Coalitional Game for the Creation of Efficient Virtual Core Network Slices in 5G Mobile Systems

Miloud Bagaa, Tarik Taleb, Abdelquoddouss Laghrissi, Adlen Ksentini, and Hannu Flinck

***Abstract*—Many ongoing research activities relevant to 5G mobile systems concern the virtualization of the mobile core network, including the Evolved Packet Core (EPC) elements, aiming for system scalability, elasticity, flexibility, and cost-efficiency. Virtual EPC (vEPC)/5G Core will principally rely on some key technologies such as Network Function Virtualization (NFV), Software Defined Networking (SDN) and Cloud Computing; enabling the concept of Mobile Carrier Cloud. The key idea beneath this concept, known also as Core Network as a Service (CNaaS), consists in deploying virtual instances (i.e., Virtual Machines or containers) of key core network functions (i.e., Virtual Network Functions - VNF of 4G or 5G) such as the Mobility Management Entity (MME), Serving GateWay (SGW), Packet Data network gateWay (PGW), Access and Mobility Management Function (AMF), Session Management Function (SMF), Authentication Server Function (AUSF), and User Plane Functions (UPF) over a federated cloud. In this vein, an efficient VNF placement algorithm is highly needed to sustain Quality of Service (QoS) while reducing the deployment cost. Our contribution in this paper is two fold. First, we devise an algorithm that derives the optimal number of virtual instances of 4G (MME, SGW, PGW) or 5G (AMF, SMF and AUSF) core network elements to meet the requirements of a specific mobile traffic. Second, we propose an algorithm for the placement of these virtual instances over a federated cloud. While the first algorithm is based on Mixed Integer Linear Programming (MILP), the second is based on Coalition formation game, wherein the aim is to build coalitions of Cloud Networks ($\mathcal{CN}$s) to host the virtual instances of the vEPC/5G Core elements. The obtained results clearly indicate the advantages of the proposed algorithms in ensuring QoS given a fixed cost for vEPC/5G Core deployment, while maximizing the profits of cloud operators.**

***Index Terms*—5G, Core Network, Network Slicing, Evolved Packet Core (EPC), VNF placement, Network Function Virtualization (NFV), Management and Orchestration (MANO), Coalitional Game, and Game Theory.**

I. Introduction

Network Function Virtualization (NFV) represents one of the key enablers of the next generation mobile network systems (5G). NFV allows running Virtual Network Functions (VNF) as software components on top of a virtualization system (i.e., Virtual Machines - VMs - or Containers) hosted in a cloud; allowing high flexibility and elasticity to deploy network services and functions. Using NFV, different mobile network components will be virtualized and that is spanning the Radio Access Network (RAN) and the core network (e.g., Evolved Packet Core – EPC – in case of 4G mobile system) [1]. RAN components will be divided into Base Band Unit (BBU) and Remote Radio Head (RRH), whereby BBU is run as a software and RRH will be kept deployed in the field. 5G RAN is going further by splitting the BBU into two entities, namely Central Unit (CU) and Distributed Unit (DU); CU hosts time tolerant RAN functions, while DU hosts time critical functions, such as MAC and part of the physical functions. On the other hand, EPC components (i.e., Mobility Management Entity - MME, Home Subscriber Sub-system - HSS, Serving Gateway - SGW, and Packet Data Network Gateway - PGW) will be fully virtualized and hosted in federated cloud; building the concept of Mobile Carrier Cloud [2]. The same applies for the main components of the 5G Core, such as the Access and Mobility Management Function (AMF), Session Management Function (SMF), Authentication Server Function (AUSF), and User Plane Functions (UPF). The virtual instances of these components will then constitute the virtual EPC/5G Core.

To structure R&D activities around NFV, ETSI [3] and 3GPP [4] have launched standardization activities on NFV, which has defined a global architecture to ensure orchestration and management of services. The proposed architectures [3], [4] define modules and interfaces that ensure the life-cycle management of a service; from definition, composition to deployment in the cloud infrastructure. In the envisioned architectures, vEPC/5G Core will represent nothing but a simple mobile service. In this case, the mobile network operator communicates its needs, in terms of traffic patterns to handle, location to cover, etc., to the NFVO (NFV orchestrator) (resp., Network Slice Management Function (NSMF)) through dedicated northbound interfaces (e.g. REST API). The NFVO/NSMF translates these needs into a number of software instances (i.e. number of VMs hosting AMF, PCF and UPF), which should be deployed in the network. To calculate the number of virtual instances needed and their placement over the federated cloud, the NFVO/NSMF has to rely on VNF placement algorithms that output a Service Instance Graph (SIG), which is a meta-data file that indicates how the VM instances should be interconnected and where they should run. Then, the NFVO/NSMF gathers the VM image from a software catalog, which stores all software images of

Manuscript received September 29, 2017; revised February 7, 2018; accepted February 27, 2018.

An abridged version of this paper has been published in the proceedings of the 2018 edition of the IEEE Wireless Communications and Networking Conference [26].

Miloud Bagaa, Tarik Taleb, and Abdelquoddous Laghrissi are with the Department of Communications and Networking, School of Electrical Engineering, Aalto University, Finland (e-mails: {firstname.lastname}@aalto.fi). Tarik Taleb is also with the Department of Computer and Information Security, Sejong University, Seoul, South Korea.

Adlen Ksentini is with Communication systems, EURECOM, Sophia-Antipolis, France (e-mail: adlen.ksentini@eurecom.fr).

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

AMF, PCF, MME and HSS, and then enforces the SIG in the federated cloud using a Network Slice Subnet Management Function (NSSMF) and Virtual Infrastructure Manager (VIM). Whilst many solutions are currently available for the NFVO [5], VIM [6] and VNF Manager (VNFM) [7], decisions on the optimal number of VM instances needed and their placement over the federated cloud are still overlooked.

In this paper, we fill this gap by proposing a novel solution that addresses both the optimal number of VM instances to instantiate and their placement over a federated cloud to create a vEPC/5G Core[1] for a specific traffic pattern. Let ϑ denote a VNF, whereby a set of different ϑs would compose the vEPC/5G Core instance. Formally, ϑ can be one of the following network elements: HSS, MME, SGW, PGW, AMF, SMF, AUSF, and UPF. Obviously, the proposed solution consists of two main parts. The first part derives the optimal number of VM instances of each VNF ϑ (i.e. HSS, MME, SGW and PGW), to handle a target traffic of mobile users. In this part, the problem is formulated using Mixed Integer Linear Program (ILP). The second part consists in placing the instantiated VNFs in the federated cloud, i.e. indicating on which cloud network $\mathcal{CN}$ a VNF should run. Here, we formulate the problem using Game Theory, and specifically Coalition Formation Game. Unlike the existing solutions, which assume that $\mathcal{CN}$s belong to the same cloud operator, in this work we relax this constraint by allowing that $\mathcal{CN}$s could belong to different cloud providers. The proposed placement algorithm considers the different $\mathcal{CN}$s as players and assumes that it is better for them to cooperate by building coalition rather than not cooperate. Indeed, a $\mathcal{CN}$ would decide to participate in a coalition (i.e., the creation of a set of instances of a VNF ϑ) only if its profit is improved. The profit of a $\mathcal{CN}$ refers to the difference between the price that the mobile operator is willing to pay and the cost needed to handle the traffic generated from different Tracking Areas (TAs) associated to this $\mathcal{CN}$. Note that the mentioned cost may include the one incurred by resources dedicated to a VM (i.e., CPU and storage) and the management of VMs (i.e., migration). Indeed, migrating a VM from a $\mathcal{CN}$ to another, in order to free space to host new VMs, could represent high cost. Therefore, it is better for a $\mathcal{CN}$ to participate to a coalition and host part of the VNF, instead of hosting all the instances of VNFs ϑ but with low profit. Moreover, in this game, for each VNF (e.g., MME or SGW and PGW), $\mathcal{CN}$s are selfish and work to maximize only their individual profits without looking into the profit of the other $\mathcal{CN}$s.

The remainder of this paper is organized in the following fashion. Section II presents some related research work. The network model and problem formulation are covered in Section III. The proposed VNF placement strategy is introduced in Sections IV and V. Section VI evaluates the performance of the different optimization solutions envisioned in this paper. The paper concludes in Section VII.

[1]It shall be noted that whilst the focus, throughout this paper, is on virtual EPC or EPCaaS (i.e., EPC VNFs), the proposed solutions are also applicable to the case of 5G core network functions.

II. Related work

The concept of carrier cloud (i.e., vEPC) assists in achieving elasticity, flexibility, and significant reduction in the operational cost of the overall system. Indeed, using NFV and general-purpose hardware in $\mathcal{CN}$s to run network functions helps in dynamically scaling up/down the network according to the demands of users for resources and can largely reduce the cost. NFV aims at offering diverse network services using network functions implemented in the format of a software, deployable in an on-demand and elastic manner on the cloud. In return of its numerous advantages, virtualizing EPC and 5G Core network functions introduces some important challenges, mainly related to the placement of the telco-specific VNFs (i.e. MME, PGW, SGW, SMF, and AUSF) over a federated cloud to ensure optimal connectivity for users and simultaneously reduce the deployment cost.

The problem of VNF placement is similar, in a sense, to the VM placement in the cloud. With regard to the latter, a large library of research work has been conducted for the placement of VMs (not necessarily hosting a VNF) in the same $\mathcal{CN}$ or across multiple $\mathcal{CN}$s (i.e. federated Cloud). These research work tackle the problem from different angles, considering specific criteria and constraints related to cost, availability, and performance, which pertain to both the network and the cloud infrastructure. For instance, the research work in [8] proposes an approach for VM placement and migration to minimize the time consumed in data transfers. The authors in [9] focus on mapping VMs to physical servers with the aim of improving server resource (e.g., CPU or memory) utilization, overcoming the lack of space in $\mathcal{CN}$s and maximizing the number of mapped VMs in one physical host. In [10], performance isolation (e.g., CPU, memory, storage, and network bandwidth), resource contention properties (among VMs on the same physical host), and VMs behavioral usage patterns are taken into account in decisions on VM placement, VM migration, and cloud resource allocations. Usually, a $\mathcal{CN}$ may start with an initial configuration and then apply adequate solutions to make a series of live migrations to transit the $\mathcal{CN}$ from a sub-optimal state to an optimal one, similar in fashion to solving an iterative rearrangement problem. For this purpose, different algorithms can be used, such as N-dimensional set or bin packing [11], the simulated annealing algorithms [12], and ant colony optimization [13]. In other research work, optimal placement of VMs, running specific services, on physical machines, consider electricity-related cost as well as transient cooling effects [14]. Other research work do autonomic placement of VMs as per policies specific by the $\mathcal{CN}$ providers and/or users [15]. Other VM placement strategies consider maximizing the profit under a particular service level agreement (SLA) and a predetermined power budget [16]. Authors in [17] proposed a solution for solving the problem of cloud federation formation. This solution aims to increase the benefit for different clouds by sharing the resources among them. Similarly to [17], authors in [18] proposed a solution for cloud federation formation that uses coalitional game. This solution explores the strength of cloud federation for ensuring the services availability, responding to

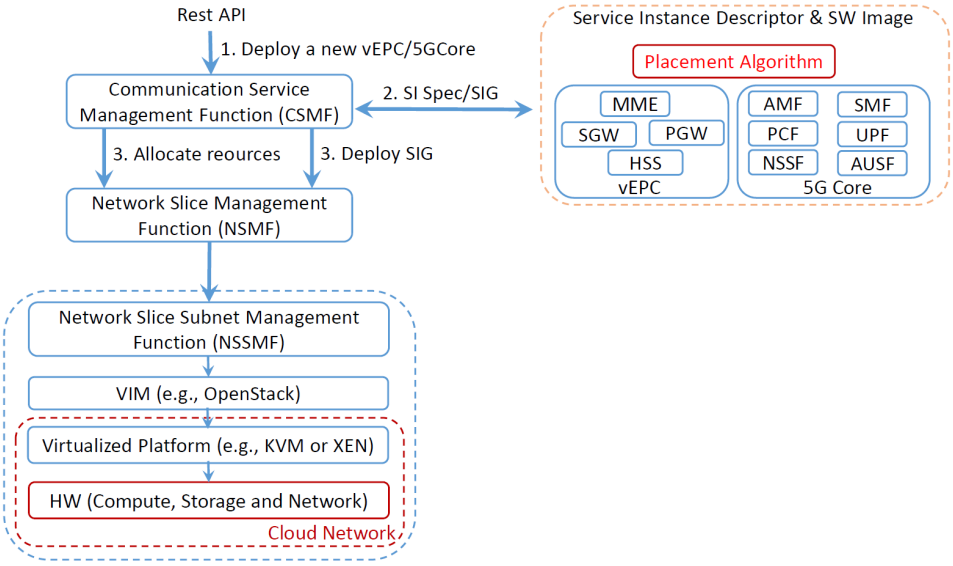


Fig. 1. NFV-based vEPC/5G Core service instantiation.

different users' requests. A solution that creates federations among a set of cloud providers is proposed in [19]. This solution aims to reduce the energy cost when forming a federated cloud. A cooperative game approach is used in that solution allowing different cloud providers to negotiate the resources in a distributed fashion.

