.e, latency), therefore providing the best user experience possible.

- *The Transcoder server:* This network function is in charge of transcoding videos in different formats and at different resolutions. It consumes high computing resources. The virtual transcoder is always listening to orders from the coordinator

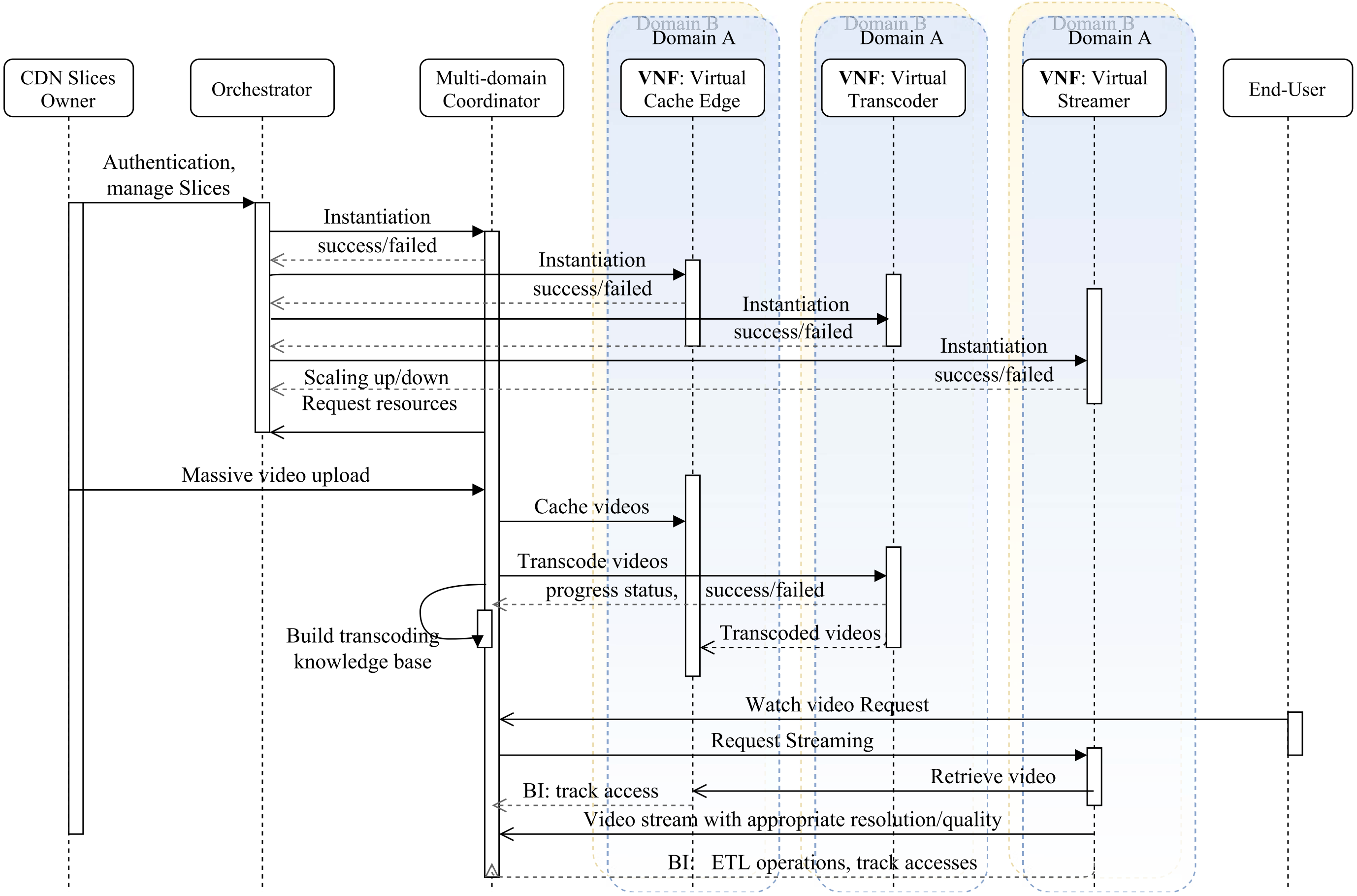


Fig. 2. Sequence diagram for managing resources of CDN slices and their videos in our envisioned CDNaaS platform.

by the mean of a queuing management server. It mounts the concerned cache server, starts transcoding, sends feedback to the coordinator specifying the progress in real-time and notifies the Coordinator once the transcoding operation is successfully completed.

- *The Streaming server:* This network function plays the role of a load balancer as it receives requests from end-users for playing specific videos and redirects the requests to proper cache servers. The streaming server tracks also the video accesses and sends the statistics back to the coordinator to be used in data analysis in order to improve the business intelligence of the CDN slice in question.

B. Sequence Diagram

Fig.2 shows the sequence diagram for managing resources of CDN slices and their videos in our envisioned CDNaaS platform. VNFs belonging to the same CDN slice communicate through a well-defined Application Program Interface (API). A log of such communication is stored to build a knowledge base for future data mining purpose. In the following, we explain the data flow between different components of the system and highlight the communication technologies used.

1) Communication Between the Orchestrator and VNFs: As stated earlier, the orchestrator of a CDN slice manages the life-cycle of its VNFs and accordingly scales out and down the CDN slice. The Orchestrator bootstraps virtual machines for hosting specific VNFs and updates constantly the respective VNF managers.

2) Communication Between the Coordinator and Transcoders: The CDN slice owner authenticates to the interface management via web and launches massive video uploads over a distributed set of virtual caches. The Coordinator load balances the uploaded videos over the network of virtual Transcoder nodes, by the means of an Advanced Message Queuing Protocol (AMQP) implemented in the Coordinator server, e.g, RabbitMQ Server. For each new instantiation of a transcoder node, a new queue is created in the system, that ensures the communication (Read/Write) between that transcoder and the coordinator. When a CDN slice owner decides to transcode some of his videos, the coordinator server publishes the jobs in the respective queue of each transcoder. The transcoders are always listening to their queues, consume the messages in First In First Out (FIFO) order. The message is an API that describes the order and contains information about the video, specifies the cache server where the video resides and information about the transcoding parameters. During the transcoding operation, the virtual transcoder publishes in the queue the progress in real time. A progress-bar will be rendered to the end-user through the coordinator web interface.

3) Communication Between the Coordinator and Streamers: Streaming servers receive requests for videos from end-users and load balance their streaming among the available caches. As a potential technology for the streaming server, we use Nginx. All streaming servers contain a running service that tracks the video accesses and publishes in the common streamers' queues the following information: the streamer's public IP address, timestamps, IP address of the requesting user, the viewer's city and country, video ID, resolution, the IP address of the cache server where the video resides, and the distance between the requesting client and the cache server in Kilometers. These information are needed to build a dashboard and statistics for the CDN slice owners to measure the performance and to also improve the Business Intelligence of the specific CDN slices.

