Service-Aware Network Function Placement for Efficient Traffic Handling in Carrier Cloud

Miloud Bagaa*, Tarik Taleb§ and Adlen Ksentini‡
* Department of Theories and Computer Engineering, CERIST, Algiers, Algeria. Email:bagaa@mail.cerist.dz
§ NEC Europe Heidelberg, Germany. Email: taleb@ieee.org
‡ IRISA, University of Rennes 1, Rennes, France Email: adlen.ksentini@irisa.fr

***Abstract*—Carrier Cloud is a promising concept towards the decentralization of mobile networks, to, in turn, alleviate mobile traffic load and reduce mobile operator cost. Carrier cloud is enabled by two main approaches, namely virtualization of the mobile network functions and networking over federated cloud. For intelligent carrier cloud dimensioning, the placement of mobile network functions over federated cloud is of vital importance. In this vein, this paper argues the need for adopting service/application type and requirements as metrics for (i) creating virtual instances of the Packet Data Network Gateways (PDN-GW) and (ii) selecting adequate virtual PDN-GWs for User Equipment receiving specific application type. After modeling this procedure as a nonlinear Optimization Problem and proving it as a NP-hard problem, we propose three solutions to solve it. The proposed solutions are evaluated through computer simulations and encouraging results are obtained.**

***Index Terms*—Carrier Cloud, Mobile Cloud, Mobile Networks, Network Function Virtualization, Load Balancing.**

I. Introduction

Building mobile networks, on demand and in an elastic manner, represents a vital solution for mobile operators to cope with the modest Average Revenues per User (*ARPU*) and the ever-increasing mobile data traffic. The key enabler of such elasticity is the virtualization of the mobile network functions, and the allowing of mobile networks through carrier cloud. The concept of carrier cloud is perceived as an important long-term solution for mobile operators to cope with the tremendous increase in their mobile data traffic and to get into the cloud computing area, seeking new business opportunities and defining new business models and strategies. As an important enabler of the carrier cloud concept, network function virtualization (*NFV*) is gaining great attention among industries. *NFV* aims for decoupling the software part from the hardware part of a carrier network node, traditionally referring to a dedicated hardware, single service and single-tenant box, and that is using virtual hardware abstraction [1]. Network functions become thus a mere code, runnable on a particular, preferably any, operating system and on top of a dedicated hardware platform. The ultimate objective is to run network functions as software in standard virtual machines (*VMs*) on top of a virtualization platform in a general purpose multi-service multi-tenant node (e.g., Carrier Grade Blade Server). A suitable Software Defined Networking (*SDN*) technology can be used to interwork between the different virtualized network functions on the different VMs within the same data center or across multiple data centers, to ultimately realize a flexible, dynamic, rapidly deployable, and elastic mobile network on the cloud. It is important to note that *SDN* is considered as complementary technology, network function can be virtualized and deployed without *SDN*.

To create an efficient carrier cloud that meets the general requirements of a mobile operator, the placement of network functions, namely the mobile Radio Access Network (RAN) functions, the mobile core network functions, and the caches or servers for the Packet Data Network, is of utmost importance and shall be based on different metrics such as application type, data center location, data center load, end-to-end Quality of Service (QoS)/Quality of Experience (QoE), that shall render the overall end-to-end communications optimal [2].

The objective of this paper is to propose a new algorithm to create virtual instance of the *PDN-GW* network functionality and its placement in the federated cloud. The number of *PDN-GW* and the adequate selection of the *PDN-GW* to *UEs* are not only based on the geographical location but also on the application type and the traffic load balancing. The number of PDN-GW for each traffic type would be reduced as much as possible, to reduce the cost of network operator, while ensuing user QoE. The process of creating *PDN-GW* and affecting these virtual instances to each UE is modeled in this paper through a nonlinear Optimization Problem (OP), where it is proved as an NP-hard problem. Accordingly, we proposed three heuristics to solve it, namely: Optimal Network Function Placement for Load Balancing Traffic Handling (ONPL), Greedy and Repeated Greedy algorithms.

The reminder of this paper is organized in the following way. Sections II is dedicated to related works on mobile network decentralization and the concept of carrier cloud. In section III we present the nonlinear optimization model of the *PDN-GW* instantiation and their assignment to UEs. This section includes also the details of the three proposed heuristics. Section IV presents the simulation results. Finally, we conclude this paper in Section V.