More recently, new research work has emerged, proposing algorithms for the placement of vEPC's/5G Core's VNFs. In [20], the authors proposed a VNF placement method, particularly for placing mobility-anchor gateways (i.e. SGW) over a federated cloud so that the frequency of SGW relocation occurrences is minimized. The aim of this work was to conduct an efficient planning of Service Areas (SAs) retrieving a trade-off between minimizing the UE handoff between SAs, and minimizing the number of created instances of the virtual SGWs. In [21], the focus was on VNF placement and instantiation of another mobile network functionality, namely data anchoring gateway or PGW. The work argued the need for adopting application type and service requirements as metrics for (i) creating VNF instances of PGWs and (ii) selecting adequate virtual PGWs for UEs receiving specific application types. The placement of PGW VNFs was modeled through a nonlinear optimization problem whose solution is NP-hard. Three heuristics were then proposed to deal with this limitation. In [22], the authors proposed a framework, dubbed softEPC, for flexible and dynamic instantiation of VNFs where most appropriate and as per the actual traffic demand. The proposed scheme addresses load-aware dynamic placement of SGW/PGW over the underlying transport network topology and as per the traffic demands. The results show that up to 25% of network resources can be saved with same network topology and service points.

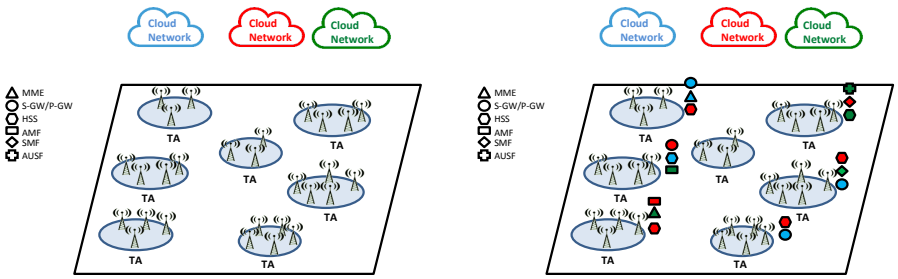
As aforementioned, different research work [17]–[19] aim to share the resources among different cloud networks by forming federated cloud. These solutions aim to ensure the stability of the grand coalition, formed by different clouds. The aim of the grand coalition is: $i)$ to share the virtual resources (e.g., RAM, virtual CPU, storage), and $ii)$ to ensure the service availability for responding to different users' requests. In contrast to these research works, the proposed solution, herein, aims to create an EPCaaS slice on top of different $\mathcal{CN}$s. The proposed solution specifies the number and locations of different instances of VNFs of virtual EPC, in each $\mathcal{CN}$. Moreover, in contrast to the literature, the proposed solution also specifies the flavor, the amount of virtual resources (i.e, number of Virtual cores - CPU, memory, storage) that should be dedicated for each instance of each VNF ϑ, in a fashion that reduces the cost while ensuring the QoS.

Unlike the cited research work [8]–[10], [14]–[22], which tackle either the optimal number of VNFs or the VNF placement, our proposed solution jointly addresses both issues in the same time. Moreover, these research work assume that $\mathcal{CN}$s belong to the same cloud operator, which is relaxed in our proposed solution (i.e., $\mathcal{CN}$s may belong to different cloud operators). To sum up, the proposed framework aims at finding: $i)$ the optimal number of VNFs to deploy (according to mobile traffic); $ii)$ the optimal placement of the determined VNFs over the underlying federated cloud.

III. Problem Formulation and Network Model

A. Problem formulation

3GPP [4] has defined a reference architecture for enabling NFV orchestration and VNF management in an efficient manner. A simplified view of the NFV architecture, featuring EPCaaS and potentially virtual 5G core, is depicted in Fig. 1. In this figure, the NFV architecture consists of three main parts: $i)$ $\mathcal{CN}$s whereby each $\mathcal{CN}$ is managed by a Virtual Infrastructure Manager VIM (e.g., OpenStack), $ii)$ Network Slice Management Function (NSMF) and Network Slice Subnet Management Function (NSSMF) that are responsible for managing, monitoring and orchestrating all VNFs in different $\mathcal{CN}$s, and $iii)$ a Communication Service Management Function (CSMF) that is responsible for: $a)$ translating the



(a) Network before deployment of vEPCs.

(b) Network after deployment of vEPCs.

Fig. 2. NFV-based vEPC service instantiation.

communication service related requirements into network slice related requirements, and b) communication with NSFM.

For enabling the VIM functionality, $\mathcal{CN}$s run different virtualization technologies (e.g., KVM, XEN or Containers) that allow the management of different virtual resources on top of hardware resources (e.g., Compute, Storage, and Network). VIM allows the instantiation of different Virtual Machines (VMs) with different virtual resources using pre-stored VM images. Different resources in a $\mathcal{CN}$ are defined through a set of flavors $\mathcal{L}^{\mathcal{CN}}$, whereby each flavor $\ell \in \mathcal{L}^{\mathcal{CN}}$ represents the amount of virtual resources (i.e., number of Virtual cores - CPU, memory, and storage) that would be dedicated to a specific VM in $\mathcal{CN}$. The Service Instance Manager (SIM) enables NSFM and NSSFM to orchestrate, monitor and manage a set of VNFs that belong to different $\mathcal{CN}$s. The description and behaviors of different components are defined in the Service Instance Descriptor (SID) that defines the Service Instance Graph (SIG), presented through two catalogues: i) VNF Catalogue, and ii) NFV Service Catalogue.

In this paper, we propose a virtual-EPC framework (i.e., easily extensible to cover the case of virtual 5G Core) that explores the flexibility of the NFV architecture to build the concept of EPCaaS over a federated cloud architecture. Our proposed algorithms can be run by the NFV orchestrator to find the number of VNFs and determine their placement over the federated cloud; building the SIG to be enforced by the VIM. The proposed VNF placement algorithm, in this paper, is located in the Service Instance Descriptor, and can be called by the CSMF using the operator inputs as entries. Note that the SIG will be created using the two devised algorithms as detailed in the following sections.

In this work, we consider that RAN consists of a number of eNodeBs, organized in Tracking Areas (TAs), as per Release 8 of the 3GPP mobile network specifications. We also assume that MME keeps record of the locations of *UEs* in idle mode at the *TA* granularity [24], [25]. As per Release 8 and beyond of the 3GPP mobile network specifications, S1-MME and S1-U interfaces are changed by the S1-Flex interface that allows each TA to be serviced by multiple MMEs and SGWs within a pool area. The set of TAs, served by the same MME/SGW node, forms an MME/SGW pool/service area, respectively. Formally, a MME pool area is defined as a set of TAs where a UE may be served without the need to change the serving MME. MME pool areas may overlap with each other [27]. On the other side, SGW service area is also defined as a set of TAs where a UE does need to change its SGW while moving within the same service area. SGW Service Areas (i.e., SA) may also overlap with each other. Knowing that the traffic generated by UEs (i.e., at both control and data plane levels) could be aggregated at the TA level, the first part of the virtual-EPC framework consists in devising algorithms that compute the optimal number of instances to deploy for each VNF ϑ (i.e. SSH, MME, SGW or PGW) to handle the expected mobile traffic. In addition, the algorithms should associate each TA with its respective MME/SGW pool.

TABLE I
NOTATIONS USED IN THE PAPER.

Notation	Description
$\mathcal{CN}$	A Cloud Network.
Υ	The set of events that can occur in a network.
Ω	The set of all tracking areas.
Σ	The set of all $\mathcal{CN}$s in the network.
$\mathcal{V}$	The set of VNFs that would be deployed to build a vEPC/5G Core. Formally, $\mathcal{V} = \{\text{MME, HSS, SGW, PGW}\}$.
$\vartheta \in \mathcal{V}$	A VNF that would be deployed to build a vEPC/5G Core. Formally, ϑ can be MME, HSS, SGW, or PGW.
$\mathcal{T}$	The time in discrete format, where each element $t \in \mathcal{T}$ represents the occurrence time of one or multiple events.
$\Gamma^x_{\mathcal{A}}$	An array that shows the number of cumulative events $x \in \Upsilon$ initiated from TA $\mathcal{A}$ during $\mathcal{T}$. For each $t \in \mathcal{T}$, $\Gamma^x_{\mathcal{A}}[t]$ represents the number of cumulative events $x \in \Upsilon$ initiated from TA $\mathcal{A}$ during the time t.
$\Phi^x_{\mathcal{A},\mathcal{B}}$	An array that shows the number of cumulative events $x \in \Upsilon$ that would be removed if TAs $\mathcal{A}$ and $\mathcal{B}$ are served by the same instance of VNF during $\mathcal{T}$. For each $t \in \mathcal{T}$, $\Phi^x_{\mathcal{A},\mathcal{B}}[t]$ represents the number of cumulative events $x \in \Upsilon$ that would be removed if TAs $\mathcal{A}$ and $\mathcal{B}$ are served by the same instance of VNF during the time t.
$\lambda_{x,\vartheta}$	The amount of CPU and memory resources required to handle one event of type $x \in \Upsilon$ by an instance of VNF $\vartheta \in \mathcal{V}$.
$\mathcal{L}^{\mathcal{CN}}$	The set of flavors in $\mathcal{CN}$.
$\mathcal{MAX}^{\mathcal{CN}}_{\ell}$	The maximum CPU and memory resources that can be offered by an instance of VNF with a flavor $\ell \in \mathcal{L}^{\mathcal{CN}}$.
$\tau^{\mathcal{CN}}_{\ell}$	The cost of one virtual instance of flavor type $\ell \in \mathcal{L}^{\mathcal{CN}}$.
$\mathcal{X}_{\delta,i}$	A decision boolean variable that shows if a subset of TAs $\delta \subseteq \omega \subseteq \Omega$ use the same instance i of a VNF ϑ.
$\mathcal{Y}_{i,\ell}$	A decision boolean variable that shows if an instance i of VNF ϑ uses the flavor $\ell \in \mathcal{L}^{\mathcal{CN}}$.
$\mathcal{Z}_{\omega,\mathcal{CN}}$	A decision boolean variable that shows if a subset of TAs $\omega \subseteq \Omega$ would be handled by a $\mathcal{CN}$.

Fig. 2 illustrates the main overview of the envisioned EP-

CaaS framework, dubbed virtual-EPC throughout this paper. Fig. $2(a)$ depicts a network architecture that consists of: $i)$ three $\mathcal{CN}$s managed by three different VIMs (i.e., Rackspace, OpenStack and Amazon AWS); $ii)$ a set of TAs that form the RAN. As discussed above, virtual-EPC will instantiate vEPC (i.e., alternatively virtual 5G Core in case of 5G core network functions) that will manage and handle the traffic generated by these TAs. As depicted in Fig. $2(b)$, virtual-EPC will specify the number of instances of each VNF ϑ, their flavors, as well as their locations in different $\mathcal{CN}$s. In this figure, we present only how to instantiate vEPC, however, following the same fashion, the virtual 5G core network slice can be also instantiated. Moreover, virtual-EPC will assign for each TA the different instances of different VNFs ϑ that are responsible for providing connectivity in that TA. As mentioned before, virtual-EPC will create the SIG that will be used by NFVO. The latter will communicate the appropriate information in SIG to different VIMs, managing different $\mathcal{CN}$s, for instantiating the VNFs. In this figure, the square and triangle shapes present HSS and MME, respectively. While SGW and PGW are presented with a circle. Different colors of the shapes besides TAs indicate the $\mathcal{CN}$s, where these VNFs are instantiated. Note that this picture is used to ease the understanding of the idea behind our proposed virtual-EPC framework. In real-life, one instance of a VNF may handle multiple TAs.

B. Network model

TABLE II
AN EXAMPLE OF A SET OF EVENTS IN A LTE NETWORK.

Event	**Event description**
Attach	When a UE attaches to the network.
Detach	When a UE detaches from the network.
PDN connection setup	When a UE connects to a PDN network.
PDN disconnection	When a UE disconnects from the network.
Handover	When a UE moves between two neighboring eNodeBs.
TA update	When a UE enters into a new tracking area. Such TA update can happen with or without SGW relocation.
S1-Release	Releasing $S1$ connection when a UE becomes inactive.

For every event in the network, such as the attach or detach of a UE, a set of procedures should be executed. Each procedure generates tens of messages exchanged among the different network components. The different notations used throughout the paper are summarized in Table I. We denote by $\lambda_{x,\vartheta}$ the amount of resources in terms of CPU and memory required by an instance of a VNF ϑ to handle an event $x \in \Upsilon$. Formally, $\lambda_{x,\vartheta}$ is defined as the amount of virtual resources needed by ϑ to handle the number of messages generated due to the event x. It is obvious that an event (e.g., S1-Release, Tracking Area Update (TAU), and attach) does not necessary have the same impact on a particular ϑ (MME, HSS, SGW or PGW). Moreover, some events $x \in \Upsilon$ do not have any impact on some ϑ. In this case, the value of $\lambda_{x,\vartheta}$ is simply set to zero. For example, the event S1-Release does not have much impact on HSS, and accordingly $\lambda_{\text{S1-Release,HSS}} = 0$. Table II shows a list of some events (and their descriptions) that can occur in vEPC/5G Core. For a specific event, not necessarily all concerned VNFs would be involved. For example, in case of TA update without SGW relocation, only the MME would communicate with eNodeBs, informing the current SGW of the target eNodeB but not involving a new SGW. Another example is the $X2$-based handover: when a UE moves from one eNodeB to another one, a set of messages would be forwarded from these eNodeBs to MME. If these eNodeBs are belonging to different SAs, SGW relocation would occur involving a new SGW in the process to start the service area (SA) relocation process.

