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

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

*Published in:*
IEEE Journal on Selected Areas in Communications

*DOI:*
10.1109/JSAC.2018.2815398

*Published: 12/03/2018*

*Document Version*
Peer reviewed version

*Please cite the original version:*
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, software defined networking, and cloud computing, enabling the concept of mobile carrier cloud. The key idea behind this concept, also known as core network as a service, 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, over a federated cloud. In this vein, an efficient VNF placement algorithm is highly sustained to service quality of traffic (QoS) while reducing the deployment cost. Our contribution in this paper is twofold. First, we devise an algorithm that derives the optimal number of virtual instances of 4G (MME, SGW, and 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, the second is based on coalition formation game, wherein the aim is to build coalitions of cloud networks 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 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 for a specific traffic pattern. Let \( \vartheta \) denote a VNF, whereby a set of different \( \vartheta \) would compose the vEPC/5G Core instance. Formally, \( \vartheta \) 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 \( \vartheta \) (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 \( \vartheta \)) 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 \( \vartheta \) 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.

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. Rabbani et al. [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

---

1 It shall be noted that whilst the focus, throughout this paper, is on virtual EPC or EPCaaS (i.e., EPC VNFs), the proposed solution is also applicable to the case of 5G core network functions.
by the 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]. Mashayekhy et al. [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], Ray et al. [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 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. Taleb and Ksentini [20] 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 VN deployment 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. Yousef et al. [22] 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 CNs. The proposed solution specifies the number and locations of different instances of VNFs of virtual EPC, in each 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 CNs belong to the same cloud operator, which is relaxed in our proposed solution (i.e., CNs 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) CNs whereby each 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 CNs, and (iii) a Communication Service Management Function (CSMF) that is responsible for: a) translating the communication service related requirements into network slice related requirements, and b) communication with NSFM.

For enabling the VIM functionality, CNs 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 CN are defined through a set of flavors $L^C_N$, whereby each flavor $\ell \in L^C_N$ 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 $N$. The Service Instance Manager (SIM) enables NSFM and NSSFM to orchestrate, monitor and manage a set of VNFs that belong to different CNs. 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 \( \vartheta \) (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.

Fig. 2 illustrates the main overview of the envisioned EPCaaS framework, dubbed virtual-EPC throughout this paper. Fig. 2(a) depicts a network architecture that consists of: (i) three CNs 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 \( \vartheta \), their flavors, as well as their locations in different CNs. In this figure, we present only how to instantiate vEPC, however, following the same fashion, the virtual 5G core network slice can also be instantiated. Moreover, virtual-EPC will assign for each TA the different instances of different VNFs \( \vartheta \) 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 CNs, for instantiating the VNFs. In this figure, the square and
TABLE I
NOTATIONS USED IN THE PAPER

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CN$</td>
<td>A Cloud Network.</td>
</tr>
<tr>
<td>$\mathcal{T}$</td>
<td>The set of events that can occur in a network.</td>
</tr>
<tr>
<td>$\Omega$</td>
<td>The set of all tracking areas.</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>The set of all $CN$’s in the network.</td>
</tr>
<tr>
<td>$\mathcal{V}$</td>
<td>The set of VNFs that would be deployed to build a vEPC/SG Core. Formally, $\mathcal{V} = {\text{MME}, \text{HSS}, \text{SGW}, \text{PGW}}$.</td>
</tr>
<tr>
<td>$\vartheta \in \mathcal{V}$</td>
<td>A VNF that would be deployed to build a vEPC/SG Core. Formally, $\vartheta$ can be MME, HSS, SGW, or PGW.</td>
</tr>
<tr>
<td>$\mathcal{T}$</td>
<td>The time in discrete format, where each element $t \in \mathcal{T}$ represents the occurrence time of one or multiple events.</td>
</tr>
<tr>
<td>$\Gamma^x_A$</td>
<td>An array that shows the number of cumulative events $x \in \mathcal{T}$ initiated from TA $A$ during $\mathcal{T}$. For each $t \in \mathcal{T}$, $\Gamma^x_A[t]$ represents the number of cumulative events $x \in \mathcal{T}$ initiated from TA $A$ during the time $t$.</td>
</tr>
<tr>
<td>$\Phi^x_{A,B}$</td>
<td>An array that shows the number of cumulative events $x \in \mathcal{T}$ that would be removed if TAs $A$ and $B$ are served by the same instance of VNF during $\mathcal{T}$. For each $t \in \mathcal{T}$, $\Phi^x_{A,B}[t]$ represents the number of cumulative events $x \in \mathcal{T}$ that would be removed if TAs $A$ and $B$ are served by the same instance of VNF during the time $t$.</td>
</tr>
<tr>
<td>$\lambda_{x, \vartheta}$</td>
<td>The amount of CPU and memory resources required to handle one event of type $x \in \mathcal{T}$ by an instance of VNF $\vartheta \in \mathcal{V}$.</td>
</tr>
<tr>
<td>$\mathcal{L}^{CN}$</td>
<td>The set of flavors in $CN$.</td>
</tr>
<tr>
<td>$\mathcal{MAX}_{x}^{CN}$</td>
<td>The maximum CPU and memory resources that can be offered by an instance of VNF with a flavor $\ell \in \mathcal{L}^{CN}$.</td>
</tr>
<tr>
<td>$\tau_{x}^{CN}$</td>
<td>The cost of one virtual instance of flavor type $\ell \in \mathcal{L}^{CN}$.</td>
</tr>
<tr>
<td>$X_{t}$</td>
<td>A decision boolean variable that shows if a subset of TAs $\delta \subseteq \omega \subseteq \Omega$ use the same instance $i$ of a VNF $\vartheta$.</td>
</tr>
<tr>
<td>$Y_{i}$</td>
<td>A decision boolean variable that shows if an instance $i$ of VNF $\vartheta$ uses the flavor $\ell \in \mathcal{L}^{CN}$.</td>
</tr>
<tr>
<td>$Z_{\omega, \vartheta, CN}$</td>
<td>A decision boolean variable that shows if a subset of TAs $\omega \subseteq \Omega$ would be handled by a $CN$.</td>
</tr>
</tbody>
</table>

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

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 $CN$’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**

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 $\vartheta$ to handle an event $x \in \mathcal{T}$. Formally, $\lambda_{x, \vartheta}$ is defined as the amount of virtual resources needed by $\vartheta$ 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 necessarily have the same impact on a particular $\vartheta$ (MME, HSS, SGW or PGW). Moreover, some events $x \in \mathcal{T}$ do not have any impact on some $\vartheta$. 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 $\lambda_{S1-Release,HSS} = 0$. 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 $CN$ offers virtual resources with a set of flavors $\mathcal{L}^{CN}$ in terms of capacities. Each flavor would be used as a model class to instantiate the different VNFs. According to the required resources, $CN$ 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 $\mathcal{MAX}_{x}^{CN}$ the maximum traffic rate that can be handled by an instance of VNF with a flavor $\ell \in \mathcal{L}^{CN}$. 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 $CN$’s have different flavors with different costs. $\tau_{x}^{CN}$ denotes the cost of a flavor $\ell$ in a $CN$. 