II. Related work

Due to the high increase in mobile data traffic, there is a general trend towards the decentralization of mobile operator networks. Decentralization will enable network operator to implement cost-effective methods to accommodate the increasing mobile traffic. Usually, there is a fundamental

978-1-4799-3083-8/14/$31.00 ©2014IEEE

technology and cost related trade-off behind the adoption of either centralized or decentralized network architecture. Costly network equipment are usually shared, creating centralized architectures. This was the case in the initial phase of 3G deployment, wherein a centralized architecture was chosen to share core network utilities and processing resources, while keeping the base station cost low [3]. However, the evolution of computer technology has significantly reduced equipment costs, advancing their deployment flexibility. Therefore, it is possible to implement cost-effective network decentralization by placing small-scale network nodes with mobility and IP access functionalities towards the network edge. Such decentralization architecture opens up a new way to further reduce data traffic either by using techniques like data offloads while accommodating user QoE, or by selecting the anchor gateway not only based on geographical location but also by using application type and requirement. Beside the standardized Selected *IP* Traffic Offload (*SIPTO*) and Local *IP* Access (*LIPA*) concepts, which were introduced by the *SA2 3GPP* group, there are different solutions that addressed the data offload issue. In [4] the authors proposed a traffic upload technique for cellular architecture, which includes Home eNodeB (HeNodeB) heterogeneous access cells. Based on *SIPTO* architecture, they proposed to implement a Traffic Offload Function (*TOF*) on both *H(e)Nodes* (Home eNodeB) gateway and *H(e)NodeBs*. In [5] the authors presented the Bearer Based Offload Mechanism In the Core Network (*B2OMICN*) mechanism. The *TOF* used in *B2OMICN* is based on the bearer information, such as *APN* (Access Point Name), destination *IP*, address range and port number, and implemented at the *PDN-GW* level. In [6] the authors presented Dynamic *SIPTO* mechanism, which combines the Fast Fourrier Transform (*FFT*) based *IP* traffic classification with the dynamic traffic offload path selection algorithm. The *FFT* technique is used to classify the traffic and according to the *QoS* of each traffic a dynamic offload path is selected. Another way to identify the traffic is by using a DNS server as presented in [7]. According to the requested application, the *DNS* server informs a *UE* of the gateway (offloading point) to connect to for establishing a connection. The gateway selection is solely based on the geographical proximity of the *UE* to the gateway and the gateway load. Unlike the precedent solutions which are based on *SIPTO* architecture, the authors in [8] presented a generic data offloading solution, which requires that a *UE* can support multi-PDN connections. The proposed solution aims at enabling a *UE* to know how and when to establish a new optimized *PDN* connection for launching new *IP* sessions to a particular *APN*, while avoiding compromising the ongoing *PDN* connection for the same APN.

In most of the above mentioned offload solutions, the gateway selection is only based on geographical proximity and/or load, which is not sufficient [9]. Accordingly, the authors in [9] proposed a new PDN gateway selection, which in addition to load balancing and geographical/topological proximity considers end-to-end connection optimization and/or the application type. Indeed, certain type of application requires specific functionalities, such as content caches or machine type communication service, which are available only at certain *PDN-GW*.

Meanwhile, network decentralization can benefit with the virtualization of mobile network functions and the enabling of carrier cloud networking, where a mobile networks are created on demand and in a flexible manner. However, an efficient decentralized mobile network cannot be built without efficient algorithms for the placement of network function over the federated cloud. In [10] the authors propose a network function placement, particularly for creating mobile gateway functionalities (Serving Gateway - (S-GW)) and their placement in the federated cloud. To create these virtual instances and hence building the Serving Area planning, the authors considered gateway relocation as metric. In other words, the aim is to find a trade-off between minimizing the UE handoff between *SA*, and minimizing the number of created instance of the virtual *S-GW*. In the present paper, we address the network placement and instantiation of another mobile network functionality, which is data anchoring or *P-GW* creation/selection.

III. Proposed solution

A. Notations

The notations used throughout the paper are summarized in Table I.

Notation	Description
M	Number of traffics types in the network.
UE	User equipment.
PDN-GW	Packet data network gateway.
UE_i	UE whose identifier is i.
$PGW_k(i)$	*PDN-GW* of UE_i for traffic k
$PA_k(i)$	Set of UEs that select the same *PDN-GW* for traffic k. Also, $PA_k(i)$ is called *PDN* Area i for traffic k.
SPA_k	Set of $PA_k(i)$ that use the same traffic k. Formally, $SPA_k = \{PA_k(1), PA_k(2), \cdots\}$, where $\forall i \neq j, PA_k(i) \cap PA_k(j) = \emptyset$
$nPGW_k$	Number of PDN-GW created in the network to manage traffic k. In other word, $nPGW_k$ is the number of $PA_k(i)$ in SPA_k. Formally, $nPGW_k = \mid SPA_k \mid$.
$\lambda_k(i)$	The amount of traffic k generated by UE_i during a period of time.
$\mu_k(i)$	The amount of traffic k generated by a $PA_k(i)$ during a period of time. Formally, $\mu_k(i) = \sum_{UE_j \in PA_k(i)} \lambda_k(j)$.
N_k	Number of UEs that use traffic k. Formally, $N_k = \sum_{PA_k(i) \in SPA_k} \mid PA_k(i) \mid$.
$PGWMAX_k$	the maximum capacity of PGW_k to handle the traffic of type k generated by the UEs.

