Efficient Virtual Evolved Packet Core Deployment Across Multiple Cloud Domains

Miloud Bagaa*, Tarik Taleb*, Abdelquoddouss Laghrissi* and Adlen Ksentini§

* Communications and Networking Department, Aalto University, Finland.
§ Communication systems, EURECOM, Sophia-Antipolis, France.
Emails: miloud.bagaa@aalto.fi, talebtarik@ieee.org, abdelquoddouss.laghrissi@aalto.fi, adlen.ksentini@eurecom.fr

***Abstract*—Many ongoing research activities relevant to $5G$ mobile systems concern the virtualization of the Evolved Packet Core (EPC) elements aiming for system scalability, elasticity, flexibility, and cost-efficiency. Virtual Evolved Packet Core (vEPC) will principally rely on some key technologies, such as Network Function Virtualization (NFV), Software Defined Networking (SDN) and Cloud Computing, for enabling the concept of Mobile Carrier Cloud. The key idea beneath this concept, known also as EPC as a Service (EPCaaS), consists in deploying virtual instances (i.e., Virtual Machines or Containers) of key core network functions (i.e., Virtual Network Functions - VNF), such as the Mobility Management Entity (MME), Serving GateWay (SGW), and Packet Data network gateWay (PGW) over a federated cloud. In this vein, an efficient VNF placement algorithm is highly needed to sustain the Quality of Service (QoS) while reducing the deployment cost. Our contribution, in this paper, is to devise an algorithm that derives the optimal number and locations of vEPC's virtual instances over the federated cloud. The proposed algorithm is based on coalition formation game, wherein the aim is to build optimal coalitions of Cloud Networks ($\mathcal{CN}$s) to host the virtual instances of the vEPC elements. The obtained results clearly indicate the advantages of the proposed algorithm in ensuring QoS given a fixed cost for vEPC deployment, while maximizing the profits of cloud operators.**

I. Introduction

NFV represents one of the key enablers of the next generation mobile network systems (5G) [1]. 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 Evolved Packet Core (EPC) [2]–[4]. RAN components will be divided into Base Band Unit (BBU) and Remote Radio Head (RRH), where BBU runs as software and RRH will be kept in the field. 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 a federated cloud; building the concept of Mobile Carrier Cloud [5]. The virtual instances of these components will then constitute the vEPC.

Whilst many solutions are currently available for the NFVO [6], such as Open Source Mano (OSM), VIM OpenStack, and VNF Manager JuJu, the optimal number of VM instances and their placement over the federated cloud are still overlooked [18]. In this paper, we aim to 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 for a specific traffic pattern. Let ϑ denote a VNF, whereby a set of different ϑs would compose the vEPC network service. Formally, ϑ can be one of the following network elements: HSS, MME, SGW or PGW. The proposed solution, dubbed virtual-EPC, derives the optimal number and location of VM instances of each VNF ϑ (i.e. HSS, MME, SGW and PGW). The virtual-EPC solution 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 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 coalitions 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 $\mathcal{CN}$.

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 Section IV. Section V evaluates the performance of the different optimization solutions envisioned in this paper. The paper concludes in Section VI.

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 network 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

978-1-5386-1734-2/18/$31.00 ©2018 IEEE

of a software, deployable in an on-demand and elastic manner on the cloud. In return of its numerous advantages, vEPC introduces some important challenges, mainly related to the placement of the telco-specific VNFs (i.e. MME, PGW, and SGW) over a federated cloud to ensure an optimal connectivity for users and simultaneously reduce the deployment cost.

More recently, new research work has emerged, proposing algorithms for the placement of vEPC's VNFs. In [14], 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 [15], the focus was on the VNF placement and instantiation of another mobile network functionality, namely the data anchoring gateway or PGW. The work argued the need for adopting the 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 [16], 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.

Unlike the research works in [7], [8], [11]–[16], which tackle either the optimal number of VNFs or the VNFs placement, our proposed solution jointly addresses both issues. Moreover, these research works 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

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, Network) [9], [10]. 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, storage) that would be dedicated to a specified VM in $\mathcal{CN}$. The Service Instance Manager (SIM) enables VNF Manager (VNFM) to monitor and manage a set of VNFs that belong to different $\mathcal{CN}$s. Meanwhile, NFVO is responsible for creating and orchestrating all the components including VNFs and VNFMs in different $\mathcal{CN}$s. The descriptions 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 work, we consider that the 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 a *TA* granularity [17], [19]. As per Release 8 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 [20]. 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. HSS, 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
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. Formally, $\mathcal{V} = \{\text{MME}, \text{HSS}, \text{SGW}, \text{PGW}\}$.
$\vartheta \in \mathcal{V}$	A VNF that would be deployed to build a vEPC. 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.
$\Phi^x_{\mathcal{A},\mathcal{B}}$	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.
$\Gamma^x_{\mathcal{A}}$	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.
$\mathcal{L}^{\mathcal{CN}}$	The set of flavors in $\mathcal{CN}$.