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

<table>
<thead>
<tr>
<th>Event</th>
<th>Event description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach</td>
<td>When a UE attaches to the network.</td>
</tr>
<tr>
<td>Detach</td>
<td>When a UE detaches from the network.</td>
</tr>
<tr>
<td>PDN connection setup</td>
<td>When a UE connects to a PDN network.</td>
</tr>
<tr>
<td>PDN disconnection</td>
<td>When a UE disconnects from the network.</td>
</tr>
<tr>
<td>Handover</td>
<td>When a UE moves between two neighboring eNodeBs.</td>
</tr>
<tr>
<td>TA update</td>
<td>When a UE enters into a new tracking area. Such TA update can happen with or without SGW relocation.</td>
</tr>
<tr>
<td>S1-Release</td>
<td>Releasing S1 connection when a UE becomes inactive.</td>
</tr>
</tbody>
</table>
Before the execution of the proposed algorithm, the network is observed for a specific learning period $D$, when information about different events are gathered from the network. We denote by $T$ the set of events that occurred in the network during the period $D$. To facilitate the management of our system, we adopted a discrete event dynamic system (DEDS) to observe the events during the period $D$. The time is counted only for an event $e \in T$ that has occurred in the network. We denote by $t$ the time in discrete format, where each element $t \in T$ represents the time corresponding to one or multiple events. $\Omega$ denotes the set of TAs in the network. Each TA $A \in \Omega$ is responsible for a set of events that would be handled by vEPC/5G Core. The number of events of type $x \in T$, generated from a TA $A \in \Omega$ until a specific time $t \in T$, is denoted by $\Gamma^x_A[t]$. If a set of TAs is handled by the same instances of a VNF, some events would be omitted. For example, if $T \ni \Omega \ni \Omega \ni \Omega \ni \Omega$, TA service request event would not be generated, if a UE moves from $x \to y$ within $\Omega$. $\Phi^x_A,\Omega$ denotes the number of events of type $x \in T$, which would be omitted, if TAs $A, B \in \Omega$ 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 game. The system model, adopted in this paper, considers the problem whereby the $CN$'s collaborate to form vEPC/5G Core, such that their profits are increased. The profit of a $CN$, in this paper, refers to the difference between $\mathcal{V}$ and $\mathcal{V}$. $\Phi^x_A,\Omega$ denotes the number of events of type $x \in T$, which is responsible for a set of events that $A \in H$ and $\Phi^x_A,\Omega$ denotes the number of events of type $x \in T$, which is used for deploying a VNF $\vartheta$. If the QoS required in vEPC/5G Core is not ensured, no profit is gathered from holding different $CN$s, such that the individual profit of each $CN$ 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 $CN$s are the players desiring to increase their individual profits. We assume that each $CN$ 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 $\vartheta$. According to the required resources, $CN$ 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 $\vartheta$ (i.e., SGW, PGW or MME), a set of $CN$s ($\mathcal{S}$) would participate in a coalition to host all the instances of $\vartheta$. Every cloud network $CN_{\vartheta} \in \mathcal{S}$ will handle the traffic caused by a set of TAs $\omega \subseteq \Omega$. We define the handling of TAs by a cloud network $CN_{\vartheta} \in \mathcal{S}$ as a function $H : \Sigma \to \Omega$, where $H(CN_{\vartheta}) = \{\omega\}$. 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 $\vartheta$; i.e. there should be no TA $A$ that does not have an instance of the VNF $\vartheta$. Formally, $\forall A \in \Omega \implies \exists CN \in \mathcal{S} \land A \in H(CN)$. Also, the different $CN$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 TAs

---

**Algorithm 1 Algorithm of Virtual-EPC**

**Input:**
- $T$: The set of events that can occur in the network.
- $\Omega$: The set of all tracking areas.
- $\mathcal{V}$: The set of VNFs type that would be deployed in vEPC.
- $\Sigma$: The set of all data centers.

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

(Algorithm 1: Lines 6 – 8). Finally, a coalitional game is executed for every VNF $\vartheta$ to create instances of $\vartheta$ in different $CN$s. This process is continued until a specific learning period $D$. We assume that the price $P$ can be split into multiple parts, where each one would be used for deploying a VNF $\vartheta$ (i.e., HSS, MME, SGW and PGW). Let $P_{\vartheta}$ denote the price for deploying a VNF $\vartheta$. If the QoS required in vEPC/5G Core is not ensured, no profit is gathered from holding different VNFs. We assume that every $CN$ 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 $T$ (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 $CN$s to handle the traffic generated from different TAs. For every component and at each time $t \in T$, virtual-EPC computes the cumulative number of events generated from different TAs $A \in \Omega$, which allocates the VNF $\vartheta$ (Algorithm 1: Lines 3 – 5). Then, for every pair of TAs $A, B \in \Omega$ and at each time $t \in T$, virtual-EPC also computes the cumulative number of events of $\vartheta$ that would be removed, if they are handled by the same VNF.
to $CN$’s should not have any impact on QoS (i.e., reduce the quality);

2) $CN$ Profit: $H$ 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) $F(\vartheta, CN, \omega \subseteq \Omega)$: A function defined as $F : \mathcal{V} \times \mathcal{X} \times \Omega \rightarrow \mathbb{R}^+$, which returns the minimum cost to create different instances of the VNF $\vartheta$ at $CN$, in order to handle the traffic generated by a set of tracking areas $\omega$. This function ensures the use of optimal flavors $L^CN$ to create different instances with minimum cost, while equally ensuring the system functionality;

b) $G(\vartheta, S, \Omega)$: A function defined as $G : \mathcal{V} \times \mathcal{X} \times \Omega \rightarrow \mathbb{R}^+$, which returns the minimum possible cost to create different instances of the VNF $\vartheta$ in a coalition $S$ to handle the mobile traffic. Besides the optimal allocation of different TAs to different $CN$'s in the coalition, this function guarantees the use of the optimal flavors in each $CN$ to create different instances of minimum cost, while equally ensuring the system functionality;

Let $\varrho$ denote the set of TAs handled by $\vartheta$ in a specific $CN$. Therefore, $\forall \varrho \subseteq \varrho$.