TABLE I: Notations

B. Problem formulation

Let us we assume that there are M *APN*, where each *APN* gives access to one or several services/applications. Therefore, M *PDN-GW* types would be considered to satisfy *UEs* requests, wherein each *PDN-GW* is associated to an *APN*. Based on the analysis of the traffic generated by each *UE* in the past, we assume that we can predict: (i) the number of *UEs* which are interested by each traffic k and hence the

APN name; (ii) the amount of traffic generated by an *UE* for each type of traffic. Let us we denote by N_k the number of *UEs* which are interested by traffic k and $\lambda_k(i)$ the amount of traffics of type k generated by UE_i, during a period of time. Further, we assume that each *PDN-GW* has a maximum capacity ($PGWMAX_k$), in terms of supported amount of traffic to handle. Each *UE* can generate one or several type of traffic (noted k). Accordingly, the proposed solution aims to instantiate the number of *PDN-GW* that supports traffic k (noted PGW_k), and select the adequate virtual *PDN-GW* to be used by each *UEs*. The objective of the proposed solution is to reduce the cost of network operator and increase the network performance. The cost reduction is achieved by: (i) decreasing the number of virtual *PDN-GW* per type of traffic to create $nPGW_k$, while ensuring high user QoE ; (ii) ensuring load balancing between PGW_k, such as each PGW_k would manage the same amount of traffics.

To represent the affectation of each *PDN-GWs* supporting traffic k (PGW_k) to *UEs*, we use a binary symmetric matrix ($N_k \times N_k$) called $adj(PGW_k)$. An element in $adj(PGW_k)$ denotes the case whether or not two *UEs* are using the same *PDN-GW*$_k(u)$ for a specific traffic type k.

$$\text{adj}_{i,j}(PGW_k) = \begin{cases} 1 & if\ PGW_k(i) = PGW_k(j) \vee i = j \\ 0 & Otherwise \end{cases} \tag{1}$$

Let $PGWMAX_k$ is the maximum capacity of PGW_k to handle the traffic generated by the *UEs*. Let ψ_k a vector represented as follows: $\psi_k = (\lambda_k(1), \cdots, \lambda_k(N_k))$. We denote by Z_k the capacity vector of PGW_k. Formally, Z_k is defined as follows:

$Z_k = (z_k(1), \cdots, z_k(N_k)) = \psi_k \times adj(PGW_k)$

i.e.,

$$\text{for } i = 1 \cdots N_k,\ z_k(i) = \sum_{j=1}^{N_k} \lambda_k(j) \times adj_{i,j}(PGW_k)$$

Each $z_k(i)$ denotes the traffic load of a PA_k. The redundancy in $adj(PGW_k)$ creates also a redundancy in $z_k(i)$, and hence the number of PGW_k is lower or equals to the number of elements in Z_k. Although it may exist two *UEs* i and j with $i \neq j$, it is possible that $z_k(i)$ and $z_k(j)$ represent the traffic load of the same PGW_k. This means that *UEs* i and j are using the same PGW_k. Since the proposed solution objective is to ensure the constraint: $\forall i \in \{1, \cdots N_k\}, z_k(i) \leq PGWMAX_k$, therefore the proposed solution is formulated into an optimization problem (OP) as follows:

$$\begin{cases} \textbf{Objectives:} & max \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} adj_{i,j}(PGW_k) \\ & min \sum_{i=1}^{N_k} \sum_{j=i+1}^{N_k} \mid z_k(i) - z_k(j) \mid \\ \textbf{Constraints:} & \\ for\ i = 1 \cdots N_k,\ z_k(i) = \sum_{j=1}^{N_k} \lambda_k(j) \times adj_{i,j}(PGW_k) & \\ for\ i = 1 \cdots N_k, z_k(i) \leq PGWMAX_k & \\ for\ i,j = 1 \cdots N_k, adj_{i,j}(PGW_k) \in \{0,1\} & \\ for\ i,j = 1 \cdots N_k, adj_{i,j}(PGW_k) = adj_{j,i}(PGW_k) & \\ for\ i,j,l = 1 \cdots N_k, adj_{i,j}(PGW_k) \times adj_{i,l}(PGW_k) = & \\ \quad i \neq j \neq l \qquad adj_{j,i}(PGW_k) \times adj_{j,l}(PGW_k) & \end{cases} \tag{2}$$