4) Communication Among Transcoders, Caches, and Streamers: In many cases, it may become too much demanding to handle constant file transfers to and from virtual machines. For instance, there could be a situation whereby a transcoder is transcoding videos and simultaneously receiving the uploads of other large files from multiple end-users. This could largely downgrade the overall system performance. To cope with such an issue, transcoders may mount Virtual Private Server (VPS) file systems that consist of cache servers where videos can be stored. The transcoder can then make transcoding on the fly and treat the mounted cache server as local storage. Similarly, the virtual streaming servers can be configured to read/stream the content from the virtual caches belonging to the same slice, rather than from local caches associated with the virtual streaming servers.

C. Management and Orchestration

The envisioned CDNaaS platform is designed to offer users an easy way to manage a great number of videos for plenty of subscribers, providing the flexibility of launching different VNF instances using resources from different cloud suppliers [8], [20]. Fig. 1.b 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 IaaS providers, e.g., Amazone AWS service, Microsoft Azure, and Rackspace. The architecture of the CDNaaS orchestrator is depicted in Fig. 3. 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 VNFs for the coordinator, cache, transcoder, and streamer images. Regarding the back-end related functions used in the orchestrator, they are as follows:

- Main Orchestration component: This component is responsible for checking the database and VNFs creation and deletion.
- Data Manager Component: This component contains all database-related methods required for data management.
- Coordinator Agent: contains required methods used for communications between the orchestrator and the coordinators of different CDN slices. For example, the agent makes the coordinator aware of the topology of its slice. It populates the coordinator database with all useful information about the slice VNFs, including public and private IPs, network function/service, cloud provider, location (latitude and longitude), and image ID.The agent constantly updates the coordinator regarding any change in the slice topology in case of instantiation of new VMs or deletion of existing ones.
- Amazon AWS Agent: contains methods required for interfacing with the EC2 controller of Amazon AWS IaaS provider.
- Microsoft Azure Agent: contains methods required for interfacing with the controller of the Microsoft Azure IaaS provider.
- OpenStack Agent: contains methods required for interfacing with an OpenStack-based Virtual Infrastructure Manager (VIM).
- RackSpace Agent: contains methods required for interfacing with the controller of the RackSpace IaaS provider.

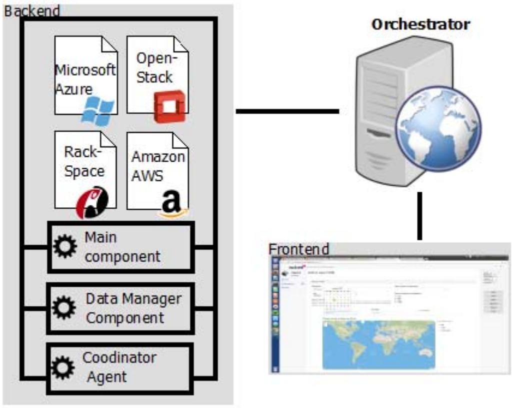


Fig. 3. High level architecture of our envisioned CDNaaS orchestrator.

The main steps beneath the creation of a CDN slice are depicted in Fig. 4:

- R1: A customer (i.e., CDN slice consumer) requests the creation of a CDN slice specifying its requirements.
- R2: Based on these requirements, the orchestrator determines the amount of virtual resources, their locations and respective IaaS provider, and the VNFs to be installed in each virtual instance. The orchestrator sends requests to each IaaS provider indicating the VNF instances to be instantiated and specifying the images (i.e., virtual cache, transcoder, streamer, and coordinator) to be run on each of them.
- R3: Using the right images, VNFs are created and information on the statues of these VNFs are communicated to the orchestrator.
- R4: The user (i.e., CDN owner) can manage the VNFs of his CDN slice via the API of the orchestrator.

The coordinator server of a specific CDN slice is responsible for getting information about available machines in its CDN slice from the orchestrator and manages communications among the different nodes such as the transcode request

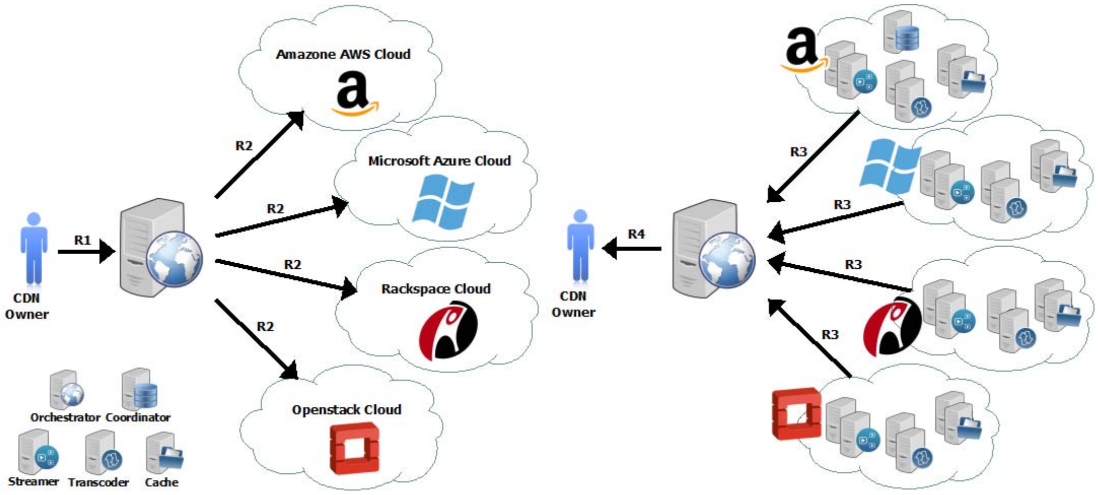


Fig. 4. Main steps beneath the creation of a CDN slice.

(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. 2. As stated earlier, a CDN slice consists of one coordinator, at least one transcoder, at least one streamer, and at least one cache. Once a CDN slice is created, the CDN slice owner can manage his videos through the coordinator, which is a mandatory component used to manage the entire CDN slice including caches, transcoders, and streamers. It enables the owner to upload, modify or delete videos, select the preferred transcoder, cache, and streamer among the available ones to transcode one or a group of videos 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 requests to proper cache servers (See Fig. 2). For the creation of cost-efficient and QoE-aware CDN slices, a smart placement of VNFs across the available IaaS along with decision on what virtual resources to allocate for each VNF must be ensured by the system. This placement concerns the geographical locations of VNFs and the flavors of their respected VMs (e.g., CPU, memory and storage) offered by available IaaS providers [38]. 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 [28], [31]). In the following section IV, the CDNaaS VNF placement problem is formulated and two Linear Integer Problem solutions are proposed. Based on these two solutions, an optimal trade-off solution based on bargaining game theory is also proposed.