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

A. Optimal TAs’ Cost Within the Same $CN$’

In this subsection, we describe the function $F(\vartheta, CN, \omega \subseteq \Omega)$. The aim of this function is to increase the profit of a $CN$ (i.e., cloud provider), while guaranteeing the system functionality (i.e., no QoS degradation). $CN$ would reduce the number of instances of $\vartheta$ as much as possible to increase its profit. In order to ensure the system functionality, the generated traffic due to all the events $\mathcal{T}$ and handled by an instance of $\vartheta$ should not exceed the instance capacity, especially during the peak hours. Let $\delta \subseteq \varrho$ denote the set of TAs handled by the same instance of $\vartheta$ in a specific $CN$. Therefore, $\forall \varrho \subseteq \varrho$. Let $\omega$ denote the weight of $\vartheta$’s weight) the amount of resources required by all TAs in $\varrho$. Formally, $\omega = \sum_{\varrho \subseteq \varrho} \omega$.

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 $\xi_\omega$ and the set of items that should be put in knapsack is the set of flavors $L^CN$, such that the $CN$ 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 $CN$ through $F$ is NP-hard, the optimal distribution of instances of a $\vartheta$ 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 $CN$. 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; $F$ 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) \( X_{\delta,i} \), which is a decision boolean variable that shows if a subset of TAs \( \delta \subseteq \omega \subseteq \Omega \) uses the same instance of \( \vartheta i \); ii) \( Y_{i,\ell} \) is a decision boolean variable that shows if the instance of \( \vartheta i \) uses the flavor \( \ell \in \mathcal{L}_{\CN} \).

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

\[
Y_{i,\ell} = \begin{cases} 
1 & \text{If the instance } i \text{ of the VNF } \vartheta \text{ uses the flavor type } \ell \\
0 & \text{Otherwise}
\end{cases}
\]

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 \( \vartheta \) that would be deployed in the network should not exceed \(|\omega|\). The function \( \mathcal{F}(\vartheta,\CN,\omega \subseteq \Omega) \) will be then formulated as follows:

\[
\min \mathcal{F}(\vartheta,\CN,\omega \subseteq \Omega) = \sum_{i=1}^{\mid\omega\mid} \sum_{\ell \in \mathcal{L}_{\CN}} r_{\CN}^{\ell} \cdot Y_{i,\ell} \tag{2}
\]

\[
\text{s.t, } \forall A \in \omega: \sum_{\forall \delta \in \mathcal{P}(\omega), A \in \delta} x_{\delta,i} = 1 \tag{3}
\]

\[
\forall \ell \in \mathcal{L}_{\CN}, \forall \delta \in \mathcal{P}(\omega), i = 1 \cdots |\omega|: \sum_{\delta \in \mathcal{P}(\omega)} x_{\delta,i} \leq M A \cdot x_{\CN}^{\ell} \cdot Y_{i,\ell} \tag{4}
\]

\[
\forall \delta \in \mathcal{P}(\omega), i = 1 \cdots |\omega|: x_{\delta,i} \in \{0, 1\} \tag{5}
\]

\[
\forall \ell \in \mathcal{L}_{\CN}, i = 1 \cdots |\omega|: Y_{i,\ell} \in \{0, 1\} \tag{6}
\]

In this optimization problem, we aim at minimizing as much as possible the cost spent by \( \CN \) to increase its profit. In Objective (2), we aim at using the minimum number of instances of \( \vartheta \), 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 \( \vartheta \) in \( \CN \), where every instance of the VNF \( \vartheta \) serves only one TA. However, we may not need all of them. If an instance of \( \vartheta i \) is not needed, the optimization problem \((2)-(6)) will set the variable \( Y_{i,\ell} \) to 0, \( \forall \ell \in \mathcal{L}_{\CN} \). Constraint (3) ensures the system functionality: Every TA \( A \) should be handled by only one instance of \( \vartheta \). In (3), \( x_{\delta,i} = 1 \), if the power set \( \delta \in \mathcal{P}(\omega) \) is handled by the instance of \( \vartheta i \). In (3), we ensure that every TA \( A \in \omega \) should belong to only one \( \delta \in \mathcal{P}(\omega) \), which is in turn handled only by one machine \( i (x_{\delta,i} = 1) \). Constraint (4) makes a relation between variables \( x_{\delta,i} \) and \( Y_{i,\ell} \). Moreover, it ensures that if a power set \( \delta \in \mathcal{P}(\omega) \) is selected \( (x_{\delta,i} = 1) \), then the best flavor \( \ell \) should be used in its VNF. The flavor \( \ell \) should have the lowest cost while offering the required resources by \( \delta \). Formally, if \( x_{\delta,i} = 0 \), then \( Y_{i,\ell} \) should be also equal to 0 in order to reduce the cost (2). Meanwhile, Constraints (5) and (6) ensure that \( x_{\delta,i} \) and \( Y_{i,\ell} \) are decision Boolean variables.

\section{B. Optimal TAs’ Cost Within a Coalition of \( \CN \)’s}

In this subsection, we will show how to create a set of instances of \( \vartheta \) in a coalition of \( \CN \)’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 \( \CN \)’s is increased. The function \( \mathcal{G}(\vartheta,\CN,\omega) \) 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 \( Z_{\omega,\CN} \), which is a decision Boolean variable that shows if a subset of TAs \( \omega \subseteq \Omega \) is handled by a \( \CN \).

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

The function \( \mathcal{G}(\vartheta,\CN,\omega) \) can be formulated as follows:

\[
\min \mathcal{G}(\vartheta,\CN,\omega) = \sum_{\forall \omega \in \mathcal{P}(\Omega)} \sum_{\forall \CN \in S} Z_{\omega,\CN} = 1 \tag{7}
\]

\[
\text{s.t, } \forall A \in \omega: \sum_{\forall \CN \in S} \mu_{\CN} \cdot Z_{\omega,\CN} \geq 1 \tag{9}
\]

\[
\forall \omega \in \mathcal{P}(\Omega), \forall \CN \in S: Z_{\omega,\CN} \in \{0, 1\} \tag{10}
\]

In this optimization problem, we aim to reduce the cost incurred by a coalition \( S \) of \( \CN \)’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 \( \CN \) a set of TAs \( \omega \), 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 \( A \) should be handled by only one \( \CN \). Meanwhile, Constraint (9) ensures that every \( \CN \) in the coalition should handle at least one power set \( \omega \) from \( \Omega \). A \( \CN \) would not get a profit from \( S \), if it does not handle a set of TAs from \( \Omega \). Constraint (10) assures that \( Z_{\omega,\CN} \) is a decision Boolean variable.