The first objective aims to increase as much as possible the number of *UEs* using the same PGW_k. This reflects the fact that reducing the number of virtual instance of *PDN-GW* (i.e., number of PA_k), and hence the operator cost. The second objective addresses the load balancing between PGW_k, by reducing the euclidean distances between the traffics handled by the PGW_ks. It is important to note that since the euclidean distances between the same PGW_k is zero, the redundancy in Z_k does not have an impact on the second objective. Meanwhile, the constraints in OP are used to ensure the following statements:

1) Constraints 1 and 2: Ensure that each PGW_k is affected to *UEs* while its maximum capacity is not exceeded.
2) Constraint 3: Ensures that $adj_{i,j}(PGW_k)$ is a binary matrix.
3) Constraint 4: Ensures that $adj_{i,j}(PGW_k)$ is a symmetric matrix.
4) Constraint 5: Ensures the transitivity in $adj_{i,j}(PGW_k)$. If there is three, u, v and w, such that $adj_{u,v}(PGW_k) = 1$ and $adj_{u,w}(PGW_k) = 1$, then $adj_{v,w}(PGW_k)$ and $adj_{w,v}(PGW_k)$ should be 1 (i.e., $\{u, v, w\} \subseteq PA_k(i)$). In constraint 5, if $adj_{u,v}(PGW_k) = 1$ and $adj_{v,u}(PGW_k) = 1$, then u and v should have the same relation with other UEs in $adj(PGW_k)$ (i.e., line $adj_u(PGW_k)$ equals to line $adj_v(PGW_k)$ in $adj(PGW_k)$).

The OP formulated in inequality 2 becomes a multi-objectives nonlinear OP. The aim is to maximize the number of UEs that use the same PGW_k and minimize the traffics difference between PGW_ks (load balancing). Usually, it is more practical to resolve an OP that aims to minimize (resp., maximize) all the objectives. For this reason, in what follows we will reformulate the OP presented in inequality 2. The maximization of number of UEs that use the same PGW_k is equivalent to reduce the number of UEs that do not use the same PGW_k. In other word, we aim to minimize the number of zero in $adj(PGW_k)$. Moreover, to remove absolute value in the objective, Least absolute deviations is used, which adds two constraints to the OP. This latter can be simplified as follows:

$$\begin{cases} \textbf{Objectives:} & min \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} 1 - adj_{i,j}(PGW_k) \\ & min \sum_{i=1}^{N_k} \sum_{j=i+1}^{N_k} T_k(i,j) \\ \textbf{Constraints:} & \\ for\ i = 1 \cdots N_k,\ z_k(i) = \sum_{j=1}^{N_k} \lambda_k(j) \times adj_{i,j}(PGW_k) & \\ for\ i = 1 \cdots N_k, z_k(i) \leq PGWMAX_k & \\ for\ i,j = 1 \cdots N_k, adj_{i,j}(PGW_k) \in \{0,1\} & \\ for\ i,j = 1 \cdots N_k, adj_{i,j}(PGW_k) = adj_{j,i}(PGW_k) & \\ for\ i,j,l = 1 \cdots N_k, adj_{i,j}(PGW_k) \times adj_{i,l}(PGW_k) = & \\ \quad i \neq j \neq l \qquad adj_{j,i}(PGW_k) \times adj_{j,l}(PGW_k) & \\ for\ i = 1 \cdots N_k\ and\ j > i, T_k(i,j) \geq z_k(i) - z_k(j) & \\ for\ i = 1 \cdots N_k\ and\ j > i, T_k(i,j) \geq -(z_k(i) - z_k(j)) & \end{cases} \tag{3}$$

The OP presented in inequality 3, can be simplified by reducing the number of objectives to one by using weighted-sum technique. To do so two parameters, $\alpha \in]0,1]$ and

$\beta \in]0,1]$, are added. If these parameters are fixed, then a priority between the two objectives is introduced. Otherwise, an optimal solution will be achieved for any α and β values. The OP presented in inequality 3, can be reformulated as follows:

$$\begin{cases} \textbf{Objective: } min\ \alpha \times \sum_{i=1}^{N_k}\sum_{j=1}^{N_k} 1 - adj_{i,j}(PGW_k) \\ \qquad\qquad + \beta \times \sum_{i=1}^{N_k}\sum_{j=i+1}^{N_k} T_k(i,j) \\ \textbf{Constraints:} \\ for\ i = 1 \cdots N_k,\ z_k(i) = \sum_{j=1}^{N_k} \lambda_k(j) \times adj_{i,j}(PGW_k) \\ for\ i = 1 \cdots N_k, z_k(i) \leq PGWMAX_k \\ for\ i,j = 1 \cdots N_k, adj_{i,j}(PGW_k) \in \{0,1\} \\ for\ i,j = 1 \cdots N_k, adj_{i,j}(PGW_k) = adj_{j,i}(PGW_k) \\ for\ i,j,l = 1 \cdots N_k, adj_{i,j}(PGW_k) \times adj_{i,l}(PGW_k) = \\ \qquad i \neq j \neq l \qquad adj_{j,i}(PGW_k) \times adj_{j,l}(PGW_k) \\ for\ i = 1 \cdots N_k\ and\ j > i, T_k(i,j) \geq z_k(i) - z_k(j) \\ for\ i = 1 \cdots N_k\ and\ j > i, T_k(i,j) \geq -(z_k(i) - z_k(j)) \\ \alpha > 0 \\ \alpha \leq 1 \\ \beta > 0 \\ \beta \leq 1 \end{cases} \tag{4}$$

Algorithm 1 ONPL solution: PDN-GW planing algorithm

Input:
$L = \{UE_k(1) \cdots UE_k(N_k)\}$: set of UEs that use traffic k.
$\psi_k = \{\lambda_k(1) \cdots \lambda_k(N_k)\}$: the amount of UEs' traffic k.
Output:
SPA_k: planning of PA_k.
$nPGW_k$: number of PDN-GW k.

1: $nPGW_k = N_k$;
2: **for all** $i \in \{1 \cdots N_k\}$ **do**
3: $PA_k(i) = \{UE_k(i)\}$;
4: $\mu_k(i) = \lambda_k(i)$;
5: **end for**
6: $SPA_k = \{PA_k(1) \cdots PA_k(N_k)\}$;
7: **while** $\boldsymbol{True}$ **do**
8: $PA_k(i) = \boldsymbol{Min}(SPA_k)$;
9: $PA_k(j) = \boldsymbol{Min}(SPA_k - \{PA_k(i)\})$;
10: $UE_u = \boldsymbol{MinUE}(PA_k(i))$;
11: **if** $\mu_k(j) + \lambda_k(u) > PGWMAX_k$ **then**
12: $\boldsymbol{Break}$;
13: **end if**
14: $PA_k(j) = PA_k(j) \cup \{UE_u\}$;
15: $PA_k(i) = PA_k(i) - \{UE_u\}$;
16: $\mu_k(j) = \mu_k(j) + \lambda_k(u)$;
17: $\mu_k(i) = \mu_k(i) - \lambda_k(u)$;
18: **if** $PA_k(i) == \emptyset$ **then**
19: $SPA_k = SPA_k - PA_k(i)$;
20: $nPGW_k = nPGW_k - 1$;
21: **end if**
22: **end while**

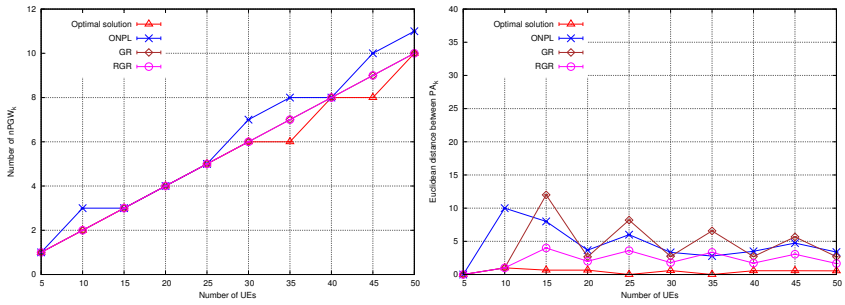
C. Algorithms for PGW_k planing

For each kind of traffic k, the number of variables in OP 4 equals to $(N_k)^2 + \frac{N_k \times (N_k - 1)}{2} + N_k$ and the number of constraints equals to $(N_k)^2 \times (N_k - 1)$. In order to resolve OP 4, we have reduced the number of variables to N_k^2 by exploiting symmetric feature of $adj(PGW_k)$. Accordingly, the complexity of existing algorithms that can resolve this optimization problem is exponential. Usually the number of *UEs* (i.e., *NbUE*) in the mobile network is very high, it can exceed one million. By consequence, resolving the OP 4 using an exact solution is NP-Hard.

In order to resolve the OP 4, we propose three heuristics: (i) Optimal Network Function Placement for Load Balancing Traffic Handling (i.e., *ONPL*); (ii) Greedy (i.e., *GR*); (iii) Repeated Greedy (i.e., *RGR*).

