JITEL 2017
XIII Jornadas de Ingeniería Telemática
27-29 Septiembre 2017, Valencia (España)

Actas de las XIII Jornadas de Ingeniería Telemática (JITEL 2017), Valencia (España), 27-29 de Septiembre de 2017.

ISBN: 978-84-9048-595-8
DOI: http://dx.doi.org/10.4995/JITEL2017.2017.6490

PaCoVNE: Power Consumption Aware Coordinated VNE with Delay Constraints

Khaled Hejja, Xavier Hesselbach
Dept. Ingeniería Telemática
Universitat Politècnica de Catalunya
C/ Jordi Girona, 1-3 - Edif.C3 - Campus Nord - 08034 Barcelona - Spain
{khaled.hejja, xavier.hesselbach}@upc.edu

***Abstract*—This paper introduces a more efficient embedding approach, called power consumption aware and coordinated VNE heuristic, denoted as (PaCoVNE). It embeds both virtual nodes and edges, simultaneously, and within one stage, while satisfying CPU and BW constraints, minimizes power consumption of the whole substrate network, and considers end-to-end delay as a major constraint. Performance of the new heuristic was compared to the energy aware algorithm OCA/EA-RH, for off-line scenario using homogeneous configurations, with and without end-to-end delay. The paper also presents simulation results once without end-to-end delay, and also when it was included.**

Keywords: Virtual Network Embedding, Power Consumption, Coordinated, Delay

I. INTRODUCTION

Networks virtualization has become an integral component of future Internet, offering network operators a way to overcome ossification of the Internet, by consolidating many of their equipments onto standardized high volume components located at centralized data centers [2]-[4]. More specifically, the key advantageous of network virtualization are basically related to efficiently utilizing physical network resources through sharing them among several virtual networks (VN), as well as providing more flexibility to manage, expand, or shrink the physical network according to VNs' characteristics.

However, allocating enough resources to satisfy all requirements of a virtual network request (VNR), on top of a substrate network (SN) that has limited residual capacities, is a very challenging task in network virtualization [5]. To realize that, VNE process is usually divided into two sub-problems, the first one is allocating virtual nodes onto physical nodes, which is known as virtual node mapping (VNM) stage. The other one, is virtual edge mapping (VEM), which embeds virtual edges onto physical paths connecting corresponding nodes in the physical network. Along such process, VNE usually trades off between minimizing embedding costs through utilizing less SN resources, and maximizing revenues through accepting as much as possible VNRs, while maintaining acceptable quality of services (QoS).

Generally, VNM and VEM stages can be carried out in two strategies, uncoordinated or coordinated [5]. Regarding the uncoordinated case, VNM and VEM used to be solved independently without any coordination between the two stages, raising the possibilities of higher VNRs rejections. This is because VEMs could be mapped on longer physical paths, therefore, utilizing additional resources, consuming more power, and adding more delay due to passing through hidden hopes [6]. The other strategy, is performing both VNM and VEM in two separate, but coordinated stages, where VNM is performed according to predefined VEM constraints to guide allocating the virtual nodes [7]. However, even through there is a sort of coordinating VNM with VEM, still, virtual nodes could be embedded at physical nodes that could be farther away from each other, enforcing edges to be mapped at longer physical paths, resulting on similar disadvantageous as in the uncoordinated scenario. Furthermore, regardless of the used strategy, VNE used to be constrained by CPU and BW resources, but occasionally considering power consumption, and almost very seldom adding delay as an additional constraint. Thus, it could be possible that, the lack of considering more constraints throughout the VNE process, would result on a degraded QoS for the whole embedding process, including raising operational costs, consuming more power, as well as generating less revenues.