IV. 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 an estimate of the number of videos to be stored and a number of subscribers to be serviced across a specific geographical area, withing the service areas of a number of IaaS providers. Each IaaS provider can offer a number of VNFs and run them on VMs with specific flavors. A VM flavor defines the category of the VM instance and is characterized by a number of cores as vCPU, an amount of RAM, and a storage capacity, in addition to other features [38]. Each flavor incurs a cost which should be paid by the CDN owner. Each instantiated VM shall run a VNF 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 (i.e., in terms of QoE).

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 considering the hops count that links two cloud platforms where Deterministic Networking (DetNet) can be used to estimate the packet delay variation and provide multi-hop forwarding path with the deterministic properties of controlled latency. In this article, we consider only the distance between subscribers' locations and the VNF hosting a streamer has an impact on the QoE of the streaming service. Hence, $\omega(k, l) = \lambda_{k,l}$ denotes the QoE of a physical link between two locations where $k \in \mathcal{V}_1$ and $l \in \mathcal{V}_2$ [39] [40].

A CDN slice 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 end-users 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 slice owner sets an estimated average duration of videos to be cached which we represent by t_v. The set of desired resolutions is denoted as $R = \{r_1, r_2 \ldots\}$ where $r_m = (w_m, l_m)$; w_m and l_m denote the width and the length of the frame, respectively.

TABLE I

NOTATIONS USED IN THE PAPER

Notation	Description
$\mathcal{V}$	Set of vertices $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$. $\mathcal{V}_1$ is the set of physical nodes constituting locations of data centers and $\mathcal{V}_2$ is the set of subscribers' locations.
μ	Quality of experience of streaming service. It varies from 1 to 5.
ρ	Capacity of the cache image. ρ is related to storage capabilities
σ	Capacity of the transcoder image. σ is related to processing capabilities
c_i	Cost of a flavor i.
$\mathcal{P}(F, \mathcal{V}_1)$	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 $i \in F$ is available in Location k, then $\mathcal{P}_{i,k} = 1$; otherwise $\mathcal{P}_{i,k} = 0$.
$\omega(k, l)$	Where $\omega(k, l) = \lambda_{k,l}$ denotes the QoE of a physical link between two locations where $k \in \mathcal{V}_1$ and $l \in \mathcal{V}_2$.
$h(i, Y)$	Capacity of flavor i running a cache image handling a size Y of videos
$g(i, Y)$	Capacity of flavor i running a transcoder image handling a size Y of videos.
$q(i, M_l, \lambda_{k,l})$	QoE of a streamer image running in a VM of flavor i in a location $k \in \mathcal{V}_1$, serving a set of subscribers M_l in Location $l \in \mathcal{V}_2$.
$h^{-1}(i, \rho)$	Size of videos handled by flavor i with the capacity ρ for the cache image.
$g^{-1}(i, \sigma)$	Size of videos handled by flavor i with the capacity σ for the transcoder image.
$q^{-1}(i, \mu, \lambda_{k,l})$	Number of subscribers in Location $l \in \mathcal{V}_2$ that can be served by flavor i in Location $k \in \mathcal{V}_1$ with a QoE μ for the streamer image.
Ω	Set of all possible solutions.
$S_{i,j}^k$	The number of VNFs running in VMs of flavor i in Location $k \in \mathcal{V}_1$ hosting an image j.

In what follows, we consider the placement of caches, transcoders, and streamers in different locations, running them on VMs with 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. Note that a flavor i can be available in a data center k but may not be available in another data center k'. The relationship of flavors to a location $k \in \mathcal{V}_1$ is represented through a constant 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 $\mathcal{P}_{i,k} = 0$. For the sake of readability, the notations used throughout the paper are summarized in Table I.

V. PROPOSED SOLUTIONS

A. ECS: Efficient Cost Solution

Knowing that the available disk memory of a VM of a specific 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 denote by $h(i, Y)$ and $g(i, Y)$ the capacities of a VM of flavor i running a cache and a transcoder image, respectively, handling a size Y of videos. $q(i, M_l, \lambda_{k,l})$ represents the QoE perceived by a streamer image running on a VM of 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 $q^{-1}(i, \mu, \lambda_{k,l})$ are respectively the inverse functions of the functions $h(i, Y)$, $g(i, Y)$ and $q(i, M_l, \lambda_{k,l})$. In other words, $h^{-1}(i, \rho)$ and $g^{-1}(i, \sigma)$ are the possible size of videos handled by a VM of flavor i with the capacity ρ for the cache image and the capacity σ for the transcoder image. Similarly, $q^{-1}(i, \mu, \lambda_{k,l})$ denotes the possible number of subscribers in a location $l \in \mathcal{V}_2$ that can be handled by a VM of flavor i in a location $k \in \mathcal{V}_1$ with a perceived 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_υ and the color depth denoted by d_υ where $\upsilon \in N$. Knowing that $s_\upsilon = (t_\upsilon, fr_\upsilon, d_\upsilon) \in \mathbb{N}^3$ and $r_m = (w_m, l_m) \in \mathbb{N}^2$, this function is defined as follows:

$$\begin{aligned} f : \mathbb{N}^3 \times \mathbb{N}^2 &\to \mathbb{R} \\ (s_\upsilon, r_m) &\mapsto \frac{d_\upsilon \times fr_\upsilon \times t_\upsilon \times w_m \times l_m}{8 \times 1024 \times 1024} \end{aligned} \tag{1}$$

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

$$Y_{TOTAL} = \sum_{\upsilon \in N} \sum_{m \in R} f(s_\upsilon, r_m) \tag{2}$$

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. As mentioned in Table I, we define the integer variable $S_{i,j}^k$ that denotes the number of VNFs running on VMs of flavor i in a location $k \in \mathcal{V}_1$ hosting an image j. $S(F, E, \mathcal{V}_1) \in \Omega$ 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:

$$C_{TOTAL}(S) = \sum_{i \in F} \sum_{j \in E} \sum_{k \in \mathcal{V}_1} S_{i,j}^k \times \mathcal{P}_{i,k} \times c_i \tag{3}$$

The aggregate utility minimization problem is shown as follows:

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

The objective is to minimize s much as possible the incurred cost and then form a cost-efficient CDN slice. Meanwhile, the constraints in linear programming (4) are used to ensure the following conditions:

- Constraints 1 and 2 ensure that the capacities of the cache and the transcoder desired by the user (i.e., CDN slice owner) are respected and not exceeded. The size of videos handled by all VMs of all selected flavors for cache or transcoders image must be higher than or equal to the total size of user's videos.

- Constraint 3 ensures that the QoE of the streaming service desired by the user is respected. Hence, the number of subscribers handled by all VMs of all selected 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 VNFs 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.