We assume that each $\mathcal{CN}$ offers virtual resources with a set of flavors $\mathcal{L}^{\mathcal{CN}}$ in terms of capacities. Each flavor would be used as a model class to instantiate the different VNFs. According to the required resources, $\mathcal{CN}$ would use the appropriate flavors to increase QoS; its profits should be increased while QoS and functionality of the system are not impacted. According to the amount of resources dedicated for every flavor, the instance of a VNF that uses it would handle a specific rate of traffic (i.e., the amount of traffic per second). The more resources used by that instance are, the higher the traffic rate it can handle. We denote by $\mathcal{MAX}_{\ell}^{\mathcal{CN}}$ the maximum traffic rate that can be handled by an instance of VNF with a flavor $\ell \in \mathcal{L}^{\mathcal{CN}}$. On the other hand, the cost of an instance of VNF increases proportionally with the resources used in its flavor. The more required resources in a flavor are, the higher the cost of a VNF becomes. We assume that $\mathcal{CN}$s have different flavors with different costs. $\tau_{\ell}^{\mathcal{CN}}$ denotes the cost of a flavor ℓ in a $\mathcal{CN}$.

Before the execution of the proposed algorithm, the network is observed for a specific learning period $\mathcal{D}$, when information about different events are gathered from the network. We denote by Υ the set of events that occurred in the network during the period $\mathcal{D}$. To facilitate the management of our system, we adopted a discrete event dynamic system (DEDS) to observe the events during the period $\mathcal{D}$. The time is counted only for an event $e \in \Upsilon$ that has occurred in the network. We denote by $\mathcal{T}$ the time in discrete format, where each element $t \in \mathcal{T}$ represents the time corresponding to one or multiple events. Ω denotes the set of TAs in the network. Each TA $\mathcal{A} \in \Omega$ is responsible for a set of events that would be handled by vEPC/5G Core. The number of events of type $x \in \Upsilon$, generated from a TA $\mathcal{A} \in \Omega$ until a specific time $t \in \mathcal{T}$, is denoted by $\Gamma_{\mathcal{A}}^{x}[t]$. If a set of TAs is handled by the same instances of a VNF, some events would be omitted. For example, if TA $\mathcal{A} \in \Omega$ and $\mathcal{B} \in \Omega$ are handled by the same instance of SGW, TA service request event would not be generated, if a UE moves from $\mathcal{A}$ to $\mathcal{B}$ within Ω. $\Phi_{\mathcal{A},\mathcal{B}}^{x}[t]$ denotes the number of events of type $x \in \Upsilon$, which would be omitted, if TAs $\mathcal{A}, \mathcal{B} \in \Omega$ use the same instance of a VNF.

Each VNF in vEPC/5G Core $\vartheta \in \mathcal{V}$ (e.g., SGW or MME) has a specific role to accomplish. This means that two VNFs with two different roles should not be considered as one VNF. Moreover, the instantiation of an instance of a VNF does not have any impact on the number of generated events on another instance of another VNF. We model the instantiation of different instances of each VNF $\vartheta \in \mathcal{V}$ as a coalitional

Algorithm 1 Algorithm of virtual-EPC

Input:

Υ: The set of events that can occur in the network.
Ω: The set of all tracking areas.
$\mathcal{V}$: The set of VNFs type that would be deployed in vEPC.
Σ: The set of all data centers.

1: $\mathcal{T} = compute\mathcal{T}()$;
2: **for all** $\vartheta \in \mathcal{V}$ **do**
3: **for all** $(\mathcal{A}, x) \in (\Omega, \Upsilon)$ **do**
4: $\Gamma^x_{\mathcal{A}} = compute\Gamma(\vartheta, \mathcal{T}, \mathcal{A}, x)$;
5: **end for**
6: **for all** $((\mathcal{A}, \mathcal{B}), x) \in (\Omega^2, \Upsilon)$ and $\mathcal{A} \neq \mathcal{B}$ **do**
7: $\Phi^x_{\mathcal{A},\mathcal{B}} = compute\Phi(\vartheta, \mathcal{T}, \mathcal{A}, \mathcal{A}, x)$;
8: **end for**
9: $\Xi = instanceVNF(\vartheta, \Gamma, \Phi)$;
10: *bestCoalition*(Ξ);
11: **end for**

game. The system model, adopted in this paper, considers the problem whereby the $\mathcal{CN}$s collaborate together to form vEPC/5G Core, such that their profits are increased. The profit of a $\mathcal{CN}$, in this paper, refers to the difference between P – the price that the operator is willing to pay – and the cost needed to handle the traffic generated from different TAs associated to this $\mathcal{CN}$, in terms of CPU, storage, and VM management. For the sake of simplicity and without loss of generality, we assume that the price P can be split into multiple parts, where each one would be used for deploying a VNF ϑ (i.e. HSS, MME, SGW and PGW). Let $\mathtt{P}_\vartheta$ denote the price for deploying a VNF ϑ. If the QoS required in vEPC/5G Core is not ensured, no profit is gathered from holding different VNFs. We assume that every $\mathcal{CN}$ is selfish and therefore will not host VNFs, if it cannot make some profit.

Algorithm 1 illustrates the steps of the proposed framework. Virtual-EPC starts by constructing $\mathcal{T}$ (Algorithm 1: Line 1). Based on the above-mentioned remark, the instantiation of VNFs with different roles can be done sequentially; one after another. For this reason, virtual-EPC creates the different VNFs of vEPC/5G Core one by one (Algorithm 1: Line 2). For each VNF $\vartheta \in \mathcal{V}$, virtual-EPC creates all the required instances in different $\mathcal{CN}$s to handle the traffic generated from different TAs. For every component and at each time $t \in \mathcal{T}$, virtual-EPC computes the cumulative number of events generated from different TAs $\mathcal{A} \in \Omega$, which allocates the VNF ϑ (Algorithm 1: Lines $3-5$). Then, for every pair of TAs $\mathcal{A}, \mathcal{B} \in \Omega$ and at each time $t \in \mathcal{T}$, virtual-EPC also computes the cumulative number of events that would be removed, if they are handled by the same VNF (Algorithm 1: Lines $6-8$). Finally, a coalitional game is executed for every VNF ϑ to create instances of ϑ in different $\mathcal{CN}$s, such that the individual profit of each $\mathcal{CN}$ in the coalition is maximized.

IV. VNF CREATION THROUGH COALITIONAL GAME

This section describes the procedure *"instanceVNF"* to instantiate the different instances of one VNF $\vartheta \in \mathcal{V}$ (e.g., MME/SGW). We model this problem as a coalitional game, whereby the different $\mathcal{CN}$s are the players desiring to increase their individual profits. We assume that each $\mathcal{CN}$ has enough resources to accommodate the different instances needed by *vEPC/5G Core*. We also assume that each instance has enough resources allocated to it to handle the traffic generated from at least one *TA*. In this section, we are interested in instantiating virtual resources to accommodate the instances of one VNF ϑ. According to the required resources, $\mathcal{CN}$ would use the appropriate flavors to increase QoS; its profits should be increased while ensuring that QoS and functionality of the system are not impacted. According to the amount of resources dedicated for every flavor, the instance of a VNF that uses that flavor would handle a specific rate of traffic (i.e., the amount of traffic per second).

For every VNF ϑ (i.e., SGW, PGW or MME), a set of $\mathcal{CN}$s ($\mathcal{S}$) would participate in a coalition to host all the instances of ϑ. Every cloud network $\mathcal{CN}_i \in \mathcal{S}$ will handle the traffic caused by a set of TAs $\omega_i \subseteq \Omega$. We define the handling of TAs by a cloud network $\mathcal{CN}_i \in \mathcal{S}$ as a function $\mathcal{H} : \Sigma \rightarrow \Omega$, where $\mathcal{H}(\mathcal{CN}_i) = \{\omega_i\}$. The different centers agree to form a coalition by ensuring the following two properties:

1) System functionality: Every TA would be associated with at least one instance of the VNF ϑ; i.e. there should be no TA $\mathcal{A}$ that does not have an instance of the VNF ϑ. Formally, $\forall \mathcal{A} \in \Omega \Longrightarrow \exists \mathcal{CN} \in \mathcal{S} \wedge \mathcal{A} \in \mathcal{H}(\mathcal{CN})$. Also, the different $\mathcal{CN}$s in a coalition $\mathcal{S}$ should not be overloaded, especially during the peak hours, when the majority of UEs are connected. This allocation of *TA*s to $\mathcal{CN}$s should not have any impact on QoS (i.e., reduce the quality);
2) $\mathcal{CN}$ profit: $\mathcal{H}$ should be defined in a way that maximizes the profits of the coalition members. In order to achieve this objective, two functions are defined:
   a) $\mathcal{F}(\vartheta, \mathcal{CN}, \omega \subseteq \Omega)$: A function defined as $\mathcal{F} : \mathcal{V} \times \Sigma \times \Omega \rightarrow \mathbb{R}^+$, which returns the minimum cost to create different instances of the VNF ϑ at $\mathcal{CN}$, in order to handle the traffic generated by a set of tracking areas ω. This function ensures the use of optimal flavors $\mathcal{L}^{\mathcal{CN}}$ to create different instances with minimum cost, while equally ensuring the system functionality;
   b) $\mathcal{G}(\vartheta, \mathcal{S}, \Omega)$: A function defined as $\mathcal{G} : \mathcal{V} \times \Sigma \times \Omega \rightarrow \mathbb{R}^+$, which returns the minimum possible cost to create different instances of the VNF ϑ in a coalition $\mathcal{S}$ to handle the mobile traffic. Besides the optimal allocation of different TAs to different $\mathcal{CN}$s in the coalition, this function guarantees the use of the optimal flavors in each $\mathcal{CN}$ to create different instances of ϑ. The set of all TAs Ω would be divided among the different coalition members $\mathcal{S}$. In order to increase the profit, each TA should be associated with only one instance of ϑ in a specific $\mathcal{CN}$. Therefore, $\forall \mathcal{CN}_i, \mathcal{CN}_j \in \mathcal{S} \wedge \mathcal{CN}_i \neq \mathcal{CN}_j \Rightarrow \mathcal{H}(\mathcal{CN}_i) \cap \mathcal{H}(\mathcal{CN}_j) = \emptyset$.

The following section presents the two optimization problems for $\mathcal{F}(\vartheta, \mathcal{CN}, \omega \in \Omega)$ and $\mathcal{G}(\vartheta, \mathcal{S}, \Omega)$.

A. Optimal TAs' cost within the same $\mathcal{CN}$

In this subsection, we describe the function $\mathcal{F}(\vartheta, \mathcal{CN}, \omega \subseteq \Omega)$. The aim of this function is to increase the profit of a $\mathcal{CN}$ (i.e., cloud provider), while guaranteeing the system functionality (i.e. no QoS degradation). $\mathcal{CN}$ would reduce the

number of instances of ϑ as much as possible to increase its profit. In order to ensure the system functionality, the generated traffic due to all the events Υ and handled by an instance of ϑ should not exceed the instance capacity, especially during the peak hours. Let $\delta \subseteq \omega$ denote the set of TAs handled by the same instance of ϑ $\mathfrak{m}$, which uses flavor ℓ. To guarantee that $\mathfrak{m}$ is not overloaded, we need to ensure that the resources required to handle the maximum traffic rate generated due to Υ should not exceed $\mathcal{MAX}_{\ell}^{\mathcal{CN}}$. Let ξ_{δ} denote the resources required to handle the maximum traffic rate generated from δ. ξ_{δ} is expressed as follows:

$$\xi_{\delta} = \max_{\forall t_1, t_2 \in \mathcal{T}} \Bigg[\sum_{\forall x \in \Upsilon} \lambda_{x,\vartheta} \cdot \Bigg(\Bigg(\sum_{\forall \mathcal{A} \in \delta} \frac{\Gamma_{\mathcal{A}}^{x}[t_2] - \Gamma_{\mathcal{A}}^{x}[t_1]}{t_2 - t_1} \Bigg) - \Bigg(\sum_{\forall \mathcal{A}, \mathcal{B} \in \delta, \mathcal{A} \neq \mathcal{B}} \frac{\Phi_{\mathcal{A},\mathcal{B}}^{x}[t_2] - \Phi_{\mathcal{A},\mathcal{B}}^{x}[t_1]}{t_2 - t_1} \Bigg) \Bigg) \Bigg] \tag{1}$$

From Eq. (1), we obtain the required resources to handle the maximum traffic rate within any interval $[t_1,\ t_2] \in \mathcal{T}$. We are interested in handling the active UEs' requests simultaneously without affecting the response time and preventing any Denial of Service issue. For this reason, we take into account the rate of each event, $x \in \Upsilon$, $\frac{\Gamma_{\mathcal{A}}^{x}[t_2] - \Gamma_{\mathcal{A}}^{x}[t_1]}{t_2 - t_1}$ instead of the amount of events. In Eq. (1), $\sum_{\forall x \in \Upsilon} \lambda_{x,\vartheta} \cdot \left(\sum_{\forall \mathcal{A} \in \delta} \frac{\Gamma_{\mathcal{A}}^{x}[t_2] - \Gamma_{\mathcal{A}}^{x}[t_1]}{t_2 - t_1} \right)$ represents the sum of all events generated between t_1 and t_2, scaled by the resources required by each event. As we have mentioned before, if the traffic generated from a set of TAs is handled by the same instance of a VNF, some events would be omitted. $\Phi_{\mathcal{A},\mathcal{B}}^{x}[t]$ denotes the number of events of type $x \in \Upsilon$, which would be omitted if TAs $\forall \mathcal{A}, \mathcal{B} \in \omega$ use the same instance of a VNF. As all TAs δ would use the same instance of a VNF, we have then to omit all resources required to handle the events $\Phi_{\mathcal{A},\mathcal{B}}^{x}[t], \forall \mathcal{A}, \mathcal{B} \in \delta \wedge \mathcal{A} \neq \mathcal{B}$ when computing ξ_{δ}. Formally, $\sum_{\forall \mathcal{A}, \mathcal{B} \in \delta, \mathcal{A} \neq \mathcal{B}} \lambda_{x,\vartheta} \cdot \left(\frac{\Phi_{\mathcal{A},\mathcal{B}}^{x}[t_2] - \Phi_{\mathcal{A},\mathcal{B}}^{x}[t_1]}{t_2 - t_1} \right)$ denotes the sum of events, which would be omitted between t_1 and t_2, scaled by the resources required by each event.