In view of that, this paper introduces the PaCoVNE approach, as a fully coordinated VNE algorithm. It performs virtual nodes and edges embeddings simultaneously and in one stage, according to the following constraints combined: CPU, throughout, power consumption and end-to-end delay. The core of PaCoVNE approach is based on formulating VNR's demands and SN paths' resources into two separate sets, called (Segments), one for VNR and another one for a precisely selected SN path. The ***VNR segment*** (Seg^r) is defined as a set of parameters, grouped as one entity, representing demands of virtual nodes and edges. While ***SN path's segment*** (Seg^S) is defined as a set of parameters; also grouped as one entity, representing resources of the physical nodes and edges belonging to a specific selected SN path. Both, VNR and SN segments must be identical in terms of number of nodes and edges in order to compare them element by element. Subsequently, PaCoVNE starts VNE process to minimize total power consumption in the whole SN, by comparing each element in the VNR segment to its corresponding element in the SN path segment, then deciding if SN path has enough CPU and throughput resources to accommodate the VNR, while considering end-to-end delay.

Main contributions:

1) PaCoVNE heuristic is introduced as a one stage coordinated VNE approach constrained by CPU, BW, and end-to-end delay to minimize total power consumption of the whole SN.
2) Analysis of PaCoVNE was performed for off-line scenario, using homogeneous and heterogeneous SN settings.
3) Comparison was conducted against one of the most refer-

enced energy efficient embedding algorithms, the energy aware relocation heuristic (OCA/EA-RH) given by [9].

Rest of the paper is organized as follows: Section II provides related work. System model is introduced in section III, followed by ILP problem formulation in section IV. Design of the proposed PaCoVNE heuristic is shown in section V, and performance evaluation is presented in VI. Then results and discussion are included in section VII, while section VIII concludes the paper and highlights some future work.

II. RELATED WORK

One of the main benefits of network virtualization is its ability to consolidate network resources by hosting them on the same substrate resource, which allows for reducing energy consumption and cost [9],[13]. In most cases, saving energy in networks has been devoted to the reduction of energy consumption in a single networking device or parts of a device, and not power saving in the whole network, where unused resources could be but into sleeping mode or turned off completely. Other approaches performed VNE on small parts of the SN, then widen the area if no sufficient power resources were found on SN. Moreover, virtual resources can be migrated to balance the overall load in an energy efficient way, thus reducing the total power consumption of the network without compromising QoS or VNRs' acceptance ratio. More details about most related and recent literature about energy aware VNE approaches are summarized in the following paragraph:

A modified VNE algorithm was presented by [8], which prefers SN nodes consuming less power and selects edges in an energy efficient path, then in [9], they developed a scalable energy-aware reconfiguration heuristic approach, including embedding cost and load balancing. The heuristic considers a set of embedded VNRs as input to perform an energy efficient relocation of resources, without impacting the acceptance ratio. [10] proposed to maximize the accepted VNRs while minimizing the energy cost of the whole system. They followed two observations, first embed VN nodes on SN nodes that has lowest electricity price, second embed VN nodes on an already active SN as much as possible, then put other nodes that has no load into sleeping mode.

Moreover, [11] developed an embedding algorithm that embeds a subset of VNRs into a subset of cleanest SN resources in terms of CO_2 emissions resulting from the energy usage, while satisfying the VNR constraints. They constrained the VNE process by introducing link delay, packet loss, used energy source, VNR priority and location. The authors showed that the embedding guarantees reduced number of substrate resources and cost, faster embedding time, and reduction of carbon footprint of the VNE operation. While in [12], the authors designed an MILP and a real time heuristic algorithm that considers granular power consumption of all devices in an IP over WDM network. They tried to consolidate the nodes embeddings by filling the ones with the least residual capacity before switching on others, as well as consolidating more than virtual node at the same data center to minimize additional hop counts. And in [13] The authors considered an energy efficient VNE in the IP network over the WDM optical network, by adapting a feedback control approach performing the embedding on a smaller set of SN resources. A limited mappable area consisting of a selection of candidate nodes is located first, then they check if VN embedding was successful, if not, then a feedback control approach is triggered to search for a wider mappable area, and the whole process repeats again. In this way, they managed to increase number of hibernated links and nodes, resulting on reducing energy consumption by the SN.