B. EQS: Efficient QoE Solution

First, 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 VMs of different flavors hosting a streamer image is calculated as follows:

$$Q_{TOTAL}(S, \mathcal{N}, l) = \sum_{i\in F}\sum_{k\in\mathcal{V}_1}\sum_{n\in M_l} q(i, n, \lambda_{k,l}) \times S^k_{i,j} \times \mathcal{P}_{i,k} \times \mathcal{N}_{i,n} \tag{5}$$

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

$$\left\{\begin{array}{l}
\mathbf{max}\min\limits_{l\in\mathcal{V}_2} \; Q_{TOTAL}(S, \mathcal{N}, l)\\
\mathbf{s.\ t.}\\
\sum\limits_{k\in\mathcal{V}_1}\sum\limits_{i\in F}\sum\limits_{j\in K} S^k_{i,j} \times \mathcal{P}_{i,k} \times c_i \leq Cost_{Max}\\
j = \mathcal{C} : \sum\limits_{k\in\mathcal{V}_1}\sum\limits_{i\in F} h^{-1}(i, \rho) \times S^k_{i,j} \times \mathcal{P}_{i,k} \geq Y_{TOTAL}\\
j = \mathcal{T} : \sum\limits_{k\in\mathcal{V}_1}\sum\limits_{i\in F} g^{-1}(i, \sigma) \times S^k_{i,j} \times \mathcal{P}_{i,k} \geq Y_{TOTAL}\\
j = \mathcal{S}, l \in \mathcal{V}_2 : \sum\limits_{k\in\mathcal{V}_1}\sum\limits_{i\in F}\sum\limits_{n\in M_l} n \times S^k_{i,j} \times \mathcal{P}_{i,k} \times \mathcal{N}_{i,n} \geq |M_l|\\
\forall i \in F, \forall j \in K, \forall k \in \mathcal{V}_1 : S^k_{i,j} \in \mathbb{N}\\
\forall i \in F, \forall k \in \mathcal{V}_1 : \mathcal{P}_{i,k} \in \{0, 1\}\\
\forall i \in F, \forall n \in M_l : \mathcal{N}_{i,n} \in \{0, 1\}\\
Y_{TOTAL}, Cost_{Max}, \rho, \sigma > 0
\end{array}\right. \tag{6}$$

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

- Constraint 1 ensures that the total cost desired by the user must be respected. The total cost of all VMs of all selected 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 the linear programming (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.

C. FTS: Fair Trade-Off Between Cost and QoE Solution

The Nash bargaining solution is the unique solution to a two-person bargaining problem that satisfies the axioms of scale invariance, symmetry, efficiency, and independence of irrelevant alternatives. The Nash bargaining game is a simple two-player game used to model bargaining interactions. In the Nash bargaining game, two players demand a portion of some good. If the total amount requested by the players is less than that available, both players get their request. If their total request is greater than what is available, neither player gets their request. In our model, we can model the two players as cost and QoE, they request the desired assignment to the appropriate VNF running on the appropriate VM of an adequate flavor. Our objective is to find the optimal assignment in order to satisfy all players. Since the utilities of the cost of steaming and the QoE are obviously conflicting, a Nash bargaining game is adopted in this paper and the Nash bargaining solution is considered as a reasonable solution to balance the utilities of both objectives in order to find a trade-off solution.

Indeed, our proposed FTS solution aims to find a fair trade-off between the conflicting objectives, the cost and the QoE. A bargaining game is used to find the fair trade-off between these conflicting objectives. The deployment of a high number of streaming VNFs in the network will enable getting streaming servers closer to the end-users, which consequently increases the QoE dramatically. Moreover, from another point of view, they will increase the cost of streaming services for the users significantly. However, the deployment of a low number of streaming VNFs will negatively impact the QoE and the end-to-end delay. In FTS, the cost and end-to-end delay are considered as two players that would like to barter goods.

1) Cooperative Games: In cooperative games, the players are assumed to attain either most desirable point when negotiation succeeds or disagreement point when negotiation fails. We consider two persons game who would like to barter goods, each one of them wants to increase his benefits. We define P as the vector payoffs of theses players. Formally, $\mathcal{P} = \{(u_1(x), u_2(x)), x = (x_1, x_2) \in X\}$, where X is the set of the two players' strategies. $u_1(x)$ and $u_2(x)$ represent the utility functions of two players, respectively. In [41], Nash bargaining model (NBS) is presented, which is a cooperative game with non-transferable utility. This means that the utility scales of the players are measured in non-comparable units. Nash bargaining game is based on two elements assumed to be given and known to the players. The first element is the set of vector payoffs $\mathcal{P}$ achieved by the players if they agree to cooperate. $\mathcal{P}$ should be a convex and compact set. The second element is the threat point, $d = (u_1^d, u_2^d) \in \mathcal{P}$, which represents the pair of utility whereby the two players fail to achieve an agreement. In NBS, we aim to find a fair and reasonable point, $(u_1^*, u_2^*) = f(\mathcal{P}, u_1^d, u_2^d) \in \mathcal{P}$. Based on Nash theory, a set of axioms is defined that leads to $f(\mathcal{P}, u_1^d, u_2^d)$ achieving a unique optimal solution (u_1^*, u_2^*):

1) **Feasibility:** $(u_1^*, u_2^*) \in \mathcal{P}$.
2) **Pareto Optimality:** There is no point $(u_1(x), u_2(x)) \in \mathcal{P}$ such that $u_1(x) \geq u_1^*$ and $u_2(x) \geq u_2^*$ except (u_1^*, u_2^*).
3) **Symmetric:** If $\mathcal{P}$ is symmetric about the line $u_1(x) = u_2(x)$, and $u_1^d = u_2^d$, then $u_1^* = u_2^*$.
4) **Independence of irrelevant alternatives:** If T is a closed convex subset of $\mathcal{P}$, and if $(u_1^d, u_2^d) \in T$ and $(u_1^*, u_2^*) \in T$, then $f(\mathcal{P}, u_1^d, u_2^d) = (u_1^*, u_2^*)$.
5) **Invariance under change of location and scale**: If $T = \{(u_1'(x), u_2'(x)), u_1'(x) = \alpha_1 \ u_1(x) + \beta_1, u_2'(x) = \alpha_2 \ u_2(x) + \beta_2, (u_1(x), u_2(x)) \in \mathcal{P}\}$ where $\alpha_1 \geq 0$, $\alpha_2 \geq 0$ and β_1 and β_2 are given numbers the $f(T, \alpha_1 \ u_1^d + \beta_1, \alpha_2 \ u_2^d + \beta_2) = (\alpha_1 \ u_1^* + \beta_1, \alpha_2 \ u_2^* + \beta_2)$.