1) ONPL: The different steps of *ONPL* are depicted in Algorithm 1. The *ONPL* will be executed for each traffic k to assign a set of *PDN-GW* (i.e., PGW_k) to UEs. For each traffic k, *ONPL* generates $SPA_k = \{PA_k(1), PA_k(2), \cdots PA_k(nPGW_k)\}$, where $PA_k(i)$ is a set of UEs managed by the same *PDN-GW*. To construct SPA_k, *ONPL* first starts by an inefficient solution, in which each PGW_k is selected only by one UE i.e., $\forall i, \mid PA_k(i) \mid = 1$ (Algorithm 1: Lines $1-6$). Afterward, *ONPL* is executed in iterations, in each one *ONLP* is improved and then converges to the optimal solution. In each iteration the number of PDN-GW (i.e., $nPGW_k$) is decreased while ensuring the load balancing. Function Min(), in Algorithm 1 line 8, allows to select $PA_k(i)$ that has the smallest $\mu_k(i)$, whereas function *MinUE()* in line 10 allows to select UE_u that has the smallest $\lambda_k(u)$ from $PA_k(i)$. To reduce $nPGW_k$, *ONPL* is executed in iterations, in each one *ONPL* aims to remove $PA_k(i)$ that has the smallest amount of traffic by distributing its *UEs* over other PA_k. To ensure the load balancing between PA_k in $SPA_k - PA_k(i)$, in each iteration, *ONPL* moves UE_u from $PA_k(i)$ that has the smallest amount of traffic to another PA_k that has the smallest amount of traffic in $SPA_k - PA_k(i)$. To do so, *ONPL*, from SPA_k, selects $PA_k(i)$ and $PA_k(j)$ that have the smallest amount of traffic $\mu_k(i)$ and $\mu_k(j)$ (Algorithm 1: Lines $8-9$), respectively. And then, UE_u is selected from $PA_k(i)$ as the UE which has the smallest amount of traffic $\lambda_k(u)$ (Algorithm 1: Line 10). A test is done on $\lambda_k(u)$ and $\mu_k(j)$. If $\mu_k(j) + \lambda_k(i) \leq PGWMAX_k$, UE_u will be moved from $PA_k(i)$ to $PA_k(j)$ (Algorithm 1: Line $14-15$). Also, the amount of traffic $\mu_k(j)$ (resp., $\mu_k(i)$) will be increased (resp., decreased) by $\lambda_k(u)$ (Algorithm 1: Lines: $16-17$). If $PA_k(i)$ is empty, it will be removed from SPA_k, as well as $nPGW_k$ will be decreased. Otherwise, $\mu_k(i) + \mu_k(j) > PGWMAX_k$, *ONPL* cannot reduce more the number of PA_k i.e., the best solution is achieved for traffic k.

2) Greedy: Algorithm 2 illustrates the different steps of the greedy solution. Similarly to *ONPL*, the greedy solution will be executed for each kind of traffic k to generate $SPA_k = \{PA_k(1), PA_k(2), \cdots PA_k(nPGW_k\}$ and $nPGW_k$. Initially, there is no *PDN-GW* in the network, and then the following tasks will be done: (i) $nPGW_k$ is initialized to zero (Algorithm 2: Line 1) and its traffic amount (i.e., $\mu_k(1)$) with zero (Algorithm 2: Line 3); (ii) the first PA_k is created and initialized with empty set (Algorithm 2: Line 4); (iii) SPA_k is created and initialized by the first PA_k (Algorithm 2: Line 5). Afterwards, the UE in L will be pushed on PA_k while ensuring that the maximum capacity of PGW_k (i.e., $PGWMAX_k$) is not exceeded (Algorithm 2: Lines $14-15$). Otherwise, a new PA_k is created and pushed on SPA_k, as



(a) (b)

Fig. 1: Comparison between proposed algorithms and the optimal solution

well as $nPGW_k$ is increased (Algorithm 2: Lines $7-13$). When all UEs are processed, the best solution is achieved for traffic k.

Algorithm 2 Greedy solution: PDN-GW planing algorithm

Input:
$L = \{UE_k(1) \cdots UE_k(N_k)\}$: set of UEs that use traffic k.
$\psi_k = \{\lambda_k(1) \cdots \lambda_k(N_k)\}$: the amount of UEs' traffic k.

Output:
SPA_k: planning of PA_k.
$nPGW_k$: number of PDN-GW k.