Theorem 1. *Achieving optimal profit through $\mathcal{F}$ is NP-hard problem.*

Proof. Let $P1$ denote the problem of creating a set of instances of a VNF with different flavor types to achieve the optimal profit for involved $\mathcal{CN}$s. For the sake of simplicity and without loss of generality, we assume that the events would not be omitted, and each TA $\mathcal{A} \in \omega$ requires an amount of resources (i.e., its weight) denoted by $\xi_{\mathcal{A}}$. Let ξ_{ω} denote (i.e., ω's weight) the amount of resources required by all TAs in ω. Formally, $\xi_{\omega} = \sum_{\forall \mathcal{A} \in \omega} \xi_{\mathcal{A}}$. Let $P2$ denote the knapsack problem, which is defined as follows: given a set of objects, each of which has a weight and value, the problem consists in determining the number of each item that should be put in the knapsack so that the total weight does not exceed a specific weight and the total value is maximized. The optimization of knapsack problem is well known in the literature that it is *NP-hard* problem. To prove that $P1$ is *NP-hard*, it is sufficient to proof that the knapsack problem $P2$ would be reduced to $P1$ in a polynomial time. If $P2$ is reduced to $P1$ in a polynomial time and $P1$ is not *NP-hard*, then $P2$ is also not *NP-hard*, which is a contradiction. Let assume that $P2$ is formulated as follows: the total weight that can be handled by the knapsack is ξ_{ω} and the set of items that should be put in knapsack is the set of flavors $\mathcal{L}^{\mathcal{CN}}$, such that the $\mathcal{CN}$ profit is maximized. $P2$ can be reduced in a polynomial time to $P1$ by dividing the knapsack into a set of TAs and then associating them with different flavors. □

As the maximization of the profit of a $\mathcal{CN}$ through $\mathcal{F}$ is NP-hard, the optimal distribution of instances of a ϑ cannot be achieved in a polynomial time. Heuristic or metaheuristic algorithms can be used to find a sub-optimal solution. However, the values obtained from these algorithms can be far from the optimal value, which can affect dramatically the benefits of $\mathcal{CN}$. Based on the observation that: i) the number of TAs is limited and ii) the proposed vEPC/5G Core instantiation algorithm would be executed only once in the initial step and would be explored for a while; $\mathcal{F}$ is modeled using Mixed Linear Integer Programming (MILP) to achieve an optimal solution. A dedicated machine can be used to execute the proposed vEPC/5G Core instantiation algorithm. Moreover, the number of TAs is usually very limited; in the worst case they are some dozens. In such scenario, the complexity of the optimization is manageable, and the convergence to the optimal solution would be within tolerable runtime delay. As indicated in Table I, we define two variables: i) $\mathcal{X}_{\delta,i}$, which is a decision boolean variable that shows if a subset of TAs $\delta \subseteq \omega \subseteq \Omega$ uses the same instance of ϑ i; ii) $\mathcal{Y}_{i,\ell}$ is a decision boolean variable that shows if the instance of ϑ i uses the flavor $\ell \in \mathcal{L}^{\mathcal{CN}}$.

$$\mathcal{X}_{\delta,i} = \begin{cases} 1 & \text{If TAs in } \delta \in \mathcal{P}(\omega) \text{ use the same instance } i \text{ of the VNF } \vartheta \\ 0 & \text{Otherwise} \end{cases}$$

$$\mathcal{Y}_{i,\ell} = \begin{cases} 1 & \text{If the instance } i \text{ of the VNF } \vartheta \text{ uses the flavor type } \ell \\ 0 & \text{Otherwise} \end{cases}$$

Based on the assumption that each flavor is able to handle at least the traffic generated by one *TA*, the number of instances of the VNF ϑ that would be deployed in the network should not exceed $|\omega|$. The function $\mathcal{F}(\vartheta, \mathcal{CN}, \omega \subseteq \Omega)$ will be then formulated as follows:

$$\mathbf{min}\, \mathcal{F}(\vartheta, \mathcal{CN}, \omega \subseteq \Omega) = \sum_{i=1}^{|\omega|} \sum_{\forall \ell \in \mathcal{L}^{\mathcal{CN}}} \tau_{\ell}^{\mathcal{CN}} \cdot \mathcal{Y}_{i,\ell} \tag{2}$$

$S.t,$

$$\forall \mathcal{A} \in \omega : \sum_{\forall \delta \in \mathcal{P}(\omega), \mathcal{A} \in \delta} \sum_{i=1}^{|\omega|} \mathcal{X}_{\delta,i} = 1 \tag{3}$$

$$\forall \ell \in \mathcal{L}^{\mathcal{CN}}, \forall \delta \in \mathcal{P}(\omega), i = 1 \cdots |\omega| : \xi_{\delta} \cdot \mathcal{X}_{\delta,i} \leq \mathcal{MAX}_{\ell}^{\mathcal{CN}} \cdot \mathcal{Y}_{i,\ell} \tag{4}$$

$$\forall \delta \in \mathcal{P}(\omega), i = 1 \cdots |\omega| : \mathcal{X}_{\delta,i} \in \{0,\ 1\} \tag{5}$$

$$\forall \ell \in \mathcal{L}^{\mathcal{CN}}, i = 1 \cdots |\omega| : \mathcal{Y}_{i,\ell} \in \{0,\ 1\} \tag{6}$$

In this optimization problem, we aim at minimizing as much as possible the cost spent by $\mathcal{CN}$ in order to increase its profit. In Objective (2), we aim at using the minimum number of

instances of ϑ, while selecting for those instances only the flavors that reduce the cost. Note that in the objective, we can have at most $|\omega|$ instances of ϑ in $\mathcal{CN}$, where every instance of the VNF ϑ serves only one TA. However, we may not need all of them. If an instance of ϑ i is not needed, the optimization problem ((2)–(6)) will set the variable $\mathcal{Y}_{i,\ell}$ to 0, $\forall \ell \in \mathcal{L}^{\mathcal{CN}}$. Constraint (3) ensures the system functionality; Every *TA* $\mathcal{A}$ should be handled by only one instance of ϑ. In (3), $\mathcal{X}_{\delta,i} = 1$, if the power set $\delta \in \mathcal{P}(\omega)$ is handled by the instance of ϑ i. In (3), we ensure that every *TA* $\mathcal{A} \in \omega$ should belong to only one $\delta \in \mathcal{P}(\omega)$, which is in turn handled only by one machine i ($\mathcal{X}_{\delta,i} = 1$). Constraint (4) makes a relation between variables $\mathcal{X}_{\delta,i}$ and $\mathcal{Y}_{i,\ell}$. Moreover, it ensures that if a power set $\delta \in \mathcal{P}(\omega)$ is selected ($\mathcal{X}_{\delta,i} = 1$), then the best flavor ℓ should be used in its VNF. The flavor ℓ should have the lowest cost while offering the required resources by δ. Formally, if $\mathcal{X}_{\delta,i} = 0$, then $\mathcal{Y}_{i,\ell}$ should be also equal to 0 in order to reduce the cost (2). Meanwhile, Constraints (5) and (6) ensure that $\mathcal{X}_{\delta,i}$ and $\mathcal{Y}_{i,\ell}$ are decision Boolean variables.

B. Optimal TAs' cost within a coalition of $\mathcal{CN}$s

In this subsection, we will show how to create a set of instances of ϑ in a coalition of $\mathcal{CN}s$ to handle the set of *TAs* in a cloud-based Evolved Universal Terrestrial Radio Access Network - *E-UTRAN*, such that the total profit of $\mathcal{CN}$s is increased. The function $\mathcal{G}(\vartheta, \mathcal{CN}, \Omega)$ is defined herein to achieve this objective. The aim of this function is to increase as much as possible the total profit while the system functionality is not affected. The system functionality is ensured in this function through the use of function $\mathcal{F}$.

The function $\mathcal{G}$ is modeled using Mixed Linear Integer Programming (MILP). As stated before, the creation of *vEPC/5G Core* would be done once and would be explored for a long time. This makes the computational complexity of MILP negligible compared to the profit which would be achieved. On the other hand, the number of TAs in a RAN is very limited. In such scenario, the complexity of the optimization is manageable and the convergence to the optimal solution can be within tolerable runtime delay. As we have mentioned in Table I, we define the variable $\mathcal{Z}_{\omega,\mathcal{CN}}$, which is a decision Boolean variable that shows if a subset of TAs $\omega \subseteq \Omega$ is handled by a $\mathcal{CN}$.

$$\mathcal{Z}_{\omega,\mathcal{CN}} = \begin{cases} 1 & \text{If a set of TAs } \omega \subseteq \mathcal{P}(\Omega) \text{ are handled by } \mathcal{CN} \\ 0 & \text{Otherwise} \end{cases}$$

The function $\mathcal{G}(\vartheta, \mathcal{S}, \Omega)$ can be formulated as follows:

$$\textbf{min}\, \mathcal{G}(\vartheta, \mathcal{S}, \Omega) = \sum_{\forall \omega \in \mathcal{P}(\Omega)} \sum_{\forall \mathcal{CN} \in \mathcal{S}} \mathcal{F}(\mathcal{CN}, \omega) \cdot \mathcal{Z}_{\omega,\mathcal{CN}} \tag{7}$$

$S.t,$

$$\forall \mathcal{A} \in \Omega : \sum_{\forall \omega \in \mathcal{P}(\Omega), \forall \mathcal{A} \in \omega} \sum_{\forall \mathcal{CN} \in \mathcal{S}} \mathcal{Z}_{\omega,\mathcal{CN}} = 1 \tag{8}$$

$$\forall \mathcal{CN} \in \mathcal{S} : \sum_{\forall \omega \in \mathcal{P}(\Omega)} \mathcal{Z}_{\omega,\mathcal{CN}} \geq 1 \tag{9}$$

$$\forall \omega \in \mathcal{P}(\Omega), \forall \mathcal{CN} \in \Sigma : \mathcal{Z}_{\omega,\mathcal{CN}} \in \{0,\ 1\} \tag{10}$$

In this optimization problem, we aim to reduce the cost incurred by a coalition $\mathcal{S}$ of $\mathcal{CN}$s to handle all the TAs in the RAN; leading to an increase in their profits. In the objective (7), we aim to assign for every $\mathcal{CN}$ a set of *TAs* ω, such that the total cost is reduced as much as possible. Constraint (8) ensures the system functionality. We guarantee in this constraint that every *TA* $\mathcal{A}$ should be handled by only one $\mathcal{CN}$. Meanwhile, Constraint (9) ensures that every $\mathcal{CN}$ in the coalition should handle at least one power set ω from Ω. A $\mathcal{CN}$ would not get a profit from $\mathcal{S}$, if it does not handle a set of *TAs* from Ω. Constraint (10) assures that $\mathcal{Z}_{\omega,\mathcal{CN}}$ is a decision Boolean variable.