IV. VIRTUAL-EPC MECHANISM

In this section, we present the basic concept of the coalitional game and the different mechanisms used for instantiating vEPCs. In what follows, the VNF ϑ of vEPC will be instantiated one by one using coalitional game mechanism. Let $v(\mathcal{S})$ denote the characteristic function of the coalitional game, which is defined as follows:

$$v(\mathcal{S}) = \begin{cases} 0 & \text{If } |\mathcal{S}| = 0 \text{ or QoS is not ensured.} \\ \mathtt{P}_\vartheta - \theta_\vartheta & \text{Otherwise} \end{cases} \quad (1)$$

where $\mathtt{P}_\vartheta$ is the price that the operator is willing to pay for deploying VNF ϑ, whereas θ_ϑ is the price for deploying ϑ in variant $\mathcal{CN}$s.

A. Coalitional game for building one vEPC instance

In this paper, the type of coalitional games used belongs to the coalitional formation game which aims to form the different coalitions, such that the profit of the different players ($\mathcal{CN}$) is increased. The coalition formation would be defined according to the gain and the cost from the cooperation. 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 [22] is used in the literature to fairly share the profit among the different players. An easy way to share the profit among different players is the use of an equal-sharing method. Due to the simplicity of its implementation, this method is widely used in the literature [22]. 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 also used in our framework with a slight modification. Based on the above-mentioned remark, the payoff of each player $\mathcal{CN}$ in a coalition $\mathcal{S}$ is defined as follows:

$$\Pi_{\mathcal{S}}(\mathcal{CN}) = \frac{v(\mathcal{S})}{|\mathcal{S}|} \quad (2)$$

In what follows, we define the two main rules for the coalitional formation game [23]; the merge and split. 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 caring about the other players. The merge $\rhd_m$ and split $\rhd_s$ relations are defined as follows:

$$\begin{aligned} C_m \rhd_m C_s \Leftrightarrow \{ & (\forall \mathcal{S} \in C_s, \forall \mathcal{CN} \in \mathcal{S} : \\ & \Pi_{C_m}(\mathcal{CN}) \geq \Pi_{\mathcal{S}}(\mathcal{CN})) \\ & \wedge (\exists \mathcal{S}' \in C_s, \exists \mathcal{CN} \in \mathcal{S}' : \\ & \Pi_{C_m}(\mathcal{CN}) > \Pi_{\mathcal{S}'}(\mathcal{CN}))\} \end{aligned} \quad (3)$$

$$\begin{aligned} C_s \rhd_s C_m \Leftrightarrow \{ & \exists \mathcal{S} \in C_s : \\ & (\forall \mathcal{CN} \in \mathcal{S} : \Pi_{\mathcal{S}}(\mathcal{CN}) \geq \Pi_{C_m}(\mathcal{CN}) \wedge \\ & \exists \mathcal{CN} \in \mathcal{S} : \Pi_{\mathcal{S}}(\mathcal{CN}) > \Pi_{C_m}(\mathcal{CN}))\} \end{aligned} \quad (4)$$

We consider two coalitions $\mathcal{S}_1$ and $\mathcal{S}_2$, where $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$. Based on (3), $\mathcal{S}_1$ 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 other 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 (4) 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 instance

In this subsection, we will explain the *instanceVNF*$(\vartheta, \Gamma, \Phi)$ function that uses coalitional game to deploy the instances

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

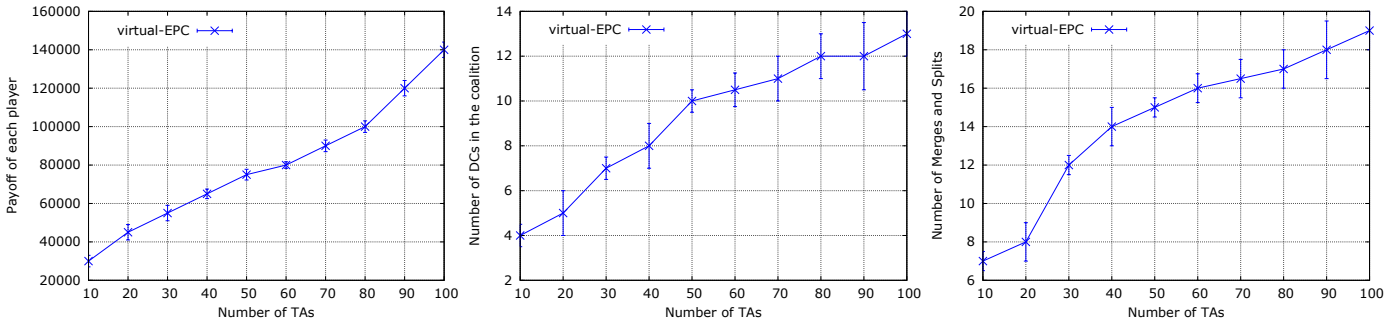
Input:
ϑ: A component of *vEPC*.
Γ: The number of cumulative events.
Φ: The number of cumulative events would be omitted.