1: $nPGW_k = 0$;
2: $i = 1$;
3: $\mu_k(i) = 0$;
4: $PA_k(i) = \{\}$;
5: $SPA_k = \{PA_k(i)\}$;
6: **for all** $UE \in L$ **do**
7: **if** $\mu_k(i) + \lambda_k(UE) > PGWMAX_k$ **then**
8: $nPGW_k = nPGW_k + 1$;
9: $i = i + 1$;
10: $\mu_k(i) = 0$;
11: $PA_k(i) = \{\}$;
12: $SPA_k = SPA_k \cup PA_k(i)$;
13: **end if**
14: $\mu_k(i) = \mu_k(i) + \lambda_k(UE)$;
15: $PA_k(i) = PA_k(i) \cup UE$;
16: **end for**

3) Repeated Greedy: Repeated greedy solution (i.e., *RGR*) is an enhancement of the greedy solution. To generate $SPA_k = \{PA_k(1), PA_k(2), \cdots PA_k(nPGW_k)\}$ and $nPGW_k$, repeated greedy executes greedy algorithm N_k times through circular permutation. In each one i, greedy algorithm is executed with a new arranging of UEs (i.e., L_i). Initially, L_1 equals $\{UE_k(1), \cdots UE_k(N_k)\}$, then the second UE in L_2, it is shifted left i.e., $L_2 = \{UE_k(2), \cdots UE_k(N_k), UE_k(N_1)\}$. Last but not least, $L_{N_k} = \{UE_k(N_k), UE_k(1) \cdots, UE_k(N_{N_k-1})\}$ is the last set of UEs which is considered. The best solution is selected from the greedy algorithm results as the one which generates the lowest number of *PDN-GW* (it has the lowest $nPGW_k$). If a tie, more than one greedy solution which have the lowest $nPGW_k$, the best solution is selected as the one which creates less euclidean distance between PA_ks.

IV. Performance Evolution

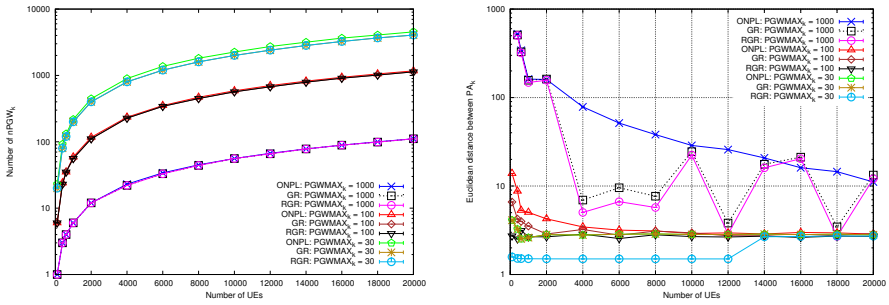
The proposed heuristics (i.e., *ONLP*, *GR* and *RGR*) are evaluated through computer simulation. To show the efficiency of the proposed schemes, we compare their performances with the optimal (or exact) solution. The performances of the optimal solution are obtained by resolving OP 4 through Mathematica [11] software. Due to the complexity of OP 4, during our experiments, the number of UEs does not exceed 50 in the optimal solution. The proposed schemes and the optimal solutions are evaluated in terms of the following metrics:

- Required number of *PDN-GW*: This metric reports the number of PA_k (i.e., $nPGW_k$) that should be created to satisfy all *UEs* in the network.
- Euclidean distances between PA_ks: This metric reports the load balancing between the *PDN-GW* in the network to ensure QoS and QoE for users. It is defined as the average euclidean distance between μ_ks. If the euclidean distance between PA_ks is higher, then there is some PA_ks in the network which are overloaded than the others.

The algorithms evaluation is performed by varying the number of *UEs* in the network and traffic types. During the experiments, the amount of traffic of each *UE* is selected randomly from 1 and 10 Mega Bytes (MB). We conduct two sets of experiments: firstly, the proposed solutions are compared to the optimal one by varying the number of *UEs* until 50, while considering only one type of traffic. $PGWMAX_k$ of considered traffic is fixed to 30 MB. Secondly, only a comparison between the proposed solutions is done by considering three kinds of traffic and varying the number of *UEs* until 20000. $PGWMAX_k$ values of considered traffic are fixed to 30, 100 and 1000 MB, respectively.

A. Required number of PDN-GW

Fig. $1(a)$ shows a comparison of the proposed schemes with the optimal solution in terms of required number of *PDN-GW*. The first observation we can draw from Fig. $1(a)$ is that the proposed solutions have performances near to the optimal one. Greedy and repeated greedy solutions slightly outperform ONPL one. In all the solutions, we note that the number of required PGW_k is proportional to the number of *UEs*. This is intuitive as the PGW_k has a maximum capacity, and increasing the number of *UEs* involves the need for more PGW_k instances to handle the generated traffic. The same trend is seen in Fig. $2(a)$, which compares the performance of only the proposed solutions when increasing



(a) (b)

Fig. 2: Comparison between proposed solutions

the number of *UEs* more than 50 and for different values $PGWMAX_k$. We recall that the exact solution is obtained only when the number of *UEs* is less than 50. We observe that augmenting the number of *UEs* leads to increase the number of virtual instance of *PDN-GW*. But, the number of required PGW_k is inversely proportional to $PGWMAX_k$ value, which is logical as augmenting *PDN-GW* capacity has a positive impact on the number of *UEs* supported by each *PDN-GW*. Moreover, we see in Fig. $2(a)$ that the Greedy and repeated greedy solutions slightly enhance *ONPL* when $PGWMAX_k = 30$ MB. However, all the proposed heuristics have the same performances when $PGWMAX_k = 100$ MB or $PGWMAX_k = 1000$ MB.