Theorem 2. *Achieving optimal profit through $\mathcal{G}$ is also NP-hard problem.*

Proof. The function $\mathcal{G}$ calls iteratively the function $\mathcal{F}$, which is NP-hard problem. This makes the function $\mathcal{G}$ NP-hard problem. □

C. Coalitional game background

In this paper, we aim to build a *vEPC/5G Core* in the carrier cloud that ensures the desired QoS. The aim herein is to increase as much as possible the profit of the different $\mathcal{CN}$s involved in this operation. We define the characteristic function of the coalitional game as follows:

$$v(\mathcal{S}) = \begin{cases} 0 & \text{If } |\mathcal{S}| = 0 \text{ or QoS is not ensured.} \\ \mathtt{P}_\vartheta - \mathcal{G}(\vartheta, \mathcal{S}, \Omega) & \text{Otherwise} \end{cases} \tag{11}$$

where $\mathtt{P}_\vartheta$ is the price for deploying VNF ϑ. The QoS is ensured in $\mathcal{S}$ if the following condition is true:

$$\forall \mathcal{CN} \in \mathcal{S}, \forall \omega \in \mathcal{P}(\Omega) : \quad \mathcal{Z}_{\omega,\mathcal{CN}} = 1 \Rightarrow \forall \mathcal{A} \in \omega : \theta_{\mathcal{A},\mathcal{CN}} > \theta_{Th} \tag{12}$$

where $\theta_{\mathcal{A},\mathcal{CN}}$ represents the data rate between *TA* $\mathcal{A}$ and $\mathcal{CN}$, and θ_{Th} is the minimum data rate threshold that should be ensured by virtual-EPC solution. The objective of this paper is to build the vEPC/5G Core in different $\mathcal{CN}$s. Thus, for every VNF ϑ, a coalition of $\mathcal{CN}$s $\mathcal{S}$ would be determined, where the different instances of ϑ would be built. Every $\mathcal{CN} \in \Sigma$ wrestles to increase its profit by joining the coalition that gives the highest profit value. Many ways are proposed in the literature to share the profit $v(\mathcal{S})$ among the different players ($\mathcal{CN}$) in the same coalition $\mathcal{S}$. The shapely value [29] is used in the literature to fairly share the profit among the different players. Another way to share the profit among players is called nucleolus [29]. In nucleolus, instead of applying a general axiomatization of fairness, the different coalitions are arranged in a non-increasing order based on the players' payoffs. This requires to look for every specific characteristic function to maximize the profit of each player. However, the use of these methods require many iterations for every power set of the coalition. An easy way to share the profit among different players is the use of equal sharing method. Due to the simplicity of its implementation, this method is widely used in the literature [29]. In this paper, we also use the equal-sharing method to define the payoff of different players in the coalition. However, any other method (i.e., shapely value or nucleolus) can be used in our framework with slight modification. Based on the above-mentioned remark, the payoff of each player $\mathcal{CN}$ in a coalition $\mathcal{S}$ is defined as follows:

$$\Pi_{\mathcal{S}}(\mathcal{CN}) = \frac{v(\mathcal{S})}{|\mathcal{S}|} \tag{13}$$

Each $\mathcal{CN}$ aims to increase its payoff as much as possible, and then it would decide to enter in a coalition $\mathcal{S}$ iff its payoff

is maximized. Formally, a $\mathcal{CN}$ would participate in a coalition $\mathcal{S}^*$ that satisfies the following equation:

$$\mathcal{S}^* = \underset{\forall \mathcal{S} \in \mathcal{P}(\Sigma)}{\operatorname{argmax}} \Pi_{\mathcal{S}}(\mathcal{CN}) \tag{14}$$

Based on (14), the different $\mathcal{CN}$s would form different coalitions, where the profit of each $\mathcal{CN}$ is maximized. The profit of $\mathcal{CN}$s in each coalition $\mathcal{S}$ would be maximized while QoS is ensured. The different coalitions would be defined through (14), and the two optimization problems ((2) - (6)) and ((7) - (10)). The set of all $\mathcal{CN}$s Σ denotes the grand coalition. The payoff vector of the grand coalition is defined as follows: $\Pi_\Sigma = (\Pi_\Sigma(\mathcal{CN}_1), \cdots, \Pi_\Sigma(\mathcal{CN}_{|\Sigma|}))$, where $\mathcal{CN}_i \in \Sigma$ refers to the i^{th} $\mathcal{CN}$ in Σ. The vector payoff Π_Σ can be fairly divided later among players according to the contribution of each one. The stability of the coalition is the most renowned property of the coalitional game, which refers to the situation that every player does not have the intention to exit its coalition. In what follows, we will define the imputation and the core that would ensure the stability of the grand coalition Σ.

The imputation is defined through the following two properties. The first property is the coalitional rationality, which means that the total amount received by the $\mathcal{CN}$s in the grand coalition should be equal to $v(\Sigma)$.

Definition 1. *According to the payoff vector in [30],* Π_Σ *is* ***coalitional rationality*** *if* $\sum_{\mathcal{CN} \in \Sigma} \Pi_\Sigma(\mathcal{CN}) = v(\Sigma)$.

Based on (13), for any payoff vector in our game, this condition is always verified for any coalition $\mathcal{S}$ including the grand coalition Σ.

The second property is the individual rationality, which means that no player would agree to receive a payoff in the coalition less than the one that will receive by acting alone.

Definition 2. *According to the payoff vector in [30],* Π_Σ *is* ***individual rationality*** *if* $\forall \mathcal{CN} \in \Sigma : v(\mathcal{CN}) \leq \Pi_\Sigma(\mathcal{CN})$.

Definition 3. *According to the payoff vector in [30],* Π_Σ *is an* ***imputation*** *iff it exhibits coalitional rationality and individual rationality. Formally, the imputation is defined as follows:* $\sum_{\mathcal{CN} \in \Sigma} \Pi_\Sigma(\mathcal{CN}) = v(\Sigma)$ *and* $\forall \mathcal{CN} \in \Sigma : v(\mathcal{CN}) \leq \Pi_\Sigma(\mathcal{CN})$.

To define the core, we need to define the unstable imputation. An imputation is unstable through a coalition $\mathcal{S}$ iff $\sum_{\mathcal{CN} \in \mathcal{S}} \Pi_\Sigma(\mathcal{CN}) \leq v(\mathcal{S})$. This means that there exists a coalition $\mathcal{S}$, where its players $\mathcal{CN}$s will gain more if they collaborate together than acting within the grand coalition Σ.

Definition 4. *According to the payoff vector in [30],* Π_Σ *is* ***core*** *iff it does not contain any unstable imputation. Formally, the core is defined as follows:* $\sum_{\mathcal{CN} \in \Sigma} \Pi_\Sigma(\mathcal{CN}) = v(\Sigma)$ *and* $\forall \mathcal{S} \subset \Sigma : v(\mathcal{S}) \leq \sum_{\mathcal{CN} \in \mathcal{S}} \Pi_\Sigma(\mathcal{CN})$.

The core is the set of the payoff vectors that leads the players to agree to form the grand coalition Σ. The core can be composed by many points or can be empty. If the payoff of the game is core, the players will then end up by forming the grand coalition Σ. Otherwise, the grand coalition would not be formed and the VNF ϑ would be created only using the coalition of $\mathcal{CN}$s that have the highest payoff value. In what follows, we will present an example that shows an empty core when forming vEPC/5G Core. We consider three $\mathcal{CN}$s, dubbed $\mathcal{CN}_1, \mathcal{CN}_2$ and $\mathcal{CN}_3$, that compete to host the VNF ϑ. We also consider that the RAN is formed with two *TAs* $\mathcal{A}$ and $\mathcal{B}$. To facilitate the explanation of the example, let $v(\mathcal{CN}, \omega)$ denote the profit of each $\mathcal{CN}$ to handle a set of TAs ω. To compute $v(\mathcal{CN}, \omega)$, similarly to $v(\mathcal{S})$, we compute the different ξ_δ, $\mathcal{F}$ and then $\mathcal{G}$. We formally define $v(\mathcal{CN}, \omega)$ as follows:

$$v(\mathcal{CN}, \omega) = \begin{cases} 0 & \text{If } |\mathcal{S}| = 0 \text{ or QoS is not ensured.} \\ \mathtt{P}_\vartheta - \mathcal{G}(\mathcal{CN}, \omega) & \text{Otherwise} \end{cases} \tag{15}$$

TABLE III
THE PROFIT OF EVERY $\mathcal{CN}$.

TAs	$v(\mathcal{CN}_1, \omega)$	$v(\mathcal{CN}_2, \omega)$	$v(\mathcal{CN}_3, \omega)$
$\{\mathcal{A}\}$	120	140	50
$\{\mathcal{B}\}$	100	90	10
$\{\mathcal{A}, \mathcal{B}\}$	70	80	0

Table III shows the profit of each $\mathcal{CN}$ when handling the different *TA*s. The profits in this table are presented as numbers that represent the difference between $\mathtt{P}_\vartheta$ and the cost incurred by the resources dedicated to a VM/container (i.e., CPU, storage) and the management of VMs/containers (i.e., instantiation and migration), as well as the QoS ensured by different $\mathcal{CN}$. Meanwhile, Table IV shows the different $\mathcal{CN}$ coalitions that can be formed, the mapping of each *TA* with its $\mathcal{CN}$ through functions $\mathcal{F}$ and $\mathcal{G}$, as well as $v(\mathcal{S})$ and $\Pi_\mathcal{S}$. Table IV is filed from Table III using (1), the two optimization problems ((2)−(6)) and ((7)−(10)), (11) and (13), respectively. Based on Constraint (9), every $\mathcal{CN}$ should handle some *TA*s, otherwise the other *TAs* in the coalition would not accept to share with it the profit. In the last row in Table IV, $\mathcal{CN}_1$ and $\mathcal{CN}_2$ would not accept to share the profit with $\mathcal{CN}_3$. Thus, the grand coalition does not exist in this game, and the core is empty. Relaxing the optimization problem ((7) - (10)) by removing Constraint (9), the grand coalition $\mathcal{S} = \{\mathcal{CN}_1, \mathcal{CN}_2, \mathcal{CN}_3\}$ can be formed. However, the coalition $\mathcal{S} = \{\mathcal{CN}_1, \mathcal{CN}_2\}$ may exit the grand coalition as the profit of its players will be increased. Both players $\mathcal{CN}_1$ and $\mathcal{CN}_2$ will gain 120 instead of 80 if they agree to exit the grand coalition. As the coalition $\mathcal{S} = \{\mathcal{CN}_1, \mathcal{CN}_2\}$ satisfies the following conditions $\Pi_\mathcal{S}(\mathcal{CN}_1) \geq v(\mathcal{CN}_1)$ and $\Pi_\mathcal{S}(\mathcal{CN}_2) \geq v(\mathcal{CN}_2)$, the instance of ϑ will be built in $\mathcal{CN}_1$ and $\mathcal{CN}_2$. According to many parameters, such as the cost of each flavor in each $\mathcal{CN}$ and QoS, the core may exist or not in this game.

TABLE IV
THE MAPPING OF DIFFERENT COALITIONS.

$\mathcal{S}$	Mapping through $\mathcal{G}$ and $\mathcal{F}$	$v(\mathcal{S})$	$\Pi_\mathcal{S}$
$\{\mathcal{CN}_1\}$	$\mathcal{A}, \mathcal{B} \longrightarrow \mathcal{CN}_1$	70	70
$\{\mathcal{CN}_2\}$	$\mathcal{A}, \mathcal{B} \longrightarrow \mathcal{CN}_2$	80	80
$\{\mathcal{CN}_3\}$	$\mathcal{A}, \mathcal{B} \longrightarrow \mathcal{CN}_3$	0	0
$\{\mathcal{CN}_1, \mathcal{CN}_2\}$	$\mathcal{A} \longrightarrow \mathcal{CN}_2$, $\mathcal{B} \longrightarrow \mathcal{CN}_1$	240	120
$\{\mathcal{CN}_1, \mathcal{CN}_3\}$	$\mathcal{A} \longrightarrow \mathcal{CN}_3$, $\mathcal{B} \longrightarrow \mathcal{CN}_1$	150	75
$\{\mathcal{CN}_2, \mathcal{CN}_3\}$	$\mathcal{A} \longrightarrow \mathcal{CN}_2$, $\mathcal{B} \longrightarrow \mathcal{CN}_3$	150	75
$\{\mathcal{CN}_1, \mathcal{CN}_2, \mathcal{CN}_3\}$	$\mathcal{A} \longrightarrow \mathcal{CN}_2$, $\mathcal{B} \longrightarrow \mathcal{CN}_1$	240	80

V. VIRTUAL-EPC MECHANISM

In this section, we present the basic concept of the coalitional game and the different mechanisms used for instantiating vEPC/5G Core. In what follows, we will present the different coalitional game mechanisms to build the instances of each VNF ϑ one by one.

A. Coalitional game for building one vEPC/5G Core instance

The coalitional games are classified into three classes [29]: i) Canonical (coalitional) games; ii) Coalitional formation games; and iii) Coalitional graph games. In the first category, the grand coalition of the players is optimal. The aim in this kind of games is how to stabilize the grand coalition by suggestion an appropriate payoff vector. The second category, used in this paper, aims to form the different coalitions, such that the profit of the different players is increased. The coalition formation would be defined according to the gain and the cost from the cooperation. The main objective of the studies on this category of games is to define the appropriate game structure and study its property. In the last category of games, the players interact between them via a communication graph. The main objective of this game is the stabilization of the grand coalition or the forming of different coalitions by considering the communication graph.