\textbf{Theorem 2: Achieving optimal profit through } \mathcal{G} \text{ is also NP-hard problem.}

\textbf{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. \hfill \Box

\section{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 CNs involved in this operation. We define the characteristic function of the coalitional game as follows:

$$v(S) = \begin{cases} 0 & \text{if } |S| = 0 \text{ or QoS is not ensured.} \\ p_\theta - G(\theta, S, \Omega) & \text{otherwise} \end{cases} \quad (11)$$

where $p_\theta$ is the price for deploying VNF $\theta$. The QoS is ensured in $S$ if the following condition is true:

$$\forall CN \in S, \forall \omega \in \mathcal{P}(\Omega) : Z_{\omega, CN} = 1 \implies \forall A \in \omega : \theta_{A, CN} > \theta_{Th}$$

(12)

where $\theta_{A, CN}$ represents the data rate between TA $A$ and CN $CN$, and $\theta_{Th}$ 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 CNs. Thus, for every VNF $\theta$, a coalition of CNs $S$ would be determined, where the different instances of $\theta$ would be built. Every CN $CN \in \Sigma$ 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 $v(S)$ among the different players (CNs) in the same coalition $S$. 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. An easy way to share the profit among players is the use of equal sharing method to define the payoff of different players among different players is the use of equal sharing method. 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. Another way to share the profit among players is the use of equal sharing method.

To define the core, we need to define the unstable imputation. An imputation is unstable through a coalition $S$ iff $\sum_{CN \in S} \Pi_{\Sigma}(CN) \leq v(S)$. This means that there exists a coalition $S$, where its players CNs will gain more if they collaborate together than acting within the grand coalition $\Sigma$. The core is the set of the payoff vectors that leads the players to agree to form the grand coalition $\Sigma$. 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 $\Sigma$. Otherwise, the grand coalition would not be formed and the VNF $\theta$ would be created only using the coalition of CNs 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 CNs, dubbed CN1, CN2, and CN3, that compete to host the VNF $\theta$. We also consider that the RAN is formed with two TAs $A$ and $B$. To facilitate the explanation of the example, let $v(CN_i, \omega)$ denote the profit of each CN $CN_i$ to handle a set of TAs $\omega$. To compute $v(CN_i, \omega)$, similarly to $v(S)$, we compute the different $\xi$, $\mathcal{F}$ and then $\mathcal{G}$. We formally define $v(CN_i, \omega)$ as follows:

$$v(CN_i, \omega) = \begin{cases} 0 & \text{if } |S| = 0 \text{ or QoS is not ensured.} \\ p_\theta - G(CN_i, \omega) & \text{otherwise} \end{cases} \quad (15)$$

Based on (14), the different CNs would form different coalitions, where the profit of each CN is maximized. The profit of CNs in each coalition $S$ 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 CNs $\Sigma$ denotes the grand coalition. The payoff vector of the grand coalition is defined as follows: $\Pi_{\Sigma} = (\Pi_{\Sigma}(CN_1), \ldots, \Pi_{\Sigma}(CN_{|\Sigma|}))$, where $\Pi_{\Sigma}$ refers to the $i^{th}$ $CN$ in $\Sigma$. The vector payoff $\Pi_{\Sigma}$ 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 $\Sigma$. 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 CNs in the grand coalition should be equal to $v(\Sigma)$.

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

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

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), $\Pi_{\Sigma}$ is individual rationality if $\forall CN \in \Sigma : v(CN) \leq \Pi_{\Sigma}(CN)$.

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

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

Definition 4: According to the payoff vector in (30), $\Pi_{\Sigma}$ is core iff it does not contain any unstable imputation. Formally, the core is defined as follows: $\sum_{CN \in S} \Pi_{\Sigma}(CN) = v(\Sigma)$ and $\forall S \subseteq \Sigma : v(S) \leq \sum_{CN \in S} \Pi_{\Sigma}(CN)$.
Table III shows the profit of each CN when handling the different TAs. The profits in this table are presented as numbers that represent the difference between $P_\theta$ 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 CN. Meanwhile, Table IV shows the different CN coalitions that can be formed, the mapping of each TA with its CN through functions $F$ and $G$, as well as $v(S)$ and $\Pi_S$. Table IV is filled from Table III using (1), the two optimization problems ((2)–(6)) and ((7)–(10)), (11) and (13), respectively.

Based on Constraint (9), every CN should handle some TAs; otherwise the other TAs in the coalition would not accept to share with it the profit. In the last row in Table IV, CN1 and CN2 would not accept to share the profit with CN3. 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 $\Sigma = \{CN_1, CN_2, CN_3\}$ can be formed. However, the coalition $S = \{CN_1, CN_2\}$ may exit the grand coalition as the profit of its players will be increased. Both players CN1 and CN2 will gain 120 instead of 80 if they agree to exit the grand coalition. As the coalition $S = \{CN_1, CN_2\}$ satisfies the following conditions $\Pi_S(CN_1) \geq v(CN_1)$ and $\Pi_S(CN_2) \geq v(CN_2)$, the instance of $\vartheta$ will be built in CN1 and CN2. According to many parameters, such as the cost of each flavor in each CN and QoS, the core may exist or not in this game.

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 $\vartheta$ one by one.