III. SYSTEM MODEL

The aim of this paper is to perform coordinated VNE that minimizes total power consumption in the whole SN. Consequently, following paragraphs explain the overall design model for VNE system, starting by defining SN model and introducing its notations. Then VN's model definition and notation will be explained, as well as defining the used power consumption model by PaCoVNE.

A. Substrate Network Model:

The physical network $G^S = (N^S, E^S)$ is modeled as a weighted directed graph, where: i and $j \in N^S$ are SN nodes, and $(i,j) \in E^S$ is an edge connecting nodes i and j. Each node $i \in N^S$ is associated with pw_i^{idle} representing average power value when the sever is idle, PW_i^{Busy} average power value when the server is fully utilized, PC_i total power consumption of i, as well as cpu_i^a representing current available CPU capacity, cpu_i consumed CPU capacity, and CPU_i as the maximum CPU capacity at node i. μ_i is a fractional value (consumed to maximum CPU capacity, which could reach a value of 1 maximum) representing the processing utilization of node i defined in the range (0-1), zero if node i is not loaded, up to 1 if its 100% loaded. Each substrate edge (i,j) is associated with bw_{ij}^a, representing current available bandwidth capacity, bw_{ij} as consumed bandwidth capacity, BW_{ij} for maximum bandwidth capacity, d_{ij}^a as current end-to-end delay in SN edge (i,j), while $f_{i,j}^a$ is current traffic flow defined as the total throughput from SN node i to j. $P^S = \{(i,j)\}$ represents a set of all directed paths connecting all pairs of SN nodes i and j with a set of edges $\{(i,j)\}$. And substrate path $P_{sd} = \{(s,n),...,(k,l),...,(m,d)\} \in P^S$ is an end-to-end path constructed of more than one physical edge, where (s,n) is the first physical edge connecting the source node s to its adjacent node n, (k,l) is an intermediate physical edges, and (m,d) is the last physical edge connecting destination node d to its previous node m. Finally, total end-to-end delay in P_{sd} is the sum of delays of each edge (i,j) between the source node s and the destination d, and is given by $d_{sd}^a = \sum_{\forall\ (i,j) \in E^S} d_{ij}^a$.

B. Virtual Network Model:

Similar to the substrate network, the virtual network is modeled as a weighted directed graph $G^V = (N^V, E^V)$, where u and $v \in N^V$ are virtual nodes, and $(u,v) \in E^V$ is a virtual edge. VNR^r is a virtual network request number r out of R total VNRs. Each virtual node $u \in N^V$ is associated with cpu_u^r, representing the demanded CPU capacity, and each virtual edge (u,v) connecting a pair of virtual nodes u and v is also associated with bw_{uv}^r as the demanded bandwidth capacity. Lastly, d_{uv}^r represents the maximum allowed end-to-end delay demanded by virtual edge (u,v).

C. Power Consumption Model:

A comprehensive survey for state of the art power consumption models were presented in [1]. Accordingly, this paper identified the linear power model introduced by [14], which defined a formula to estimate the power consumption of network's servers PC including its idle power. The model approximated the aggregate behavior of a server system while being active, by measuring the total power consumption of the server PC_i against it's CPU utilization. In addition to idle power, the model includes total power consumed by the server when loaded as shown in Fig.(1). The formula is given as follows:

$\forall\ i \in N^S$

$$PC_i = pw_i^{idle} + [PW_i^{Busy} - pw_i^{idle}] \times \mu_i \tag{1}$$

$$\mu_i = \left(\frac{cpu_i}{CPU_i} \times 100 \right) \tag{2}$$

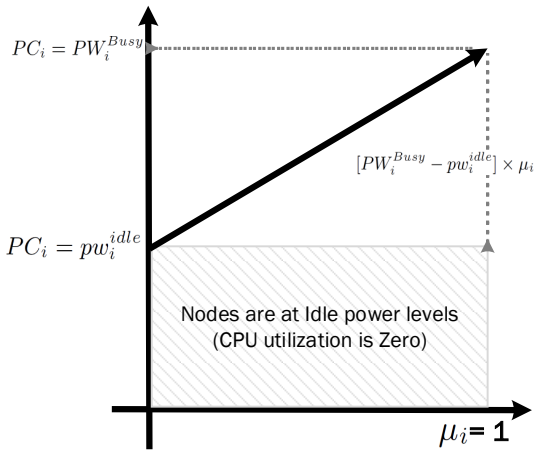

Fig. 1. Power Consumption Model

IV. PROBLEM FORMULATION

VNE problems are traditionally modeled as an optimization problem of objective function with positive integer and linear variables, usually referred to as integer linear programming (ILP) problem. However, optimal solution for the VNE as an ILP problem, implies introducing binary constraints to connect one edge only for each node, then, mapping all virtual nodes and edges on their physical counterparts having enough resources to accommodate their demands. Accordingly, virtual edges associated with bandwidth constraints is usually treated as a commodity between pair of nodes, and therefore, embedding a virtual edge optimally is similar to finding an optimal flow for the commodity in any network model [6],[15],[17].

To formally introduce the VNE problem as an ILP, following paragraphs define and formulate VNR and SN path's segments, in addition to the objective function and its constraints, as follows:

A. Segments formulation:

To solve VNE problem in one stage by fully coordinating nodes and edges embedding together at the same time, segment based formulations for the VNR and SN path are defined and formulated as follows:

Definition: Segment is defined as a set of parameters, grouped as one entity, for VNRs, Seg^r representing demands of virtual nodes and edges in VNR number r, and for a specific SN path, Seg^S represents resources of the physical nodes and edges in the selected SN path.

1) VNR segment formulation (Seg^r): each VNR is reformulated into a segment listing its CPU, BW, and delay demands together as a set. Eq(3) shows general design of Seg^r. Starting by the processing power capacity of its virtual nodes denoted by cpu_u^r for the source node, cpu_w^r for all intermediate nodes, and cpu_v^r for the destination node. Next, the segment lists all virtual edges' resources, including bandwidth capacity per each edge denoted by bw_{uo}^r for the edge connecting source virtual node u to next virtual node o, then it lists all bandwidth capacities for all intermediate virtual edges including virtual edge bw_{wx}^r, connecting intermediate virtual node w to next virtual node x, in addition to virtual path bw_{pv}^r connecting destination virtual node v to its previous virtual node p, and finally Seg^r lists the demanded end-to-end delay d_{uv}^r between the virtual source node u and its virtual destination node v.

$$Seg^r = \{cpu_u^r, .., cpu_w^r, .., cpu_v^r, bw_{uo}^r, .., bw_{wx}^r, .., bw_{pv}^r, d_{uv}^r\} \quad (3)$$

o, p, w, x are virtual nodes $\in$ VNR^r

2) SN path segment formulation (Seg^S): Similarly, P_{sd} segment is shown in eq(4). The segment lists all P_{sd} resources, namely: current available processing power capacities for all nodes in the path, given as cpu_s^a, cpu_k^a, and cpu_d^a for source, all intermediate, and destination physical nodes respectively. In addition, Seg^S lists current available bandwidth capacities for all of its edges starting by bw_{sn}^a, connecting source node s to next physical node n, all edges connecting intermediate nodes including bw_{kl}^a, and bw_{md}^a connecting destination node d to its previous physical node m. Lastly, Seg^S segment lists its end-to-end current delay d_{sd}^a between P_{sd} source and destination nodes.