Based on the above-mentioned example (Tables III and IV), the core can be empty in this game. Effectively, if some $\mathcal{CN}$s cannot offer the required QoS, the core would be then empty in this game. For this reason, this paper falls in the coalitional formation games category, where the goal is to increase the profit of the coalition of $\mathcal{CN}$s, in order to increase their profit when forming vEPC/5G Core. This game would be repeated for every VNF ϑ until the whole vEPC/5G Core would be deployed. As indicated in Algorithm 1, the function *instanceVNF*$(\vartheta, \Gamma, \Phi)$ is called repetitively for every VNF ϑ. This function will call the coalitional game mechanism to optimally place instances of ϑ. In the end of *instanceVNF*$(\vartheta, \Gamma, \Phi)$ execution, a partition Ξ of the grand coalition Σ should be formed. $\Xi = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_\mathcal{K}\}$, where $\mathcal{K}$ is the number of coalitions which would be created. Each $\mathcal{CN}$ should belong to only one coalition and all $\mathcal{CN}$s should belong to a coalition. Ξ is defined through the following properties: i) $\forall \mathcal{S}_i, \mathcal{S}_j \in \Xi \wedge i \neq j \Longrightarrow \mathcal{S}_i \cap \mathcal{S}_j = \emptyset$; ii) $\bigcup_{\forall \mathcal{S} \in \Xi} \mathcal{S} = \Sigma$. The function *instanceVNF*$(\vartheta, \Gamma, \Phi)$ will return a set of coalitions, where the VNF ϑ would be deployed. virtual-EPC, via *bestCoalition*(Ξ), will choose the best coalition. The selected coalition would be the one that ensures high profit of its members. The coalitions that do not ensure QoS will not be considered as their payoff is zero. In what follows, we will define the concept used in the proposed coalitional game.

Definition 5. *Let C be a set of coalitions defined as $C = \{\mathcal{S}_1, ..., \mathcal{S}_K\}$. C is collection if it satisfies the following condition: $\forall \mathcal{S}_i, \mathcal{S}_j \in C \wedge i \neq j \Longrightarrow \mathcal{S}_i \cap \mathcal{S}_j = \emptyset$. If $\bigcup_{\mathcal{S} \in C} = \Sigma$, then C is called a partition of the grand coalition Σ.*

Definition 6. *In the coalitional game, the different collections would be compared to the coalition that increases the payoff of different players. We denote by $\rhd$ the comparison relation between two partitions C_1 and C_2, that partition the same set η of $\mathcal{CN}$s. Formally, $\bigcup_{\mathcal{S} \in C_1} = \bigcup_{\mathcal{S} \in C_2} = \eta$. We mean by $C_1 \rhd C_2$ the way of grouping different players in C_1 is better than the one in C_2.*

As we have mentioned in Table IV, each $\mathcal{CN}$ aims to increase its profit as much as possible without taking into account the other players. In this game, we are interested in the payoff of every player rather than in the coalition value. The comparison in coalitional game is based on the merge and split rules to form the coalition that will hold the instances of ϑ later. In what follows, we will define the merge and split rules of our game. We consider two collections $C_s = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_K\}$ and $C_m = \bigcup_{i=1}^{K} \mathcal{S}_i$. C_s consists of a set of coalitions, whereas C_m forms one coalition. Both C_s and C_m are partitions of the same set of $\mathcal{CN}$s. We define two comparison relations $\rhd_m$ and $\rhd_s$ for the merge and split, respectively. Note that $\mathcal{CN}s$ are selfish as each of them aims to increase its payoff without carrying about the other players. The merge $\rhd_m$ and split $\rhd_s$ relations are defined as follows:

$$\begin{aligned} C_m \rhd_m C_s \Leftrightarrow \{&(\forall \mathcal{S} \in C_s, \forall \mathcal{CN} \in \mathcal{S} : \\ &\Pi_{C_m}(\mathcal{CN}) \geq \Pi_{\mathcal{S}}(\mathcal{CN})) \\ &\wedge (\exists \mathcal{S}' \in C_s, \exists \mathcal{CN} \in \mathcal{S}' : \\ &\Pi_{C_m}(\mathcal{CN}) > \Pi_{\mathcal{S}'}(\mathcal{CN}))\} \end{aligned} \tag{16}$$

$$\begin{aligned} C_s \rhd_s C_m \Leftrightarrow \{&\exists \mathcal{S} \in C_s : \\ &(\forall \mathcal{CN} \in \mathcal{S} : \Pi_{\mathcal{S}}(\mathcal{CN}) \geq \Pi_{C_m}(\mathcal{CN}) \wedge \\ &\exists \mathcal{CN} \in \mathcal{S} : \Pi_{\mathcal{S}}(\mathcal{CN}) > \Pi_{C_m}(\mathcal{CN}))\} \end{aligned} \tag{17}$$

From (16) C_m is preferred than C_s iff at least one player $\mathcal{CN}$ will enhance its payoff while the payoffs of the others would not be decreased when merging the coalitions in C_s into C_m. Meanwhile, Equation (17) shows that C_s will be preferred than C_m, iff there is at least one coalition $\mathcal{S} \in C_s$, which guarantees that its players would save their profits if they split from C_m and at least one of them will enhance its payoffs. The split process is selfish in a sense that the split of $\mathcal{S}$ from C_m does not take into account the waste of the other players in $C_m - \mathcal{S}$. Based on the split and merge comparison relations, in our game, we propose two rules for the merge and split process:

- Merge rule: The function *instanceVNF*$(\vartheta, \Gamma, \Phi)$ will merge any collection of coalitions $C_s = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_K\}$ to C_m iff $C_m \rhd_m C_s$;
- Split rule: The function *instanceVNF*$(\vartheta, \Gamma, \Phi)$ will split a coalition C_m to $C_s = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_K\}$ iff $C_s \rhd_s C_m$.

A set of coalitions will be merged using merge rule, iff one player will enhance its payoff while the profit of others is saved. Meanwhile, the split rule is selfish in the sense that a coalition $\mathcal{S}$ will leave C_m iff at least one player in $\mathcal{S}$ enhances its payoff, while the payoffs of the other players in $\mathcal{S}$ are not affected and without taking into account the payoffs of the other players in $C_m - \mathcal{S}$. In the end of the *instanceVNF*$(\vartheta, \Gamma, \Phi)$ function, a collection of coalitions will be returned. Using the procedure *bestCoalition*(Ξ), the best coalition from this collection will be selected to hold the instances of ϑ later.

In what follows, we will show how the merge and split processes operate. We consider two coalitions $\mathcal{S}_1$ and $\mathcal{S}_2$,

where $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$. Based on (13) and (16), $\mathcal{S}_1$ and $\mathcal{S}_2$ would be merged into $\mathcal{S}_m = \{\mathcal{S}_1, \mathcal{S}_2\}$ iff the following conditions are fulfilled:

1) The two following conditions are correct:
   a) $\forall \mathcal{CN} \in \mathcal{S}_1 : \Pi_{\mathcal{S}_m}(\mathcal{CN}) \geq \Pi_{\mathcal{S}_1}(\mathcal{CN})$
   b) $\forall \mathcal{CN} \in \mathcal{S}_2 : \Pi_{\mathcal{S}_m}(\mathcal{CN}) \geq \Pi_{\mathcal{S}_2}(\mathcal{CN})$
2) One of the following conditions is correct:
   a) $\exists \mathcal{CN} \in \mathcal{S}_1 : \Pi_{\mathcal{S}_m}(\mathcal{CN}) > \Pi_{\mathcal{S}_1}(\mathcal{CN})$
   b) $\exists \mathcal{CN} \in \mathcal{S}_2 : \Pi_{\mathcal{S}_m}(\mathcal{CN}) > \Pi_{\mathcal{S}_2}(\mathcal{CN})$

The *instanceVNF*$(\vartheta, \Gamma, \Phi)$ function will merge two coalitions $\mathcal{S}_1$ and $\mathcal{S}_2$ iff at least the profit of one player in this coalition will increase while the profit of all the others players will remain unaffected. For the split mechanism, a coalition $\mathcal{S}_m = \{\mathcal{S}_1, \mathcal{S}_2\}$ would split into two coalitions $\mathcal{S}_1$ and $\mathcal{S}_2$, iff the conditions in (17) are met. $\mathcal{S}_1$ (resp., $\mathcal{S}_2$) will split from $\mathcal{S}_m$, iff at least one player in $\mathcal{S}_1$ (resp., $\mathcal{S}_2$) enhances its payoff while the payoffs of the other players in $\mathcal{S}_1$ (resp., $\mathcal{S}_2$) are not affected. Formally, $\mathcal{S}_m = \{\mathcal{S}_1, \mathcal{S}_2\}$ will split into two coalitions $\mathcal{S}_1$ and $\mathcal{S}_2$ iff one of the following conditions are fulfilled:

1) The two following conditions are correct:
   a) $\forall \mathcal{CN} \in \mathcal{S}_1 : \Pi_{\mathcal{S}_1}(\mathcal{CN}) \geq \Pi_{\mathcal{S}_m}(\mathcal{CN})$
   b) $\exists \mathcal{CN} \in \mathcal{S}_1 : \Pi_{\mathcal{S}_1}(\mathcal{CN}) > \Pi_{\mathcal{S}_m}(\mathcal{CN})$
2) The two following conditions are correct:
   a) $\forall \mathcal{CN} \in \mathcal{S}_2 : \Pi_{\mathcal{S}_2}(\mathcal{CN}) \geq \Pi_{\mathcal{S}_m}(\mathcal{CN})$
   b) $\exists \mathcal{CN} \in \mathcal{S}_2 : \Pi_{\mathcal{S}_2}(\mathcal{CN}) > \Pi_{\mathcal{S}_m}(\mathcal{CN})$

The *instanceVNF*$(\vartheta, \Gamma, \Phi)$ function will split any coalition $\mathcal{S}_m$ if the above-mentioned conditions are met. Note that in the split process, the QoS would be ensured as the players would not accept to split if their profits are reduced. As stated earlier, only the best coalition from the collection returned by *instanceVNF*$(\vartheta, \Gamma, \Phi)$ will hold the instances of ϑ. The split process does not affect the best coalition, as a coalition $\mathcal{S}$ would split from another coalition $\mathcal{S}_m$ iff all the profits of its players are not reduced and at least one of them should increase its payoff. Another important feature of a coalitional game is the stability. In a coalitional game, a collection Ξ is ID$_p$-stable if no player has the intention to leave its respective coalition. ID$_p$-stable can be also defined as the set of coalitions in Ξ that do not have the intention to merge or split any further.

B. Algorithm description for building one vEPC/5G Core instance

In this subsection, we will explain the *instanceVNF*$(\vartheta, \Gamma, \Phi)$ function that uses coalitional game to deploy the instances of ϑ across different $\mathcal{CN}$s. In this function, we assume that the QoS desired for a *TA* $\mathcal{A}$ can be assured by every $\mathcal{CN}$. Since both functions $\mathcal{F}$ and $\mathcal{G}$ require an important amount of resources, we will use the dynamic programming technique when implementing the function *instanceVNF*$(\vartheta, \Gamma, \Phi)$. The function $\mathcal{F}$ should not be computed twice for the same $\mathcal{CN}$ and the same $\omega \in \Omega$. Similarly, the function $\mathcal{G}$ should not be computed twice for the same subset of $\mathcal{CN}$s $\mathcal{S}$. Algorithm 2 is used to explain the general functionality of *instanceVNF*$(\vartheta, \Gamma, \Phi)$.

Algorithm 2 *instanceVNF*$(\vartheta, \Gamma, \Phi)$

Input:
ϑ: A component of *vEPC*.
Γ: The number of cumulative events.
Φ: The number of cumulative events would be omitted.

1: $\Xi = \{\{\mathcal{DC}_1\}, \{\mathcal{DC}_2\}, ..., \{\mathcal{DC}_{|\Sigma|}\}\}$;
2: $visited = \emptyset$;
3: **while** $True$ **do**
4: $stop = True$
5: **for all** $\mathcal{S} \in \Xi$ **do**
6: **if** $\mathcal{S} \notin \Psi$ **then**
7: $\Psi[\mathcal{S}] = v(\mathcal{S})$
8: **end if**
9: **end for**
// Merging process
10: **for all** $\mathcal{S}_i, \mathcal{S}_j \in combinations(\Xi, 2) \setminus visited$ **do**
11: $visited = visited \cup \{(\mathcal{S}_i, \mathcal{S}_j)\}$;
12: **if** $\{\mathcal{S}_i \cup \mathcal{S}_j\} \notin \Psi$ **then**
13: $\Psi[\mathcal{S}_i \cup \mathcal{S}_j] = v(\mathcal{S}_i \cup \mathcal{S}_j)$
14: **end if**
// Using Ψ the values of $\Pi_{\mathcal{S}_i}$, $\Pi_{\mathcal{S}_j}$ and $\Pi_{\mathcal{S}_i \cup \mathcal{S}_i}$ are computed
15: **if** $\{\mathcal{S}_i \cup \mathcal{S}_j\} \rhd_m \{\{\mathcal{S}_i\}, \{\mathcal{S}_j\}\}$ **then**
16: $\Xi = \Xi \setminus \{\mathcal{S}_i\}; \Xi = \Xi \setminus \{\mathcal{S}_j\}; \Xi = \Xi \cup \{\mathcal{S}_i \cup \mathcal{S}_j\}$;
17: $stop = False$;
18: $break$;
19: **end if**
20: **end for**
// Split process
21: **for all** $\mathcal{S} \in \Xi \wedge |\mathcal{S}| > 1$ **do**
22: $break = False$;
23: **for all** $\{\mathcal{S}_i, \mathcal{S}_j\} \in \mathcal{S} \wedge \mathcal{S}_i \cup \mathcal{S}_j = \mathcal{S} \wedge \mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ **do**
24: **if** $\mathcal{S}_i \notin \Psi$ **then**
25: $\Psi[\mathcal{S}_i] = v(\mathcal{S}_i)$
26: **end if**
27: **if** $\mathcal{S}_j \notin \Psi$ **then**
28: $\Psi[\mathcal{S}_j] = v(\mathcal{S}_j)$
29: **end if**
// Using Ψ the values of $\Pi_{\mathcal{S}_i}$, $\Pi_{\mathcal{S}_j}$ and $\Pi_{\mathcal{S}}$ are computed
30: **if** $\{\{\mathcal{S}_i\}, \{\mathcal{S}_j\}\} \rhd_s \mathcal{S}$ **then**
31: $\Xi = \Xi \setminus \{\mathcal{S}\}; \Xi = \Xi \cup \mathcal{S}_i; \Xi = \Xi \cup \mathcal{S}_j$;
32: $stop = False$;
33: $break = True$;
34: **break**;
35: **end if**
36: **end for**
37: **if** $break = True$ **then**
38: **break**;
39: **end if**
40: **end for**
41: **if** $stop = True$ **then**
42: **break**;
43: **end if**
44: **end while**
45: **return** Ξ;

Due to space limitation, the calculation redundancy of function $\mathcal{F}$ is not presented in Algorithm 2.