Moreover, the unique solution $(\overline{u}, \overline{v})$, satisfying the above axioms, is proven to be the solution of the following optimization problem:

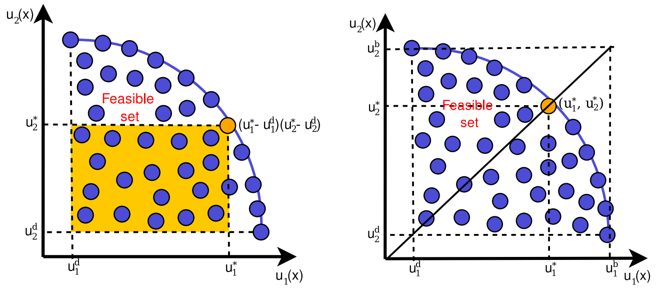
$$\begin{cases} \textbf{max} \ \ (u_1(x) - u_1^d)(u_2(x) - u_2^d) \\ \textbf{s. t.} \\ (u_1(x), u_2(x)) \in \mathcal{P} \\ (u_1(x), u_2(x)) \geq (u_1^d, u_2^d) \end{cases} \tag{7}$$

An enhanced solution of Nash bargaining game, named KSBS, is proposed by Kalai and Smorodinsky [42]. KSBS aims to enhance more the fairness between the players by sharing the same utility fraction r among them. KSBS preserves the same Nash bargaining axioms except the independence of irrelevant alternatives. In addition, it has new axioms called monotonically. In contrast to the Nash bargaining game, in addition to the disagreement point $d = (u_1^d, u_2^d) \in \mathcal{P}$, KSBS needs the ideal point for both players $x^b = (u_1^b, u_1^b)/x^b \in \mathcal{P}$, which is the best utility that both players can achieve separately without bargaining. Kalai and Smorodinsky proves that the unique solution that satisfies KSBS's axioms is the solution of the following optimization problem:

$$\begin{cases} \textbf{max} \ \ r \\ \textbf{s. t.} \\ (u_1(x), u_2(x)) \in \mathcal{P} \\ r = \dfrac{u_1(x) - u_1^d}{u_1^b - u_1^d} \\ r = \dfrac{u_2(x) - u_2^d}{u_2^b - u_2^d} \end{cases} \tag{8}$$

Fig. 5 shows how the KSBS game enhances NBS in terms of fairness and Pareto-optimality. As shown in Fig. 5(a), NSB aims to increase as much as possible the size of the orange rectangle. However, this strategy may favor one player over the other. As shown in Fig. 5(b), KSBS enhances the trade-off between the two players by sharing the same utility fraction. To increase the benefit of each player, KSBS increases as much as possible the fraction r.

$$r = \frac{u_1^* - u_1^d}{u_1^b - u_1^d} = \frac{u_2^* - u_2^d}{u_2^b - u_2^d}$$



(a) Nash bargaining solution.

(b) Kalai and Smorodinsky bargaining solution.

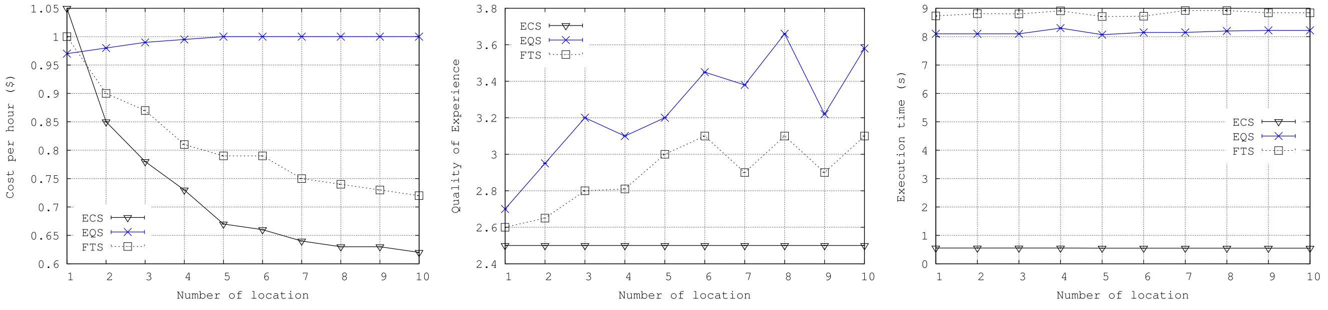
Fig. 5. NBS and KSBS solutions.

2) FTS Description: In the remainder of this section, we will describe the FTS solution (i.e., Fair Trade-off between cost and QoE Solution). We denote by $d = (u_C^d, u_Q^d)$ and $b = (u_C^b, u_Q^b)$ the threat and best points of the KSBS game that solves FTS. In the KSBS game, both players (i.e., ECS and EQS) should bargain for increasing their benefits. However, from the optimization (4), ECS aims to reduce the cost, which is the opposite to its utility function. In order to use the KSBS game for ensuring a fair trade-off between the QoE and the cost, as depicted in Fig. 5(b), we need to change the utility function of ECS to be a maximization problem. The utility function of the ECS player is then updated using the following optimization problem:

$$\begin{cases} \textbf{max} \ \ -C_{TOTAL}(S) \\ \textbf{s. t.} \\ j = \mathcal{C} : \displaystyle\sum_{k \in \mathcal{V}_1} \sum_{i \in F} h^{-1}(i, \rho) S_{i,j}^k \mathcal{P}_{i,k} \geq Y_{TOTAL} \\ j = \mathcal{T} : \displaystyle\sum_{k \in \mathcal{V}_1} \sum_{i \in F} g^{-1}(i, \sigma) S_{i,j}^k \mathcal{P}_{i,k} \geq Y_{TOTAL} \\ j = \mathcal{S}, l \in \mathcal{V}_2 : \displaystyle\sum_{k \in \mathcal{V}_1} \sum_{i \in F} q^{-1}(i, \mu, \lambda_{k,l}) S_{i,j}^k \mathcal{P}_{i,k} \geq M_l \\ \forall i \in F, \forall j \in E, \forall k \in \mathcal{V}_1 : S_{i,j}^k \in \mathbb{N} \\ \forall i \in F, \forall k \in \mathcal{V}_1 : \mathcal{P}_{i,k} \in \{0, 1\} \\ Y_{TOTAL}, M_l, \rho, \mu, \sigma > 0 \end{cases} \tag{9}$$

Now, we introduce the proposed FTS solution. In what follows, we will show how d and b would be computed. Let $\dot{\mathcal{S}}$ and $\ddot{\mathcal{S}}$ denote the two matrices of $S_{i,j}^k$ variables obtained by solving the two optimizations (6) and (10), respectively. Then, $d = (u_C^d, u_Q^d)$ and $b = (u_C^b, u_Q^b)$ would be computed as follows:

1) $u_C^d = -C_{TOTAL}(\dot{\mathcal{S}}) = -\displaystyle\sum_{i \in F} \sum_{j \in E} \sum_{k \in \mathcal{V}_1} \dot{\mathcal{S}}_{i,j}^k \mathcal{P}_{i,k} \times c_i$
2) $u_C^b = -C_{TOTAL}(\ddot{\mathcal{S}}) = -\displaystyle\sum_{i \in F} \sum_{j \in E} \sum_{k \in \mathcal{V}_1} \ddot{\mathcal{S}}_{i,j}^k \mathcal{P}_{i,k} \times c_i$
3) $u_Q^d = \displaystyle\min_{l \in \mathcal{V}_2} Q_{TOTAL}(\ddot{\mathcal{S}}, \mathcal{N}, l) = \min_{l \in \mathcal{V}_2} \sum_{i \in F} \sum_{k \in \mathcal{V}_1} \sum_{n \in M_l} q(i, n, \lambda_{k,l}) \times \ddot{\mathcal{S}}_{i,j}^k \times \mathcal{P}_{i,k} \times \mathcal{N}_{i,n}$
4) $u_Q^b = \displaystyle\min_{l \in \mathcal{V}_2} Q_{TOTAL}(\dot{\mathcal{S}}, \mathcal{N}, l) = \min_{l \in \mathcal{V}_2} \sum_{i \in F} \sum_{k \in \mathcal{V}_1} \sum_{n \in M_l} q(i, n, \lambda_{k,l}) \times \dot{\mathcal{S}}_{i,j}^k \times \mathcal{P}_{i,k} \times \mathcal{N}_{i,n}$

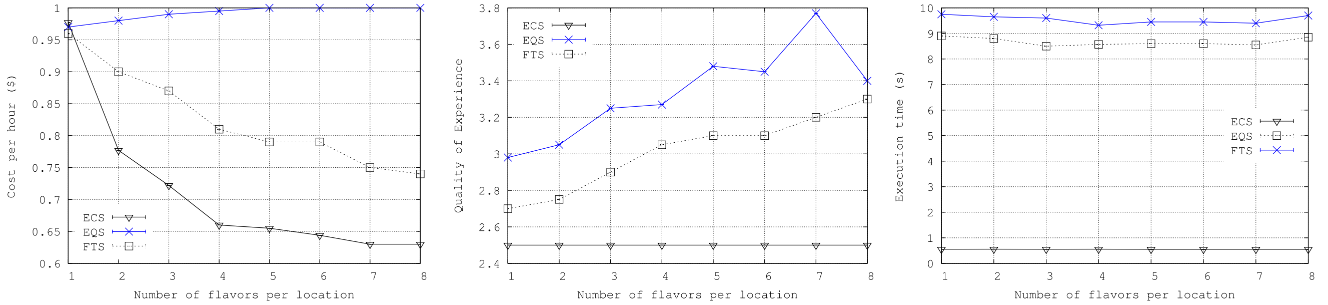


(a) The cost of virtual network functions.

(b) The QoE of the streaming service.

(c) The operation time.

Fig. 6. Performance evaluation by increasing the number of locations.



(a) The cost of virtual network functions.

(b) The QoE of the streaming service.

(c) The operation time.

Fig. 7. Performance evaluation by increasing the number of flavors per location.

The fair Pareto optimal solution FTS will be then formulated as follows:

$$\left\{\begin{array}{l}
\textbf{max} \quad r \\
\textbf{s. t.} \\
\sum_{k\in\mathcal{V}_1}\sum_{i\in F}\sum_{j\in K} S_{i,j}^{k} \times \mathcal{P}_{i,k} \times c_i \leq Cost_{Max} \\
j=\mathcal{C}: \sum_{k\in\mathcal{V}_1}\sum_{i\in F} h^{-1}(i,\rho) \times S_{i,j}^{k} \times \mathcal{P}_{i,k} \geq Y_{TOTAL} \\
j=\mathcal{T}: \sum_{k\in\mathcal{V}_1}\sum_{i\in F} g^{-1}(i,\sigma) \times S_{i,j}^{k} \times \mathcal{P}_{i,k} \geq Y_{TOTAL} \\
j=\mathcal{S}, l\in\mathcal{V}_2: \sum_{k\in\mathcal{V}_1}\sum_{i\in F}\sum_{n\in M_l} n \times S_{i,j}^{k} \times \mathcal{P}_{i,k} \times \mathcal{N}_{i,n} \geq |M_l| \\
j=\mathcal{S}, l\in\mathcal{V}_2: \sum_{k\in\mathcal{V}_1}\sum_{i\in F} q^{-1}(i,\mu,\lambda_{k,l}) \times S_{i,j}^{k} \times \mathcal{P}_{i,k} \geq |M_l| \\
u_C(S) = -\sum_{i\in F}\sum_{j\in E}\sum_{k\in\mathcal{V}_1} S_{i,j}^{k} \mathcal{P}_{i,k} \times c_i \\
\forall l \in \mathcal{V}_2: u_Q(S) \geq \sum_{i\in F}\sum_{k\in\mathcal{V}_1}\sum_{n\in M_l} q(i,n,\lambda_{k,l}) \\
\qquad\qquad \times S_{i,j}^{k} \times \mathcal{P}_{i,k} \times \mathcal{N}_{i,n} \\
r = \dfrac{u_C(S) - u_C^d}{u_C^b - u_C^d} \\
r = \dfrac{u_Q(S) - u_Q^d}{u_Q^b - u_Q^d} \\
\forall i \in F, \forall j \in E, \forall k \in \mathcal{V}_1 : S_{i,j}^{k} \in \mathbb{N} \\
\forall i \in F, \forall k \in \mathcal{V}_1 : \mathcal{P}_{i,k} \in \{0,1\} \\
Y_{TOTAL}, M_l, \rho, \mu, \sigma > 0 \\
\forall i \in F, \forall n \in M_l : \mathcal{N}_{i,n} \in \{0,1\}
\end{array}\right. \tag{10}$$

Then, the minimum QoE in the network $\mathcal{Q}_{\mathcal{F}}$ and the cost $\mathcal{C}_{\mathcal{F}}$ of FTC will be computed as follows:

$$\mathcal{Q}_{\mathcal{F}} = u_Q(S) \tag{11}$$

$$\mathcal{C}_{\mathcal{F}} = -u_C(S) \tag{12}$$

VI. 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 the Gurobi optimization tool and are evaluated using the following metrics: *i)* the paid cost of VNFs; *ii)* the QoE of the streaming service; and *iii)* the operation time. 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 defined after examining the prices of 87 flavors offered by Amazone AWS service, Microsoft Azure, and Rackspace. 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. 6 and 7 show the performance evaluation of the proposed solutions in terms of the number of flavors and the number of data centers' locations. In case of the ECS and FTS algorithms, the minimum acceptable QoE of the streaming service μ is set to 2.5, while in case of the EQS and FTS algorithms, $Cost_{Max}$ is set to 1$. In the two figures, the QoE value in case of the EQS and FTS algorithms is presented as the average of the QoE values of all VMs hosting a streamer image as VNF but not by Q_{TOTAL}.