A. Coalitional Game for Building One vEPC/5G Core Instance

The coalition 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 to stabilize the grand coalition by suggesting 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 CNs 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 CNs, in order to increase their profit when forming vEPC/5G Core. This game would be repeated for every VNF $\vartheta$ 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 $\vartheta$. This function will call the coalitional game mechanism to optimally place instances of $\vartheta$. In the end of $instanceVNF(\vartheta, \Gamma, \Phi)$ execution, a partition $\Xi$ of the grand coalition $\Sigma$ should be formed. $\Xi = \{S_1, S_2, \ldots, S_K\}$, where $K$ is the number of coalitions which would be created. Each CN should belong to only one coalition and all CNs should belong to a coalition. $\Xi$ is defined through the following properties: i) $\forall S_i, S_j \in \Xi \land i \neq j \Rightarrow S_i \cap S_j = \emptyset$; ii) $\bigcup_{S \in \Xi} S = \Sigma$. The function $instanceVNF(\vartheta, \Gamma, \Phi)$ will return a set of coalitions, where the VNF $\vartheta$ would be deployed. virtual-EPC, via bestCoalition($\Xi$), 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 $C = \{S_1, \ldots, S_K\}$. $C$ is collection if it satisfies the following condition: $\forall S_i, S_j \in C \land i \neq j \Rightarrow S_i \cap S_j = \emptyset$. If $\bigcup_{S \in \Xi} S = \Sigma$, then $C$ is called a partition of the grand coalition $\Sigma$.

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 $\triangleright$ the comparison relation between two partitions $C_1$ and $C_2$, that partition the same set $\eta$ of CNs. Formally, $\bigcup_{S \in C_1} S = \bigcup_{S \in C_2} S = \eta$. We mean by $C_1 \triangleright C_2$ “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 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 $\vartheta$ later. In what follows, we will define the
merge and split rules of our game. We consider two collections \( C_s = \{S_1, S_2, \ldots, S_K\} \) and \( C_m = \bigcup_{i=1}^{K} S_i \). \( C_s \) 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 \( CN \)'s. We define two comparison relations \( \triangleright_m \) and \( \triangleright_s \) for the merge and split, respectively. Note that \( CN \)'s are selfish as each of them aims to increase its payoff without caring about the other players. The merge \( \triangleright_m \) and split \( \triangleright_s \) relations are defined as follows:

\[
C_m \triangleright_m C_s \iff \{\forall S \in C_s, \forall CN \in S : \\
\Pi_{C_m}(CN) \geq \Pi_{C_s}(CN) \} \\
\land (\exists S' \in C_m, \exists CN \in S' : \\
\Pi_{C_m}(CN) > \Pi_{S'}(CN)) \}
\]

(16) \[
C_s \triangleright_s C_m \iff \{\exists S \in C_s : \\
(\forall CN \in S : \Pi_{S}(CN) \geq \Pi_{C_m}(CN) \\
\land \exists CN \in S : \Pi_{C_m}(CN) > \Pi_{C_s}(CN)) \}
\]

(17)

From (16), \( C_m \) is preferred than \( C_s \) iff at least one player \( CN \) 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, Eq. (17) shows that \( C_s \) will be preferred than \( C_m \), iff there is at least one coalition \( S \in C_s \), 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 \( S \) from \( C_m \) does not take into account the waste of the other players in \( C_m - S \). Based on the split and merge comparison relations, in our game, we propose two rules for the merge and split processes:

- **Merge Rule:** The function instanceVNF(\( \theta, \Gamma, \Phi \)) will merge any collection of coalitions \( C_s = \{S_1, S_2, \ldots, S_K\} \) to \( C_m \) iff \( C_m \triangleright_m C_s \).
- **Split Rule:** The function instanceVNF(\( \theta, \Gamma, \Phi \)) will split a coalition \( C_m \) to \( C_s = \{S_1, S_2, \ldots, S_K\} \) iff \( C_s \triangleright_s C_m \).

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 \( S \) will leave \( C_m \) iff at least one player in \( S \) enhances its payoff, while the payoffs of the other players in \( S \) are not affected and without taking into account the payoffs of the other players in \( C_m - S \). In the end of the instanceVNF(\( \theta, \Gamma, \Phi \)) function, a collection of coalitions will be returned. Using the procedure bestCoalition(\( \Xi \)), the best coalition from this collection will be selected to host the instances of \( \theta \) later.

In what follows, we will show how the merge and split processes operate. We consider two coalitions \( S_1 \) and \( S_2 \), where \( S_1 \cap S_2 = \emptyset \). Based on (13) and (16), \( S_1 \) and \( S_2 \) would be merged into \( S_m = \{S_1, S_2\} \) iff the following conditions are fulfilled:

1) The two following conditions are correct:
   a) \( \forall CN \in S_1 : \Pi_{S_1}(CN) \geq \Pi_{S_2}(CN) \)
   b) \( \forall CN \in S_2 : \Pi_{S_2}(CN) \geq \Pi_{S_1}(CN) \)

2) One of the following conditions is correct:
   a) \( \exists CN \in S_1 : \Pi_{S_1}(CN) > \Pi_{S_2}(CN) \)
   b) \( \exists CN \in S_2 : \Pi_{S_2}(CN) > \Pi_{S_1}(CN) \)

The instanceVNF(\( \theta, \Gamma, \Phi \)) function will merge two coalitions \( S_1 \) and \( S_2 \) iff at least the profit of one player in this coalition will increase while the profit of all the other players will remain unaffected. For the split mechanism, a coalition \( S_m = \{S_1, S_2\} \) would split into two coalitions \( S_1 \) and \( S_2 \), iff the conditions in (17) are met. \( S_1 \) (resp., \( S_2 \)) will split from \( S_m \), iff at least one player in \( S_1 \) (resp., \( S_2 \)) enhances its payoff while the payoffs of the other players in \( S_1 \) (resp., \( S_2 \)) are not affected. Formally, \( S_m = \{S_1, S_2\} \) will split into two coalitions \( S_1 \) and \( S_2 \) iff one of the following conditions are fulfilled:

1) The two following conditions are correct:
   a) \( \forall CN \in S_1 : \Pi_{S_1}(CN) \geq \Pi_{S_m}(CN) \)
   b) \( \exists CN \in S_1 : \Pi_{S_m}(CN) > \Pi_{S_m}(CN) \)

2) The two following conditions are correct:
   a) \( \forall CN \in S_2 : \Pi_{S_2}(CN) \geq \Pi_{S_m}(CN) \)
   b) \( \exists CN \in S_2 : \Pi_{S_2}(CN) > \Pi_{S_m}(CN) \)

The instanceVNF(\( \theta, \Gamma, \Phi \)) function will split any coalition \( S_m \) 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(\( \theta, \Gamma, \Phi \)) will hold the instances of \( \theta \). The split process does not affect the best coalition, as a coalition \( S \) would split from another coalition \( S_m \) 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 \( \Xi \) is ID\( \rho \)-stable if no player has the intention to leave its respective coalition. ID\( \rho \)-stable can be also defined as the set of coalitions in \( \Xi \) that do not have the intention to merge or split any further.