The function *instanceVNF*$(\vartheta, \Gamma, \Phi)$ first starts by forming a collection Ξ by putting every player $\mathcal{CN}$ in a separate coalition (Algorithm 2: Line 1). Then, a variable $visited$ is initialized by $\emptyset$ (Algorithm 2: Line 2). The variable $visited$ is used to keep track of every pair of coalitions which was already visited for the merge. Every visited pair of coalitions will be put in the set $visited$. Then, a while loop is executed where the merge and split processes are executed repetitively until achieving the ID$_p$-stable collection (Algorithm 2: Lines $3-44$). *instanceVNF*$(\vartheta, \Gamma, \Phi)$ computes the values of $v(\mathcal{S})$ using (11)

and ensures that the function $\mathcal{G}$ is executed only when needed (Algorithm 2: Lines $5-9$). To prevent the redundancy in the computation, the vector Ψ is used to store the values of $v(\mathcal{S})$.

In *instanceVNF*$(\vartheta, \Gamma, \Phi)$, the merge and split processes are executed one after the other. In other words, only one merge (Algorithm 2: Lines $10-20$) would be executed, and then only one split (Algorithm 2: Lines $21-40$) would be executed until achieving the ID$_p$-stable collection. In the merging process, every pair of coalitions $\mathcal{S}_i$ and $\mathcal{S}_j$, which are not yet visited, are tested if they can be merged or not (Algorithm 2: Line 10). These pairs of coalitions are put in the vector $visited$ to prevent redundancy checks (Algorithm 2: Line 11). To prevent the execution of the function $\mathcal{G}$ twice, the value of $v(\mathcal{S}_i \cup \mathcal{S}_j)$ will be put in the vector Ψ (Algorithm 2: Lines $12-14$). If the merging condition ($\{\mathcal{S}_i \cup \mathcal{S}_j\} \rhd_m \{\{\mathcal{S}_i\} \cup \{\mathcal{S}_j\}\}$) is verified, we merge these coalitions in the same coalition, and then exit the merging process to execute the split process (Algorithm 2: Lines $15-19$). Otherwise, another pair of coalitions which was not visited yet, will be tested. Meanwhile, in the split process, we will consider every coalition $\mathcal{S}$ that has more than one player $\mathcal{CN}$ (Algorithm 2: Line 21). We try to split every two sub-coalitions $\mathcal{S}_i$ and $\mathcal{S}_j$ of $\mathcal{S}$ that satisfy the following conditions: $i)$ $\mathcal{S}_i \cup \mathcal{S}_j = \mathcal{S}$; $ii)$ $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ (Algorithm 2: Line 23). The partitioning of the coalition $\mathcal{S}$ is done through the partitioning of an integer into two parts [28]. For example, the coalition $\{\mathcal{CN}_1, \mathcal{CN}_2, \mathcal{CN}_3\}$ can be presented with a number 7 (i.e., 111), whereas the coalitions $\{\mathcal{CN}_1, \mathcal{CN}_3\}$ and $\{\mathcal{CN}_2\}$ would be presented with the numbers 5 (i.e., 101) and 2 (i.e., 010), respectively. Enumerating all the possible two sub-coalitions of $\mathcal{S}$ that satisfy the condition in Algorithm 2: Line 23 is equivalent to finding all the two numbers whereby the sum of these numbers equals to the number that represents $\mathcal{S}$. Using the same approach, the redundancy in the computation of $\mathcal{G}$ is also prevented in the split process (Algorithm 2: Lines $24-29$). Then, the function *instanceVNF*$(\vartheta, \Gamma, \Phi)$ splits $\mathcal{S}$ if it is better for the collection Ξ (Algorithm 2: Lines $30-35$). If one split succeeds, we exit the split process and re-initiate the merge process. Note that the variable $stop$ will be set to $false$ if only one merge or one split is carried out, and then the algorithm keeps repeating the loop (Algorithm 2: Lines $3-44$) until achieving the ID$_p$-stable collection. Then, no further merge or split processes will be carried out.

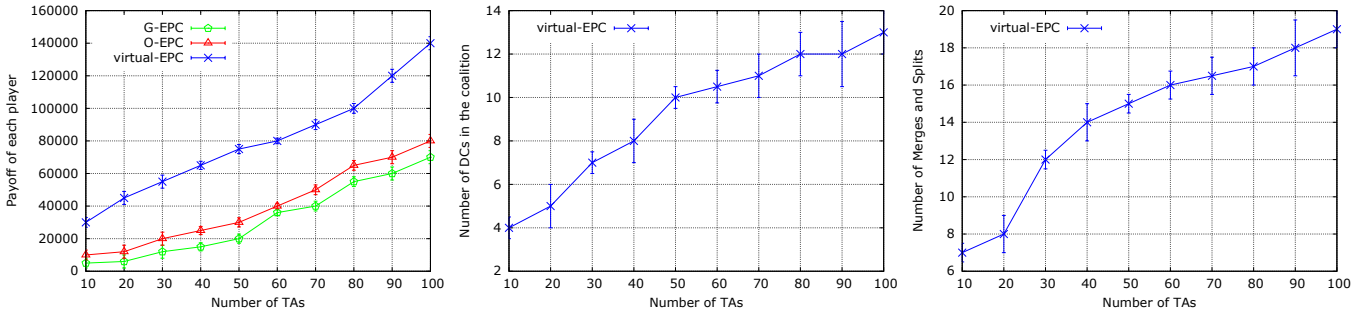
In what follows, we will refer to the example of Tables III and IV to show how the split and merge rules are applied to form the collection of the coalitions Ξ. As mentioned in Table III, the network consists of three $\mathcal{CN}$s $\mathcal{CN}_1$, $\mathcal{CN}_2$ and $\mathcal{CN}_3$. The E-UTRAN is formed in two TAs $\mathcal{A}$ and $\mathcal{B}$. The function *instanceVNF*$(\vartheta, \Gamma, \Phi)$ starts the execution by putting every player $\mathcal{CN}$ in a separate coalition $\Xi = \{\{\mathcal{CN}_1\}, \{\mathcal{CN}_2\}, \{\mathcal{CN}_3\}\}$. Based on Algorithm 2, the merge process starts first. From Table IV, we have $\{\mathcal{CN}_1, \mathcal{CN}_3\} \rhd_m \{\{\mathcal{CN}_1\}, \{\mathcal{CN}_3\}\}$, since both players will increase their payoffs. Then, the collection Ξ will be updated as follows: $\Xi = \{\{\mathcal{CN}_1, \mathcal{CN}_3\}, \{\mathcal{CN}_2\}\}$. In this case, Ξ is composed of two coalitions. In the split process, as the condition $\{\{\mathcal{CN}_1\}, \{\mathcal{CN}_3\}\} \rhd_s \{\mathcal{CN}_1, \mathcal{CN}_3\}$ is not verified, the split process is omitted in this iteration. As the collection Ξ is not ID$_p$-stable, we re-initiated the merging process. Now, we have $\{\mathcal{CN}_1, \mathcal{CN}_3, \mathcal{CN}_2\} \rhd_m \{\{\mathcal{CN}_1, \mathcal{CN}_3\}, \{\mathcal{CN}_2\}\}$, as both players $\mathcal{CN}_1$ and $\mathcal{CN}_3$ will increase their profits while the profit of player $\mathcal{CN}_2$ remains the same. Applying this merge rule will impact the collection Ξ, which will be updated to contain only one coalition $\Xi = \{\mathcal{CN}_1, \mathcal{CN}_2, \mathcal{CN}_3\}$. In the split process, we have $\{\{\mathcal{CN}_1, \mathcal{CN}_2\}, \{\mathcal{CN}_3\}\} \rhd_s \{\mathcal{CN}_1, \mathcal{CN}_3, \mathcal{CN}_2\}$ as the profit of both players in the coalition $\{\mathcal{CN}_1, \mathcal{CN}_2\}$ is increased to 120 instead of 80. Then, the coalition $\{\{\mathcal{CN}_1, \mathcal{CN}_2\}, \{\mathcal{CN}_3\}\}$ will split into two coalitions and then the collection Ξ is updated as $\Xi = \{\{\mathcal{CN}_1, \mathcal{CN}_2\}, \{\mathcal{CN}_3\}\}$. In this state, the coalition $\{\mathcal{CN}_1, \mathcal{CN}_2\}$ will not accept neither a merge nor a split and the game will accordingly terminate. The same collection can be achieved even if we adopt any other order of merge and split rules. As the coalitions in the collection Ξ are not able to merge or split (no player has the intention to exit its respective coalition), Ξ is ID$_p$-stable. *instanceVNF*$(\vartheta, \Gamma, \Phi)$ will return the collection Ξ, and the coalition $\{\mathcal{CN}_1, \mathcal{CN}_2\}$ will be accordingly selected to hold the instances of ϑ as it has the highest payoff value.

Theorem 3. *The function instanceVNF*$(\vartheta, \Gamma, \Phi)$ *terminates and produces an ID$_p$-stable collection.*

Proof. The merge and split processes are repetitively called in the function *instanceVNF*$(\vartheta, \Gamma, \Phi)$ until no further split or merge process is required. In this case, the proposed algorithm terminates and returns the ID$_p$-stable collection. Since the number of partitions of the grand coalition is finite, the function *instanceVNF*$(\vartheta, \Gamma, \Phi)$ would not terminate iff there is a cycle in the merge or split process. In what follows, we will show that there is no cycle in the merge and split processes and accordingly demonstrate that the function *instanceVNF*$(\vartheta, \Gamma, \Phi)$ terminates. The merge and split operations are done based on (16) and (17), where a new collection is more preferred than the previous one. Let $\mathcal{S}_i$ and $\mathcal{S}_j$ be two coalitions in the collection Ξ. $\mathcal{S}_i$ and $\mathcal{S}_j$ would be merged into $\mathcal{S} = \mathcal{S}_i \cup \mathcal{S}_j$ iff $\mathcal{S} \rhd_m \{\mathcal{S}_i, \mathcal{S}_j\}$. In order that the above condition is correct (16), there should exist at least one player $\mathcal{CN} \in \mathcal{S}_i$ (resp., $\mathcal{CN} \in \mathcal{S}_j$) that enhances its payoff in $\mathcal{S}$; $\Pi_{\mathcal{S}}(\mathcal{CN}) > \Pi_{\mathcal{S}_i}(\mathcal{CN})$ (resp., $\Pi_{\mathcal{S}}(\mathcal{CN}) > \Pi_{\mathcal{S}_j}(\mathcal{CN})$). Based on the observation that the proposed algorithm moves from a collection to a better collection after one or multiple merge and split operations, $\mathcal{S}$ cannot split to the coalitions $\mathcal{S}_i$ and $\mathcal{S}_j$. Based on (17), if $\mathcal{S}_i$ and $\mathcal{S}_j$ are merged into $\mathcal{S}$ then $\{\mathcal{S}_i, \mathcal{S}_j\} \rhd_s \mathcal{S}$ would be not verified. We argue this by the fact that there is at least one player $\mathcal{CN} \in \mathcal{S}_i$ (resp., $\mathcal{CN} \in \mathcal{S}_j$), where $\Pi_{\mathcal{S}}(\mathcal{CN}) > \Pi_{\mathcal{S}_i}(\mathcal{CN})$ (resp., $\Pi_{\mathcal{S}}(\mathcal{CN}) > \Pi_{\mathcal{S}_j}(\mathcal{CN})$). Then, no cycle in the merge and split process would be created. This means that the function *instanceVNF*$(\vartheta, \Gamma, \Phi)$ terminates and produces an ID$_p$-stable collection. □

VI. Simulation results

In this section, we evaluate the performance of the proposed scheme to instantiate vEPC/5G Core instances over a federated cloud of $\mathcal{CN}$s. As comparison terms, we use two trivial solutions. The first one, named *G-EPC*, uses the Grand coalition Σ for the deployment of *v-EPC*. In this solution, all



(a) Payoff of each $\mathcal{CN}$. (b) Number of $\mathcal{CN}$s in the selected coalition. (c) Number of merges and split operations.

Fig. 3. The performance evaluation of the proposed virtual-EPC scheme for a varying number of *TAs*.