Figs. 6(a) and 7(a) show the total cost incurred by the instantiated VNFs when varying the number of locations and 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. 7(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. From Figs. 6(a) and 7(a), FTS exhibit a performance between FCS and QCS in terms of cost. We also observe from these figures that the curve of FCS has the same trend as the one of ECS.

Figs. 6(b) and 7(b) show the QoE of the streaming service while varying the number of data centers' locations and the number of flavors per location, respectively. The first observation that we can draw from these figures is that the EQS scheme exhibits the best performance in terms of QoE in comparison to ECS and FTS. This is due to the fact that EQS is designed to optimize the QoE without taking into account the cost overhead. From these figures, we observe that FTS has a performance similar to EQS in terms of QoE. Mainly, the curve of FTS has the same trend as the one of EQS. As depicted in these two figures, the QoE of the streaming service in case of the EQS and FTS algorithms 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. We also observe from these figures that the numbers of locations and flavors do not have an impact on the QoE for the ECS solution. This is attributable to the fact that the ECS solution aims at reducing the cost without taking the QoE into account, and then the worst QoE values would be achieved by the ECS solution. From the simulation, it seems that the worst QoE value that was achieved in the network is 2.5.

In Figs. 6(c) and 7(c), the execution time of the three solutions is presented. As observed from these figures, the ECS algorithm exhibits better performance than the EQS and FTS algorithms in terms of execution time, regardless the number of data centers and the number of flavors per location. From Figs. 6(a) and 7(a), from one side, and Figs. 6(b) and 7(b), from another side, we conclude that the FTS solution achieves a fair trade-off between QoE and cost. FTS performs similarly to ECS in terms of the cost, and similarly to EQS in terms of the QoE.

VII. Conclusion

In this paper, we introduced a CDNaaS platform that allows the creation of CDN slices across multiple cloud domains. The platform is able to scale out and down by deploying virtual resources from multiple IaaS providers, running different VNFs (i.e., virtual caches, virtual transcoders, and virtual streamers). In order to create a cost-efficient and QoE-aware virtual CDN slice, the optimal placement of these VNFs, along with decision on the amount of virtual resources to allocate for each of them, is of vital placement. In this vein, this paper introduced three relevant solutions. The first solution aims at minimizing the incurred total cost, while the second solution aims at maximizing QoE of the streaming services. The third solution uses a bargaining game theory for ensuring a fair trade-off between the cost and QoE. A mathematical model is developed to evaluate the performance of these three solutions. Simulations were conducted and the obtained results demonstrated the efficiency of the proposed solutions in achieving their key design goals.

References

[1] I. Benkacem, T. Taleb, M. Bagaa, and H. Flinck, "Performance benchmark of transcoding as a virtual network function in CDN as a service slicing," in *Proc. IEEE WCNC*, Barcelona, Spain, Apr. 2018, pp. 1–6.

[2] S. Retal, M. Bagaa, T. Taleb, and H. Flinck, "Content delivery network slicing: QoE and cost awareness," in *Proc. IEEE ICC*, May 2017, pp. 1–6.

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

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

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

[6] S. Dutta, T. Taleb, and A. Ksentini, "QoE-aware elasticity support in cloud-native 5G systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

[7] 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.

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

[9] F. B. Jemaa, G. Pujolle, and M. Pariente, "Qos-aware VNF placement optimization in edge-central carrier cloud architecture," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016, pp. 1–7.

[10] D. B. Oljira, K.-J. Grinnemo, J. Taheri, and A. Brunstrom, "A model for QoS-aware VNF placement and provisioning," in *Proc. IEEE Conf. (NFV-SDN)*, Berlin, Germany, Nov. 2017, pp. 1–7.

[11] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, and R. Boutaba, "Delay-aware VNF placement and chaining based on a flexible resource allocation approach," in *Proc. 13th Int. Conf. Netw. Service Manage. (CNSM)*, Tokyo, Japan, Nov. 2017, pp. 1–7.

[12] M. Abu-Lebdeh, D. Naboulsi, R. H. Glitho, and C. W. Tchouati, "On the placement of VNF managers in large-scale and distributed NFV systems," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 4, pp. 875–889, Dec. 2017.

[13] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[14] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies and solutions," *IEEE Commun. Surveys Tuts.*, to be published.

[15] *Description of Network Slicing Concept*, NGMN Alliance, San Diego, CA, USA, Jan. 2016.

[16] K. Samdanis and X. Costa-Pérez, and V. Sciancalepore, “From network sharing to multi-tenancy: The 5G network slice broker,” *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 32–39, May 2016.

[17] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Muñoz, J. Lorca, and J. Folgueira, “Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges,” *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 80–87, May 2017.

[18] R. Peter *et al.*, “Network slicing to enable scalability and flexibility in 5G mobile networks,” *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 72–79, May 2017.

[19] B. Tan, J. Wu, Y. Li, H. Cui, W. Yu, and C. W. Chen, “Analog coded SoftCast: A network slice design for multimedia broadcast/multicast,” *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2293–2306, Oct. 2017.

[20] A. Nakao *et al.*, “End-to-End network slicing for 5G mobile networks,” *J. Inf. Process.*, vol. 25, pp. 153–163, Feb. 2017. [Online]. Available: https://www.jstage.jst.go.jp/article/ipsjjip/25/0/25_153/_article

[21] T. Taleb, B. Mada, M. I. Corici, A. Nakao, and H. Flinck, “PERMIT: Network slicing for personalized 5G mobile telecommunications,” *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 88–93, May 2017.

[22] H. Zhang, N. Liu, X. Chu, K. Long, A. Aghvami, and V. C. M. Leung, “Network slicing based 5G and future mobile networks: Mobility, resource management, and challenges,” *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 138–145, Sep. 2017.

[23] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby, “On the optimal placement of Web proxies in the Internet,” in *Proc. IEEE INFOCOM*, vol. 3. New York, NY, USA, Mar. 1999, pp. 1282–1290.

[24] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, “On the placement of Web server replicas,” in *Proc. IEEE INFOCOM*, Anchorage, AK, USA, Apr. 2001, pp. 1587–1596.

[25] Y. Chen, R. H. Katz, and J. Kubiatowicz, “Dynamic replica placement for scalable content delivery,” in *Proc. 1st Int. Workshop Peer-to-Peer Syst. (IPTPS)*, Cambridge, MA, USA, Mar. 2002, pp. 306–318.