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;

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 $\mathcal{CN}$. Algorithm 1 is used to explain the general functionality of *instanceVNF*$(\vartheta, \Gamma, \Phi)$.

The function *instanceVNF*$(\vartheta, \Gamma, \Phi)$ first starts by forming a collection Ξ by putting every player $\mathcal{CN}$ in a separate coalition (Algorithm 1: Line 1). Then, a variable $visited$ is initialized by $\emptyset$ (Algorithm 1: 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 1: Lines $3-44$). *instanceVNF*$(\vartheta, \Gamma, \Phi)$ computes the values of $v(\mathcal{S})$ (Algorithm 1: 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 1: Lines $10-20$) would be executed, and then only one split (Algorithm 1: 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 1: Line 10). These pairs of coalitions are put in the vector $visited$ to prevent redundancy checks (Algorithm 1: Line 11). To prevent the computation of the function $v(.)$ twice, the value of $v(\mathcal{S}_i \cup \mathcal{S}_j)$ will be put in the vector Ψ (Algorithm 1: 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 1: 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 1: 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 1: Line 23). The partitioning of the coalition $\mathcal{S}$ is done through the partitioning of an integer into two parts [21]. 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 1: 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 $v(.)$ is also prevented in the split process (Algorithm 1: Lines $24-29$). Then, the function *instanceVNF*$(\vartheta, \Gamma, \Phi)$ splits $\mathcal{S}$ if it is better for the collection Ξ (Algorithm 1: 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 1: Lines $3-44$) until achieving the ID$_p$-stable collection. Then, no further merge or split processes will be carried out. Later, for the VNF ϑ, the coalition $\mathcal{C}$ is selected from the collection Ξ that has the highest payoff value. Then, the VNF ϑ will be instantiated in $\mathcal{C}$.



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

Fig. 1. The performance evaluation of the proposed virtual-EPC scheme for a varying number of *TAs*.

V. Simulation Results

In this section, we evaluate the performance of the proposed scheme to instantiate vEPC instances over a federated cloud of $\mathcal{CN}$s. The proposed algorithm is 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.

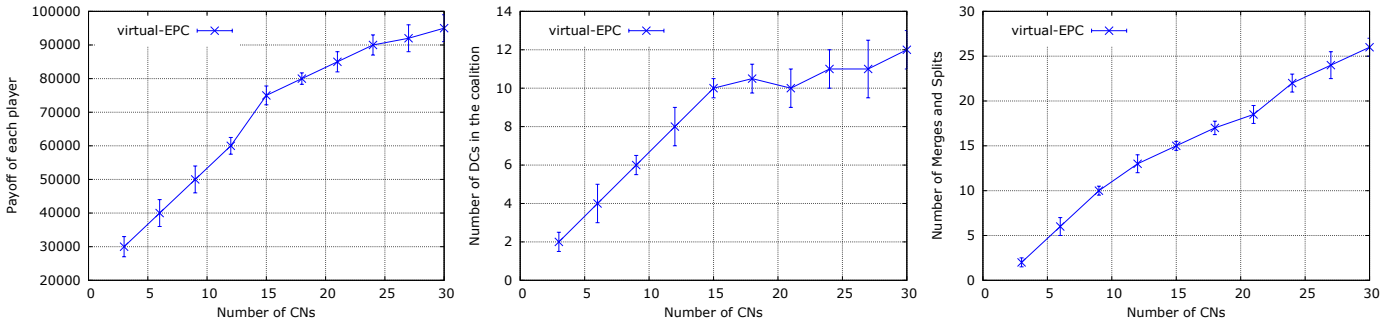
The algorithms are evaluated using the python programming language and an extended package for graph theory called networkx [24]. We implement the proposed virtual-EPC scheme using IBM ILOG CPLEX version 12.6.1, using the branch-and-bound method to solve the optimization problems. We used historical 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 operator is willing to pay – is set proportional to the number of *TAs* in the network.