$$Seg^S = \{cpu_s^a, .., cpu_k^a, .., cpu_d^a, bw_{sn}^a, .., bw_{kl}^a, .., bw_{md}^a, d_{sd}^a\} \quad (4)$$

k, l, n, m are physical nodes $\in P_{sd}^a$

B. Objective function definition and formulation:

Following the same analogy of estimating the power consumption of SN nodes, formula shown in eq.(1) will be applied to formulate the objective function as an ILP optimization problem. The main target is to minimize overall power consumption in the whole substrate network, by putting into sleeping mode all non utilized SN resources that are at idle power consumption, while accommodating VNR's demands. The rational behind that, is that for all SN nodes that are at idle mode, still, they are consuming considerable amount of power, even if their consumed CPU were zero. This is because when nodes are at idle mode, the power consumed by chassis (backplane) and cooling systems could be at least 40% or higher of the total power [19]. Accordingly, setting them into sleeping mode will result on minimizing the total substrate network's power consumption.

1) Objective Function: To make sure that a specific SN node is active and hosting at least one virtual node, variable x_i^{ur} is used in the ILP objective function formulation, which takes a binary value of (1) if substrate node i is active and assigned to host the virtual node u, and (0) otherwise. The objective function is shown in eq.(5) as follows:

$$\forall u \in N^V and \; \forall r \in R$$

$$\min PC_i = \sum_{\forall i \in N^S} (pw_i^{idle} + [PW_i^{Busy} - pw_i^{idle}] \times \mu_i) \times x_i^{ur} \quad (5)$$

C. Constraints definition and formulation:

Objective function solution will be constrained by capacity, flow and domain constraints as shown bellow. However, power consumption constraint was intentionally omitted, since it relies on CPU utilization of each node, nevertheless, it will be satisfied if constraints (6) and (7) were satisfied.

1) Capacity constraints: To ensure current available CPU processing power capacity in substrate node i is greater than or equal to demanded capacity by virtual network node u, constraint (6) is defined as follows:

$$\forall \; i \in N^S \quad cpu_i^a \geq cpu_u^r \quad (6)$$

To ensure total consumed CPU processing power capacity at substrate network node i, is less than or equal to maximum CPU capacity at that SN node, constraint (7) is defined as follows:

$$\forall \; u \leftarrow i \quad \sum_{r \in R} cpu_u^r \leq CPU_i \quad (7)$$

Note: $u \leftarrow i$ means that virtual network node u is hosted at substrate network node i.

To ensure that current available bandwidth capacity on substrate network edge (i, j) is greater than or equal to demanded

bandwidth capacity by virtual network edge (u, v), constraint (8) is defined as follows:

$$\forall\ (i,j) \in P_{sd} \quad bw_{ij}^{a} \geq bw_{uv}^{r} \tag{8}$$

To ensure that total consumed bandwidth capacity in substrate network edge (i, j), is less than or equal to maximum bandwidth capacity at that edge, constraint (9) is defined as follows:

$$\forall\ (u,v) \leftarrow (i,j) \quad \sum_{r \in R} bw_{uv}^{r} \leq BW_{ij} \tag{9}$$

Note: $(u, v) \leftarrow (i, j)$ means that virtual network edge (u, v) is embedded on the substrate network edge (i, j).

To ensure that current end-to-end delay in substrate network path P_{sd} is less than or equal to maximum allowed delay d_{uv}^{r} by VNR^{r}, constraint (10) is defined as follows:

$$d_{sd}^{a} \leq d_{uv}^{r} \tag{10}$$

2) Flow constraints: To ensure that a flow getting in a substrate node must go out, the following constraints has to be satisfied:

$$\sum_{\forall n \in N^{S}} f_{sn}^{a} - \sum_{\forall n \in N^{S}} f_{ns}^{a} = bw_{uo}^{r} \tag{11}$$

$$\sum_{\forall m \in N^{S}} f_{dm}^{a} - \sum_{\forall m \in N^{S}} f_{md}^{a} = -bw_{pv}^{r} \tag{12}$$

$$\sum_{\forall k,l \in N^{S}} f_{kl}^{a} = \sum_{\forall k,l \in N^{S}} f_{lk}^{a} \tag{13}$$

Constraint (11) ensures that the total flow getting out of source node s is the demanded flow bw_{uo}^{r}, while constraint (12) ensures that total flow getting into destination node d is the forwarded flow bw_{pv}^{r}, and constraint (13) ensures that all demanded flow is transfered from source to destination node, and nothing remains at any intermediate node within SN path P_{sd}.