[26] T. Taleb, “Toward carrier cloud: Potential, challenges, and solutions,” *IEEE Wireless Commun.*, vol. 21, no. 3, pp. 80–91, Jun. 2014.

[27] T. Taleb *et al.*, “Ease: Epc as a service to ease mobile core network deployment over cloud,” *IEEE Netw.*, vol. 29, no. 2, pp. 78–88, Mar. 2015.

[28] T. Taleb and A. Ksentini, “Gateway relocation avoidance-aware network function placement in carrier cloud,” in *Proc. 16th ACM Int. Conf. Modeling, Anal. Simulation Wireless Mobile Syst.*, Barcelona, Spain, Nov. 2013, pp. 341–346.

[29] M. Bagaa, T. Taleb, and A. Ksentini, “Service-aware network function placement for efficient traffic handling in carrier cloud,” in *Proc. IEEE Wireless Commun. Netw. Conf.*, Istanbul, Turkey, Apr. 2014, pp. 2402–2407.

[30] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, “The dynamic placement of virtual network functions,” in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–9.

[31] T. Taleb, M. Bagaa, and A. Ksentini, “User mobility-aware virtual network function placement for virtual 5G network infrastructure,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 3879–3884.

[32] J. Zhang, F. Dong, D. Shen, and J. Luo, “Game theory based dynamic resource allocation for hybrid environment with cloud and big data application,” in *Proc. IEEE Int. Conf. Syst., Man, Cybern., (SMC)*, San Diego, CA, USA, Oct. 2014, pp. 1128–1133.

[33] G. N. Iyer and B. Veeravalli, “On the resource allocation and pricing strategies in compute clouds using bargaining approaches,” in *Proc. 17th IEEE Int. Conf. Netw. (ICON)*, Singapore, Dec. 2011, pp. 147–152.

[34] A. Baharlouei and B. Jabbari, “A dynamic resource allocation scheme using nash bargaining game for the uplink of multiuser OFDM systems,” in *Proc. 78th IEEE Veh. Technol. Conf. VTC-Fall*, Las Vegas, NV, USA, Sep. 2013, pp. 1–5.

[35] Z. Zheng, L. Song, D. Niyato, and Z. Han, “Resource allocation in wireless powered relay networks through a nash bargaining game,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

[36] G. Zhang, H. Zhang, L. Zhao, W. Wang, and L. Cong, “Fair resource sharing for cooperative relay networks using nash bargaining solutions,” *IEEE Commun. Lett.*, vol. 13, no. 6, pp. 381–383, Jun. 2009.

[37] T. Taleb and K. Hashimoto, “MS2: A novel multi-source mobile-streaming architecture,” *IEEE Trans. Broadcast.*, vol. 57, no. 3, pp. 662–673, Sep. 2011.

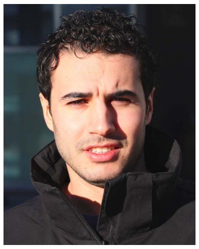
[38] F. Z. Yousaf and T. Taleb, “Fine-grained resource-aware virtual network function management for 5G carrier cloud,” *IEEE Netw.*, vol. 30, no. 2, pp. 110–115, Feb. 2016.

[39] A. Ksentini, T. Taleb, and K. B. Letaif, “Qoe-based flow admission control in small cell networks,” *IEEE Trans. Wireless Commun.*, vol. 15, no. 4, pp. 2474–2483, Nov. 2016.

[40] L. Koskimies, T. Taleb, and M. Bagaa, “QoE estimation-based server benchmarking for virtual video delivery platform,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.

[41] J. F. Nash, Jr., “The bargaining problem,” *Econometrica*, vol. 18, no. 2, pp. 155–162, 1950.

[42] E. Kalai and M. Smorodinsky, “Other solutions to Nash’s bargaining problem,” *Econometrica*, vol. 43, no. 3, pp. 513–518, May 1975.



Ilias Benkacem received the B.Sc. degree in Mathematics and Physics Higher School Preparatory Classes for Engineering Schools (CPGE), Tangier, Morocco, in 2013, and the Engineer’s degree from the National Superior School of Computer Science and System Analysis, Mohammed V University, Rabat, Morocco, in 2016. He is currently pursuing the Ph.D. degree with the School of Electrical Engineering, Aalto University, Finland. Since 2017, he has been involved in European Project 5G!Pagoda Horizon 2020 for a scalable 5G network slicing management and orchestration framework for distributed edge dominated network infrastructures. His current research interests include MEC, NFV, SDN, and content delivery in ICN/CDN networks, particularly in 5G networks.

Tarik Taleb received the B.E. degree (Hons.) in information engineering and the M.Sc. and Ph.D. degrees in information sciences from GSIS, Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively. He is currently a Professor with the School of Electrical Engineering, Aalto University, Espoo, Finland. He is a member of the IEEE Communications Society Standardization Program Development Board. In an attempt to bridge the gap between academia and industry, he founded the IEEE-Workshop on Telecommunications Standards: From Research to Standards, a successful event that was recognized with the BestWorkshop Award by the IEEE Communication Society (ComSoC). Based on the successof this workshop, he has also founded and has been the Steering Committee Chair of the IEEE Conference on Standards for Communications and Networking. He is the General Chair of the 2019 edition of the IEEE Wireless Communications and Networking Conference to be held in Marrakech, Morocco. He is /was on the Editorial Board of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the *IEEE Wireless Communications Magazine*, the IEEE JOURNAL ON INTERNET OF THINGS, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE COMMUNICATIONS SURVEYS & TUTORIALS, and a number of Wiley Journals. He is an IEEE Communications Society (ComSoc) Distinguished Lecturer.

Miloud Bagaa received the Engineer’s, master’s, and Ph.D. degrees from the University of Science and Technology Houari Boumediene, Algiers, Algeria, in 2005, 2008, and 2014, respectively. From 2009 to 2015, he was a Researcher with the Research Center on Scientific and Technical Information, Algiers. From 2015 to 2016, he was with the Norwegian University of Science and Technology, Trondheim, Norway. He is currently a Senior Researcher with Aalto University. His research interests include wireless sensor networks, Internet of Things, 5G wireless communication, security, and networking modeling. From 2015 to 2016, he received the Post-Doctoral Fellowship from the European Research Consortium for Informatics and Mathematics

Hannu Flinck received the M.Sc. and Lic.Tech. degrees in computer science and communication systems from the Helsinki University of Technology (Aalto University) in 1986 and 1993, respectively. He was with Nokia Research Center and the Technology and Innovation Unit of Nokia Networks in various positions. He is currently a Research Manager with Nokia Bell Labs, Espoo, Finland. He has been actively participating in a number of EU research projects. His current research topics include 5G network architecture, IP protocols, and virtualization.