### B. Algorithm Description for Building One vEPC/CSG Core Instance

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

The function instanceVNF(\( \theta, \Gamma, \Phi \)) first starts by forming a collection \( \Xi \) by putting every player \( CN \) 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\( \rho \)-stable collection (Algorithm 2: Lines 3–44).
Algorithm 2  \text{instanceVNF}(\vartheta, \Gamma, \Phi) \\
\textbf{Input:} \\
\quad \vartheta: A component of vEPC. \\
\quad \Gamma: The number of cumulative events. \\
\quad \Phi: The number of cumulative events would be omitted. \\
1: \Xi = \{\{DC_1\}, \{DC_2\}, \ldots, \{DC_{|\Xi|}\}; \\
2: \text{visited} = \emptyset; \\
3: \textbf{while} True \textbf{do} \\
4: \quad \text{stop} = True; \\
5: \quad \textbf{for all} S \notin \Xi \textbf{do} \\
6: \quad \quad \text{if} S \in \Psi \textbf{then} \\
7: \quad \quad \quad \Psi[S] = v(S) \\
8: \quad \end{cases} \\
9: \quad \textbf{end if} \\
10: \quad \text{// Merging process} \\
11: \quad \textbf{for all} S_i, S_j \in \text{combinations}(\Xi, 2) \textbf{visited} \textbf{do} \\
12: \quad \quad \text{visited = visited} \cup \{(S_i, S_j)\}; \\
13: \quad \quad \text{if} \{S_i \cup S_j\} \notin \Psi \textbf{then} \\
14: \quad \quad \quad \Psi[S_i \cup S_j] = v(S_i \cup S_j) \\
15: \quad \quad \textbf{end if} \\
16: \quad \quad \text{// Using } \Psi \text{ the values of } \Pi_{S_i, S_j} \text{ and } \Pi_{S_i \cup S_j} \text{ are computed} \\
17: \quad \quad \textbf{if} \{S_i \cup S_j\} \succ_m \{(S_i), (S_j)\} \textbf{then} \\
18: \quad \quad \quad \Xi = \Xi \setminus \{S_i\}; \Xi = \Xi \cup \{S_j\}; \Xi = \Xi \cup \{S_i \cup S_j\}; \\
19: \quad \quad \quad \text{stop = False}; \\
20: \quad \quad \quad \text{break}; \\
21: \quad \quad \textbf{end if} \\
22: \quad \text{// Split process} \\
23: \quad \textbf{for all} S \in \Xi \land |S| > 1 \textbf{do} \\
24: \quad \quad \text{break = False}; \\
25: \quad \quad \textbf{for all} \{S_i, S_j\} \in S \land S_i \cup S_j = S \land S_i \cap S_j = \emptyset \textbf{do} \\
26: \quad \quad \quad \text{if} S_i \notin \Psi \textbf{then} \\
27: \quad \quad \quad \quad \Psi[S_i] = v(S_i) \\
28: \quad \quad \quad \textbf{end if} \\
29: \quad \quad \quad \text{if} S_j \notin \Psi \textbf{then} \\
30: \quad \quad \quad \quad \Psi[S_j] = v(S_j) \\
31: \quad \quad \quad \textbf{end if} \\
32: \quad \quad \text{// Using } \Psi \text{ the values of } \Pi_{S_i, S_j} \text{ and } \Pi_S \text{ are computed} \\
33: \quad \quad \textbf{if} \{(S_i), (S_j)\} \succ_m S \textbf{then} \\
34: \quad \quad \quad \Xi = \Xi \setminus \{S_i\}; \Xi = \Xi \cup \{S_i\}; \Xi = \Xi \cup \{S_j\}; \\
35: \quad \quad \quad \text{stop = False}; \\
36: \quad \quad \quad \text{break = True}; \\
37: \quad \quad \quad \text{break}; \\
38: \quad \quad \textbf{end if} \\
39: \quad \quad \textbf{end for} \\
40: \quad \text{if} \text{break = True} \textbf{then} \\
41: \quad \quad \text{break}; \\
42: \quad \textbf{end if} \\
43: \textbf{end for} \\
44: \quad \textbf{if} \text{stop = True} \textbf{then} \\
45: \quad \text{break}; \\
46: \textbf{end while} \\
47: \textbf{return} \Xi.