Fig. 1 shows the performance of the proposed solution for a varying number of *TAs*. Fig. $1(a)$ depicts the impact of the number of TAs on individual payoffs of each $\mathcal{CN}$. We clearly observe that an increase in the number of TAs has a positive impact on the individual payoffs. For 100 TAs, the proposed virtual-EPC solution achieves an individual payoff of 140000. 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. $1(b)$, 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 instances. On the other hand, we observe from Fig. $1(c)$ 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.

Fig. 2 depicts the performance of virtual-EPC for varying numbers of players $\mathcal{CN}$s. In Fig. $2(a)$, 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. 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. Fig. $2(b)$ shows that the number of $\mathcal{CN}$s in the selected coalitions increases 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. $2(c)$ 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.

VI. Conclusion

The upcoming $5G$ mobile system will be based on the concept of carrier cloud to facilitate the upgrade for other



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

Fig. 2. The performance evaluation of virtual-EPC for varying numbers of $\mathcal{CN}$s.

next generation mobile systems. In this paper, we proposed an efficient VNF placement algorithm that aims to sustain the Quality of Service (QoS) while reducing the deployment cost. The developed framework aims 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, an algorithm was proposed that uses coalitional game to place the VNF instances across different $\mathcal{CN}$s, such that the QoS is ensured and the profit of each $\mathcal{CN}$ is maximized. The simulation results demonstrate the efficiency of the proposed framework. The obtained results clearly indicate the advantages of the proposed algorithm in ensuring QoS given a fixed cost for vEPC deployment, while maximizing the profits of cloud operators.

ACKNOWLEDGMENTS

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] I. Farris, T. Taleb, M. Bagaa, and H. Flinck, "Optimizing Service Replication for Mobile Delay-sensitive Applications in 5G Edge Network," in Proc. IEEE ICC 2017, Paris, France, May 2017.

[2] 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.

[3] I. Afolabi, A. Ksentini, M. Bagaa, T. Taleb, M. Corici, and A. Nakao, "Towards 5G Network Slicing over Multiple-Domains," in IEICE Trans. on Communications, Vol. E100.B, No. 11, Nov. 2017, pp. 1992-2006.

[4] I. Afolabi, M. Bagaa, T. Taleb, and H. Flinck, "End-to-End Network Slicing Enabled Through Network Function Virtualization," in Proc. IEEE CSCN17, Helsinki, Finland, Sep. 2017.

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

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

[7] 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.

[8] 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.

[9] I. Farris, J.B. Bernabe, N. Toumi, D. Garcia, T.Taleb, A.F. Skarmet, and B. Sahlin, "Towards Provisioning of SDN/NFV-based Security Enablers for Integrated Protection of IoT Systems," in Proc. IEEE CSCN17, Helsinki, Finland, Sep. 2017.

[10] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, "Towards Edge Slicing: VNF Placement Algorithms for a Dynamic & Realistic Edge Cloud Environment," in Proc. IEEE Globecom 2017, Singapore, Dec. 2017.

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

[12] 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.

[13] 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.

[14] T. Taleb and A. Ksentini, "Gateway Relocation Avoidance-Aware Network Function Placement in Carrier Cloud," in Proc. ACM MSWIM, Barcelona, Spain, Nov 2013, pp. 341-346.

[15] 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, pp. 2402-2407.

[16] 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, 2013, pp. 3602-3606.

[17] M. Bagaa, T. Taleb, and A. Ksentini, "Efficient Tracking Area Management in Carrier Cloud," in proc. IEEE Globecom'15, San Diego, CA, 2015, pp. 1-6.

[18] A. Nakao, P. Du, Y. Kiriha, F. Granelli, A. A. Gebremariam, T. Taleb, and M. Bagaa, "End-to-End Network Slicing for 5G Mobile Networks," in J. Information Processing, Vol. 25, No. 1, Jan. 2017, pp. 153-163.

[19] M. Bagaa, T. Taleb, and A. Ksentini, "Efficient Tracking Area Management Framework for 5G Networks", in IEEE Trans. on Wireless Communications, Vol. 15, No. 6, Jun. 2016, pp. 4117 4131.

[20] A. Kunz, T. Taleb, and S. Schmid, "On Minimizing SGW/MME Relocations in LTE," in Proc. ACM IWCMC'10, Caen, France, Jun 2010, pp. 960-965.

[21] D. Knuth, "The Art of Computer Programming", Vol. 4, Fascicle 3:"Generating All Combinations and Partitions", Addison-Wesley Professional, 2005.

[22] 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, Sep 2009, pp. 77-97.

[23] Thomas S. Ferguson, "Game Theory, Second Edition", 2014, Mathematics Department, UCLA, http://www.math.ucla.edu/~tom/Game_Theory/Contents.html.

[24] Networkx, "http://networkx.lanl.gov".