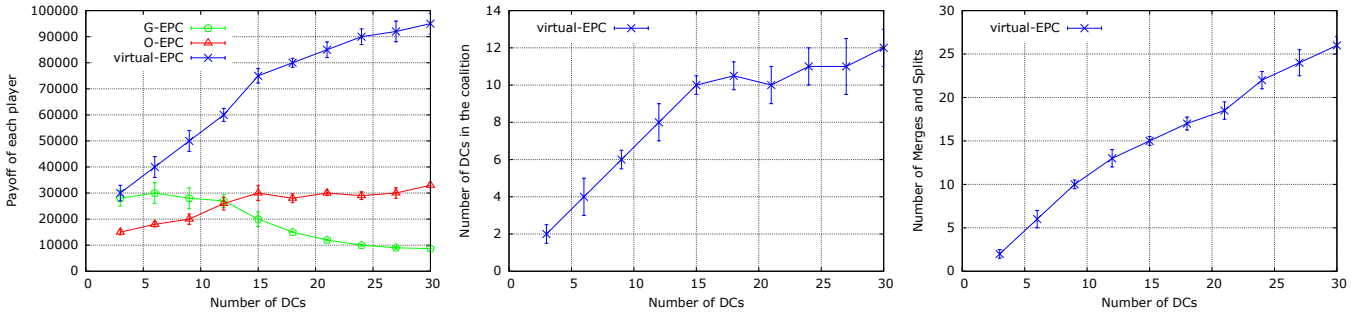
$\mathcal{CN}$s participate in creating different instances of each VNF ϑ, whereas the second solution uses a random deployment over $\mathcal{CN}$s. In this solution, a greedy Algorithm is used, whereby different $\mathcal{CN}$s are randomly selected one by one, as well as the instancing of VNFs are randomly created one by one, until the traffic generated from Ω is handled. Only then, the greedy Algorithm is finished and the random coalition of $\mathcal{CN}$s is selected. The algorithms are evaluated in terms of the following metrics:

- Payoff of individual $\mathcal{CN}$: is defined as the average value of individual payoffs for each player in the selected coalitions of different instances of each VNF ϑ;
- Number of merge and split: is defined as the average number of merge and split operations needed to deploy each VNF ϑ. This metric shows the complexity of the underlying scheme virtual-EPC;
- Number of $\mathcal{CN}$s in the selected coalition: is defined as the average number of players in the selected coalition for each instance of each VNF ϑ.

The algorithms are evaluated using the python programming language and an extended package for graph theory called networkx [31]. We implement the proposed virtual-EPC scheme using IBM ILOG CPLEX version 12.6.1, using the branch-and-bound method to solve the optimization problems. We used historical data from real-life mobile operator network to evaluate the different solutions; i.e., the different events generated in the network, such as the attach or detach operation of a UE, the executed procedure and the number of generated messages. In the simulation results, each plotted point represents the average of 10 executions. The plots are presented with 95% confidence interval. The different algorithms are evaluated by varying the number of TAs and the number of $\mathcal{CN}$s. We conduct two sets of experiments. Firstly, we vary the number of TAs and fix the number of $\mathcal{CN}$s to 15. In the second scenario, we vary the number of $\mathcal{CN}$s while fixing the number of *TAs* to 50. The value of P – the price that a vEPC/5G Core operator is willing to pay – is set proportional to the number of *TAs* in the network.

Fig. 3 shows the performance of the three solutions for a varying number of *TAs*. Fig. 3(a) depicts the impact of the number of TAs on individual payoffs of each $\mathcal{CN}$. We clearly observe that the proposed virtual-EPC solution outperforms both base-line approaches. Moreover, we remark that the use of grand coalition or only one coalition does not yield an optimal solution. From this figure, an increase in the number of TAs has a positive impact on the individual payoffs in all the solutions. For 100 TAs, the proposed virtual-EPC solution achieves an individual payoff of 140000, while the individual payoff of base-line approaches does not exceed 80000, which represents an enhancement of 75%. Indeed, the proposed virtual-EPC solution succeeds in forming the optimal coalition for each instance among all the players $\mathcal{CN}$s, which reduces the cost and hence increases the profit of each player in the selected coalition. In Fig. 3(b), we notice that the number of involved $\mathcal{CN}$s increases proportionally with the number of *TAs* in the network; from which we conclude that the proposed virtual-EPC solution uses the average number of $\mathcal{CN}$s to form vEPC/5G Core instances. On the other hand, we observe from Fig. 3(c) that the number of merge and split operations in the proposed solutions does not exceed 20. This means that the proposed solution converges to the optimal solution within reasonable time. From this figure, we also observe that the number of *TAs* has a negative impact on the number of merge and split operations. The higher the number of *TAs*, the higher the value of $\mathcal{F}(\vartheta, \mathcal{CN}, \omega \in \Omega)$, and consequently the more split and merge operations.

Fig. 4 depicts the performance of the three solutions for varying numbers of players $\mathcal{CN}$s. In Fig. 4(a), we illustrate the evaluation of individual payoff of each $\mathcal{CN}$. From this figure, we remark that an increase in the number of players has a positive impact on the individual payoff of each player in the selected coalitions formed by the proposed solution. Furthermore, we remark that *G-EPC* outperforms *O-EPC* when the number of players is less than 12 $\mathcal{CN}$s. When the number of $\mathcal{CN}$s exceeds 12, *O-EPC* outperforms *G-EPC*. This can be explained as follows: when the number of $\mathcal{CN}$s is less than 12 (the ratio of TAs per $\mathcal{CN}$ is high), then the players in the selected coalitions equitably participate to handle the different *TAs*. Whereas, when the ratio of TAs per $\mathcal{CN}$ decreases, many $\mathcal{CN}$s in *G-EPC* end up sharing the profits for handling only a small number of *TAs*. In contrast to these solutions, the proposed virtual-EPC solution selects the coalitions in an efficient way, such that the profit of the players is increased as much as possible. For example, in case the number of $\mathcal{CN}$s is 30, the proposed scheme outperforms the baseline approaches with more than 233%. Fig. 4(b) shows that the number of $\mathcal{CN}$s in the selected coalitions increases



(a) Payoff of each $\mathcal{CN}$. (b) Number of $\mathcal{CN}$s in the selected coalition. (c) Number of merge and split processes.

Fig. 4. The performance evaluation of virtual-EPC for varying numbers of $\mathcal{CN}$s.

proportionally with the number of $\mathcal{CN}$s in the network. The higher the number of $\mathcal{CN}$s in the network is, the higher the likelihood to select them in the best coalition becomes. Fig. $4(c)$ shows that the number of merge and split operations in the proposed solution increases proportionally with the number of players $\mathcal{CN}$s. An increase in the number of players leads to an increase in the number of possible combinations, which intuitively has a negative impact on the number of merge and split operations. Finally, we observe from this figure that the number of merge and split operations still does not exceed 25; meaning that the proposed solution would converge to the optimal solution within reasonable time.

VII. Conclusion

The upcoming $5G$ mobile system will be based on the concept of carrier cloud to facilitate the upgrade for other next generation mobile systems. The carrier cloud would be enabled through the use of emerging technologies, such as Network Function Virtualization (NFV), Software Defined Networking (SDN) and Cloud Computing. In this paper, we developed a new framework for building virtual EPC instances as a Service (EPCaaS). The aim of this framework is the placement of VNF of virtual EPC in an efficient way over a federated $\mathcal{CN}$. To achieve the desired objectives, two algorithms were proposed: the first one uses Mixed Integer Linear Programming (MILP) to devise the optimal number of virtual resource instances of the different VNFs of vEPC/5G Core that should be deployed in the network; the second algorithm uses coalitional game to place these instances across different $\mathcal{CN}$s, such that the QoS is ensured and the profit of each $\mathcal{CN}$ is maximized. The simulation results demonstrates the efficiency of the proposed framework in achieving its key design objectives.

Acknowledgements

This work was partially supported by the European Union's Horizon 2020 research and innovation programme under the MATILDA and 5G!Pagoda projects under grant agreements No. 761898 and No. 723172, respectively.

References

[1] 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", in IEEE Network Magazine, Vol. 29, No. 2, Mar. 2015. pp.78 - 88.

[2] T. Taleb, "Towards Carrier Cloud: Potential, Challenges & Solutions", in IEEE Wireless Communications Magazine, Vol. 21, No. 3, Jun. 2014. pp. 80-91.

[3] T. Taleb, A. Ksentini, R. Jantii, "Anything as Service for 5G Mobile Systems", to appear in IEEE Network Magazine.

[4] 3GPP TR 28.801 V15.1.0, "Study on management and orchestration of network slicing for next generation network (Release 15)", January 2018.

[5] Open Source Mano (OSM), http://osm.etsi.org/

[6] OpenStack, https://www.openstack.org/

[7] Juju, https://jujucharms.com/

[8] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing", in proc. IEEE GCC'10, 2010, pp. 8792.

[9] M. G. Rabbani, R. Pereira Esteves, M. Podlesny, G. Simon, L. Zambenedetti Granville, and R. Boutaba, "On tackling virtual data center embedding problem", in proc. IFIP/IEEE IM'13, pp. 177 184.

[10] G. Somani, P. Khandelwal, and K. Phatnani, "VUPIC Virtual Machine Usage Based Placement in IaaS Cloud", CoRR abs/1212.0085 (2012)

[11] S. Skiena, The Algorithm Design Manual, ISBN 0-387-94860-0.

[12] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, "Optimization by Simulated Annealing Science", 220 (1983) 671-680.

[13] M. Dorigo and T. Sttzle, Ant Colony Optimization, ISBN 978-0-262-04219-2.

[14] K. Le, J. Zhang, J. Meng, R. Bianchini, Y. Jaluria, and T.D. Nguyen, "Reducing Electricity Cost Through Virtual Machine Placement in High Performance Computing Clouds", in Proc. Intl Conf. on High Performance Computing, Networking, Storage and Analysis (SC), Seatle, WA, USA, Nov. 2011.

[15] C. Hyser, B. McKee, R. Gardner, and B.J. Watson, "Autonomic Virtual Machine Placement in the Data Center", in Proc. HP Labs, HPL-2007-189, Feb. 2008

[16] W. Shi and B. Hong, "Towards Profitable Virtual Machine Placement in the Data Center", in Proc. IEEE Intl Conf. on Utility and Cloud Computing, 2011.

[17] L. Mashayekhy and D. Grosu, "A Coalitional Game-Based Mechanism for Forming Cloud Federations", in Proc. IEEE UCC'12, Chicago, IL, 2012, pp. 223-227.

[18] Ray B.K., Khatua S. and Roy S. "Cloud Federation Formation Using Coalitional Game Theory", in Proc. Springer ICDCIT'15 ACM IWCMC'10, Bhubaneswar, India, Feb. 2015.

[19] M. Guazzone, C. Anglano and M. Sereno, "A Game-Theoretic Approach to Coalition Formation in Green Cloud Federations", in Proc. IEEE ICCG'14, Chicago, IL, 2014, pp. 618-625.

[20] T. Taleb and A. Ksentini, "Gateway Relocation Avoidance-Aware Network Function Placement in Carrier Cloud, in Proc. ACM MSWIM, Barcelona, Nov. 2013.

[21] M. Bagaa, T. Taleb, and A. Ksentini, "Service-Aware Network Function Function Placement for Efficient Traffic Handling in Carrier Cloud", in Proc. of IEEE WCNC 2014, Istanbul, 2014.

[22] F. Z. Yousef, J. Lessmann, P. Loureiro, and S. Schmid, SoftEPC Dynamic Instantiation of Mobile Core Network Entities for Efficient Resource Utilization, in Proc. of IEEE ICC 2013, Budapest, Hungary, Jun. 2013.

[23] ETSI GS NFV 002: "Network Functions Virtualisation (NFV); Architectural Framework"

[24] M. Bagaa, T. Taleb, and A. Ksentini, Efficient Tracking Area Management in Carrier Cloud, in proc. IEEE Globecom'15, San Diego, USA, Dec. 2015.

[25] M. Bagaa, T. Taleb, and A. Ksentini, "Efficient Tracking Area Management Framework for 5G Networks", to appear in IEEE Trans. on Wireless Communications.
[26] M. Bagaa, T. Taleb, A. Laghrissi, and A. Ksentini, "Efficient Virtual Evolved Packet Core Deployment Across Multiple Cloud Domains," to appear in Proc. IEEE WCNC'18, Barcelona, 2018.
[27] A. Kunz, T. Taleb, and S. Schmid, "On Minimizing SGW/MME Relocations in LTE", in Proc. ACM IWCMC'10, Caen, France, Jun. 2010.
[28] D. Knuth, "The Art of Computer Programming", Vol. 4, Fascicle 3: "Generating All Combinations and Partitions", Addison-Wesley Professional, 2005.
[29] W. Saad, Z. Han, M. Debbah, A. Hjorungnes and T. Basar, "Coalitional game theory for communication networks", in IEEE Signal Processing Magazine, vol. 26, no. 5, pp. 77-97, September 2009.
[30] Thomas S. Ferguson, "Game Theory, Second Edition", 2014, Mathematics Department, UCLA, http://www.math.ucla.edu/~tom/Game_Theory/Contents.html.
[31] Networkx, "http://networkx.lanl.gov".