instanceVNF(\vartheta, \Gamma, \Phi) \text{ computes the values of } v(S) \text{ using (11) and ensures that the function } \Theta \text{ is executed only when needed (Algorithm 2: Lines 5 – 9). To prevent the redundancy in the computation, the vector } \Psi \text{ is used to store the values of } v(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_{\nu}-stable collection. In the merging process, every pair of coalitions \(S_i\) and \(S_j\), 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 \(\Theta\) twice, the value of \(v(S_i \cup S_j)\) will be put in the vector \(\Psi\) (Algorithm 2: Lines 12 – 14). If the merging condition ((\(\{S_i \cup S_j\} \succ_m \{(S_i) \cup (S_j)\}\)) 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 \(S\) that has more than one player \(CN\) (Algorithm 2: Line 21). We try to split every two sub-coalitions \(S_i\) and \(S_j\) of \(S\) that satisfy the following conditions: i) \(S_i \cup S_j = S\); ii) \(S_i \cap S_j = \emptyset\) (Algorithm 2: Line 23). The partitioning of the coalition \(S\) is done through the partitioning of an integer into two parts [28]. For example, the coalition \(\{CN_1, CN_2, CN_3\}\) can be presented with a number 7 (i.e., 111), whereas the coalitions \(\{CN_1, CN_3\}\) and \(\{CN_2\}\) would be presented with the numbers 5 (i.e., 101) and 2 (i.e., 010), respectively. Enumerating all the possible two sub-coalitions of \(S\) 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 \(S\). Using the same approach, the redundancy in the computation of \(\Theta\) is also prevented in the split process (Algorithm 2: Lines 24 – 29). Then, the function instanceVNF(\vartheta, \Gamma, \Phi) splits \(S\) if it is better for the collection \(\Xi\) (Algorithm 2: Lines 30 – 35). If one split succeeds, we exit the split process and re-initiate the merge process. Note that the variable \text{stop} will be set to \text{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_{\nu}-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 \(\Xi\). As mentioned in Table III, the network consists of three \(CN\)‘s \(CN_1\), \(CN_2\) and \(CN_3\). The E-UTRAN is formed in two TAs \(A\) and \(B\). The function instanceVNF(\vartheta, \Gamma, \Phi) starts the execution by putting every player \(CN\) in a separate coalition \(\Xi = \{(CN_1)\}, \{CN_2\}, \{CN_3\}\). Based on Algorithm 2, the merge process starts first. From Table IV, we have \(\{CN_1, CN_3\} \succ_m \{(CN_1), \{CN_3\}\}\), since both players will increase their pay-offs. Then, the collection \(\Xi\) will be updated as follows: \(\Xi = \{(CN_1, CN_3), \{CN_2\}\}\). In this case, \(\Xi\) is composed
of two coalitions. In the split process, as the condition \( \{CN_1\}, \{CN_3\} \triangleright_s \{CN_1, CN_3\} \) is not verified, the split process is omitted in this iteration. As the collection \( \Xi \) is not ID\(_p\)-stable, we re-initiated the merging process. Now, we have \( \{CN_1, CN_3, CN_2\} \triangleright_m \{\{CN_1, CN_3\}, \{CN_2\}\} \), as both players \( CN_1 \) and \( CN_3 \) will increase their profits while the profit of player \( CN_2 \) remains the same. Applying this merge rule will impact the collection \( \Xi \), which will be updated to contain only one coalition \( \Xi = \{CN_1, CN_2, CN_3\} \). In the split process, we have \( \{\{CN_1, CN_2\}, \{CN_3\}\} \triangleright_s \{CN_1, CN_3, CN_2\} \) as the profit of both players in the coalition \( \{CN_1, CN_2\} \) is increased to 120 instead of 80. Then, the coalition \( \{\{CN_1, CN_2\}, \{CN_3\}\} \) will split into two coalitions and then the collection \( \Xi \) is updated as \( \Xi = \{\{CN_1, CN_2\}, \{CN_3\}\} \). In this state, the coalition \( \{CN_1, CN_2\} \) 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 \( \Xi \) are not able to merge or split (no player has the intention to exit its respective coalition), \( \Xi \) is ID\(_p\)-stable. \( \text{instanceVNF}(\vartheta, \Gamma, \Phi) \) will return the collection \( \Xi \), and the coalition \( \{CN_1, CN_2\} \) will be accordingly selected to hold the instances of \( \vartheta \) as it has the highest payoff value.

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

**Proof:** The merge and split processes are repetitively called in the function \( \text{instanceVNF}(\vartheta, \Gamma, \Phi) \) 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 \( \text{instanceVNF}(\vartheta, \Gamma, \Phi) \) 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 \( \text{instanceVNF}(\vartheta, \Gamma, \Phi) \) 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 \( S_i \) and \( S_j \) be two coalitions in the collection \( \Xi \), and \( S_i \) and \( S_j \) would be merged into \( S = S_i \cup S_j \) if \( S \triangleright_m \{S_i, S_j\} \). In order that the above condition is correct (16), there should exist at least one player \( CN \in S_i \) (resp., \( CN \in S_j \)) that enhances its payoff in \( S \); \( \Pi_S(CN) > \Pi_{S_i}(CN) \) (resp., \( \Pi_S(CN) > \Pi_{S_j}(CN) \)). Based on the observation that the proposed algorithm moves from a collection to a better collection after one or multiple merge and split operations, \( S \) cannot split to the coalitions \( S_i \) and \( S_j \). Based on (17), if \( S_i \) and \( S_j \) are merged into \( S \) then \( \{S_i, S_j\} \triangleright_s S \) would be not verified. We argue this by the fact that there is at least one player \( CN \in S_i \) (resp., \( CN \in S_j \)), where \( \Pi_S(CN) > \Pi_{S_i}(CN) \) (resp., \( \Pi_S(CN) > \Pi_{S_j}(CN) \)). Then, no cycle in the merge and split process would be created. This means that the function \( \text{instanceVNF}(\vartheta, \Gamma, \Phi) \) 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 CN’s. As comparison terms, we use two trivial solutions. The first one, named G-EPC, uses the Grand coalition \( \Sigma \) for the deployment of vEPC/5G Core. In this solution, all CN’s participate in creating different instances of VNF \( \vartheta \), whereas the second solution uses a random deployment over CN’s. In this solution, a greedy algorithm is used, whereby different CN’s are randomly selected one by one, as well as the instantiations of VNFs are randomly created one by one, until the traffic generated from \( \Omega \) is handled. Only then, the greedy algorithm is finished and the random coalition of CN’s is selected. The algorithms are evaluated in terms of the following metrics:

- **Payoff of Individual CN:** is defined as the average value of individual payoffs for each player in the selected coalitions of different instances of each VNF \( \vartheta \);
- **Number of Merge and Split:** is defined as the average number of merge and split operations needed to deploy each VNF \( \vartheta \). This metric shows the complexity of the proposed scheme virtual-EPC;
- **Number of CN’s in the Selected Coalition:** is defined as the average number of players in the selected coalition for each instance of each VNF \( \vartheta \).

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 CN’s. We conduct two sets of experiments. Firstly, we vary the number of TAs and fix the number of CN’s to 15. In the second scenario, we vary the number of 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 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 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 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 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 solution 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 $F(\vartheta, CN, \omega \in \Omega)$, and consequently the more split and merge operations.

Fig. 4 depicts the performance of the three solutions for varying numbers of players $CN$s. In Fig. 4(a), we plot the evaluation of individual payoff of each $CN$. 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 $CN$s. When the number of $CN$s exceeds 12, $O$-$EPC$ outperforms $G$-$EPC$. This can be explained by the fact that when the number of $CN$s is less than 12 (the ratio of TAs per $CN$ is high), then the players in the selected coalitions equitably participate to handle the different TAs. Whereas when the ratio of TAs per $CN$ decreases, many $CN$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 $CN$s is 30, the proposed scheme outperforms the baseline approaches with more than 233%. Fig. 4(b) shows that the number of $CN$s in the selected coalitions increases proportionally with the number of $CN$s in the network. The higher the number of $CN$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 $CN$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 NFV, SDN, and cloud computing. In this paper, we developed a new framework for building virtual core network slices as a service (e.g., EPCaaS). The aim of this framework is the placement of VNFs of virtual EPC/5G Core in an efficient way over a federated $CN$. 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 a coalitional game to place these instances across different $CN$s, such that the QoS is ensured and the profit of
each $CN$ is maximized. The simulation results demonstrate the efficiency of the proposed framework in achieving its key design objectives.