B. Euclidean distances between PA_ks

Fig. $1(b)$ and Fig. $2(b)$ illustrate the load balancing between the *PDN-GW* as a function of *UEs*. Note that in Fig. $2(b)$, we used a logarithmic scale to represent the y axe (Eucledian distance). We clearly remark from these two figures that the capacity of the *PDN-GW* has a direct impact on the Eucledian distance and hence on the load balancing metric. In fact, increasing of $PGWMAX_k$ (i.e., the capacity of PGW_k) leads to reduce the number of *PDN-GW* required by the network, which has a negative impact on the distribution of the traffic load among the *PDN-GW* instances. This is well confirmed in Fig. $2(b)$ for the case of Greedy algorithm. In this algorithm, a worst performance are achieved when $PGWMAX_k$ has a higher value. Moreover we notice that the performances of the repeated greedy solution is near to the optimal one when *UEs* number is less than 50, and outperforms the other solutions for high number of *UEs*. We argue this by the fact that the repeated greedy solution is able to find near optimal solution as it is repeated N times, which allows it to explore different solutions and select the best one, which is not the case of the two others solutions.

V. Conclusion

In this paper we have presented one of the important component of the carrier cloud vision, as to know Network Virtualization Function. We focused on the data anchor (*PDN-GW*) virtualization, and more specifically on how instantiating and assigning the virtual *PDN-GW* to *UEs*. Rather than using only geographical location, we proposed to consider applications/services type when selecting a *PDN-GW* to *UEs*. We modeled this process through a nonlinear optimization problem and showed that the optimal solution is NP-hard. Accordingly, we proposed three heuristics to deal with this issue. Simulation results have proved that the proposed schemes have a performance near to the optimal one, in terms of two objectives: (i) the created number of virtual *PDN-GW*; (ii) ensuring fair load balancing among these *PDN-GW*. The first one is for reducing the operator costs, and the second objective is for ensuring high QoS/QoE for users.

Acknowledgment

The research work presented in this paper is conducted as part of the Mobile Cloud Networking project, funded from the European Union Seventh Framework Program under grant agreement number [318109].

References

[1] Authored by network operators, "Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges, & Call for Action," Oct. 2012

[2] T. Taleb and A. Ksentini, "Follow Me Cloud: Interworking Federated Clouds & Distributed Mobile Networks," to appear in IEEE Network Magazine.

[3] P. Bosch, L. Samuel, S. Mullender, P. Polakos, and G. Rittenhouse, "Flat Cellular (UMTS) Networks", in Proc. IEEE WCNC Hong Kong, Mar 2007.

[4] J. Roh, Y. Ji, YG. Lee, T. Ahn, "Femtocell traffic offload scheme for core networks", in Proc. IFIP International Conference on new Technologies, mobility and security, Paris, France, 2011.

[5] L. Ma, W. Li, "Traffic offload mechanism in EPC based on bearer type", Internationale Conference on Wireless Communications, networking and Mobile Computing (WiCOM), Wuhan, China, 2011.

[6] H. Han, Y. Han, Y. Zhou, L. Huang, M. Qian, J. Hu, J. Shi, "FFT traffic classification-based dynamic selected IP traffic offload mechanism for LTE HeNB networks", Springer Mobile Network Application, doi 10.007/s11036-012-0426-7.

[7] T. Taleb, K. Samdanis, S. Schmid, "DNS-based solution for operator control of selected IP traffic offload", in Proc. IEEE International Conference on Communications (ICC), Kyoto, Japan, 2011.

[8] T. Taleb, Y. Hadjadj-Aoul, and S. Schmid, "Geographical Location and Load based Gateway Selection for Optimal Traffic Offload in Mobile Networks," in Proc. IFIP Networking, Valencia, Spain, May 2011.

[9] T. Taleb and A. Ksentini, "On Efficient Data Anchor Point Selection in Distributed Mobile Networks," in Proc. IEEE ICC 2013, Budapest, Hungary, Jun. 2013.

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

[11] Wolfram Mathematica Ver 9. [Online]. Available: http://reference.wolfram.com/mathematica/guide/Mathematica.html.