REFERENCES


Miloud Bagaa received the bachelor’s, master’s, and Ph.D. degrees from the University of Science and Technology Houari Boumediene Algeria, in 2005, 2008, and 2014, respectively. From 2009 to 2015, he was a Researcher with the Research Center on Scientific and Technical Information, Algeria, where he was a member of the Wireless Sensor Networks Team. From 2015 to 2016, he held a post-doctoral fellowship position at the European Research Consortium for Informatics and Mathematics and also worked with the Norwegian University of Science and Technology, Trondheim, Norway. He is currently a Senior Researcher with the Communications and Networking Department, Aalto University. His research interests include wireless sensor networks, the Internet of Things, 5G wireless communication, security, and networking modeling.

Tarak Taleb received the B.E. degree (Hons.) in information engineering and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Japan, in 2001, 2003, and 2005, respectively. He is currently a Professor with the School of Electrical Engineering, Aalto University, Finland. He was an Assistant Professor with the Graduate School of Information Sciences, Tohoku University, until 2009, in a lab fully funded by KDDI. He was a Senior Researcher and a 3GPP Standards Expert with NEC Europe Ltd., Heidelberg, Germany. From 2005 to 2006, he was a Research Fellow with the Intelligent Cosmos Research Institute, Sendai, Japan.

His research interests lie in the field of architectural enhancements to mobile core networks, network softwarization and slicing, mobile cloud networking, network function virtualization, software defined networking, mobile multimedia streaming, inter vehicular communications, and social media networking. He has also been directly involved in the development and standardization of the evolved packet system as a member of 3GPP’s System Architecture working group. He is a member of the IEEE Communications Society Standardization Program Development Board. He founded the IEEE Workshop on Telecommunications Standards from Research to Standards, a successful event that received Best Workshop Award from IEEE Communication Society. Based on the success of this workshop, he has also founded and has been the Steering Committee Chair of the IEEE Conference on Standards for Communications and Networking.

He is the General Chair of the 2019 edition of the IEEE Wireless Communications and Networking Conference to be held in Marrakech, Morocco. He is/was on the Editorial Board of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE WIRELESS COMMUNICATIONS Magazine, the IEEE INTERNET OF THINGS JOURNAL, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and a number of Wiley journals. From 2006 to 2010, he also served as the Vice Chair of the Satellite and Space Communications Technical Committee of IEEE ComSoc. Until 2016, he served as the Chair of the Wireless Communications Technical Committee, the largest in IEEE ComSoc. He has been on the technical program committee of different IEEE conferences, including Globecom, ICC, WCNC, and chaired some of their symposia. He is the Guest Editor-in-Chief of the IEEE JSAC Series on Network Softwarization and Enablers.
He was a recipient of the 2017 IEEE ComSoc Communications Software Technical Achievement Award in 2017 for his outstanding contributions to network softwarization. He is also a co-recipient of the 2017 IEEE Communications Society Fred W. Ellersick Prize, the 2009 IEEE ComSoc Asia-Pacific Best Young Researcher Award, the 2008 TELECOM System Technology Award from the Telecommunications Advancement Foundation, the 2007 Funai Foundation Science Promotion Award, the 2006 IEEE Computer Society Japan Chapter Young Author Award, the Niwa Yasujirou Memorial Award in 2005, and the Young Researcher’s Encouragement Award from the Japan Chapter of the IEEE Vehicular Technology Society in 2003. Some of his research works have also received best paper awards at prestigious IEEE-flagged conferences.

Abdelquoddous Laghrissi received the bachelor’s degree in mathematics and computer science and the master’s degree in applied computer science from the School of Sciences, Mohamed V University, in 2012 and 2014, respectively. He is currently pursuing the Ph.D. degree with the School of Electrical Engineering, Aalto University, Finland. His master’s dissertation was on empathy in vehicular ad hoc networks. From 2014 to 2015, he was a Volunteer Member of the cloud computing working group within a Euro-Mediterranean H2020 Project—Cooperation with Mediterranean Partners to build Opportunities around ICT and Societal And Industrial Challenges—of Horizon 2020, and he is currently involved in a European project on network slicing and 5G in the context of H2020 project 5G/Pagoda, during which he authored articles and contributed to several deliverables. He is member of the MOSAIC Lab. His research interests include network softwarization and slicing mechanisms, mobile cloud computing, network function virtualization, software defined networking, and their modeling and simulation.

Adlen Ksentini is currently a COMSOC Distinguished Lecturer. He received the Ph.D. degree in computer science from the University of Cergy-Pontoise in 2005. His Ph.D. dissertation was on QoS provisioning in IEEE 802.11-based networks. From 2006 to 2016, he was an Assistant Professor with the University of Rennes I and also a member with the Dionysos Team, INRIA, Rennes. Since 2016, he has been an Assistant Professor with the Communication Systems Department, EURECOM. He has been involved in several national and European projects on QoS and QoE support in future wireless, network virtualization, cloud networking, mobile networks, and more recently on network slicing and 5G in the context of H2020 projects 5G/Pagoda and 5GTransformer. He has co-authored over 100 technical journal and international conference papers. He received the Best Paper Award from the IEEE IWCMC 2016, the IEEE ICC 2012, and ACM MSWiM 2005. He has received the 2017 IEEE Comsoc Fred W. Ellersick for the Best IEEE Communications Magazine paper.

He has given several tutorials in the IEEE international conferences, the IEEE Globecom 2015, the IEEE CCNC 2017, the IEEE ICC 2017, and the IEEE/IFIP IM 2017. He was the TPC Symposium Chair of the IEEE ICC 2016/2017, the IEEE GLOBECOM 2017, the IEEE Cloudnet 2017 and the IEEE 5G Forum 2018. He has been a Guest Editor of the IEEE Journal of Selected Area on Communications Series on Network Softwarization, the IEEE Wireless Communications, IEEE Communications Magazine, and two issues of ComSoc MMTC Letters. He has been on the Technical Program Committees of the major IEEE ComSoc, ICC/GLOBECOM, ICME, WCNC, and PIMRC conferences. He is currently the Vice-Chair of the IEEE COMSOC Technical Committee on Software and the Secretary of the IEEE COMSOC France.

Hannu Flink received the M.Sc. and Lic. Tech. degrees in computer science and communication systems from the Helsinki University of Technology, Aalto University, Finland, in 1986 and 1993, respectively. He is currently a Research Manager with Nokia Bell Labs, Espoo, Finland. His current research topics include 5G network architecture, IP protocols, and virtualization.