3) Domain constraints: To solve the problem as ILP, constraint (14) is defined as follows:

$$\forall\ i \in N^{S}\ x_{i}^{ur} \in \{0, 1\} \tag{14}$$

To ensure each virtual node is mapped only to one substrate node, constraint (15) is defined as follows:

$$\forall\ u \in N^{V} \sum_{\forall i \in N^{S}} x_{i}^{ur} = 1, \tag{15}$$

To ensure virtual nodes from the same VNR are mapped to different substrate nodes, constraint (16) is defined as follows::

$$\forall\ i \in N^{S} \sum_{\forall u \in N^{V}} x_{i}^{ur} \leq 1, \tag{16}$$

V. HEURISTIC DESIGN

Optimal solution for VNE is known to be NP-Hard and computationally intractable, since it can be reduced to multi-way separator problem, which is NP-Hard by itself [7]. As a summary, [18] listed some of the main reasons highlighting why solving VNEs is challenging, such as: randomness of the arrival of VNRs depending on users' demands, topology and resources constraints by each VNR, and limited SN resources. However, the virtual edges embedding problem is what makes the VNE problem exceptionally an NP-hard, because it could be mapped to one or more physical edges that are not necessarily physically connected. Even for offline VNE case, given that all nodes were embedded, still virtual edge embedding stage can be reduced to the unsplittable flow problem, which is NP-hard [15],[16]. Consequently, solving VNE problem in polynomial time is not possible.

Therefore, majority of VNE approaches followed heuristic or meta-heuristic algorithms to solve VNE optimization problems in a reasonable polynomial time [5]. For example, one of the most referenced VNE heuristic approaches is the algorithm presented by [7]. It coordinates node and edge embedding, through mapping virtual nodes onto substrate nodes in a way that facilitates mapping of virtual edges. Nevertheless, the authors performed VNM and VEM in two interrelated stages. First they designed a node embedding algorithm to embed the virtual nodes on a suitable physical nodes, which could be separated a part from each other. Second, once node mapping was successful, they triggered another algorithm to embed the associated virtual edges on substrate paths, noting that it mostly would include hidden nodes to be used as hops. However, other ideas could be explored to better coordinate embedding VNM and VEM stages, and at the same time avoid including non necessary hidden hops and edges beyond VNRs needs.

Therefore, this paper proposed the PaCoVNE algorithm as a new heuristic methodology to solve VNE optimization problem more efficiently. Its main strength, is that it coordinates node and edge embedding in one step, based on matching each element in VNR^{r} segment, Seg^{r}, against their counterparts in the SN path's segment, Seg^{S}, considering the following four constrains, namely: CPU and BW capacity constraints, in addition to power consumption and end-to-end delay constraints.

A. Heuristic code explained:

Pseudo-code for PaCoVNE heuristic is shown in Algorithm 1 bellow, and is explained by the following main four steps:

1) ***Initialization****:* it starts by generating SN topology, lists all its possible paths, and categorizes them into types according to number of nodes and edges per each SN path. Notice that, number of lists and paths per list varies depending on the size and topology of SN. Since SN topology is physically fixed in real life, the main elements formulating any SN path (number and connectivity of SN nodes and edges) are also fixed and does not change, but only their capacities varies due to consumption. Therefore, to avoid searching for SN paths while VNE algorithm is running, and in contrary to most available heuristics in literature, this paper performs the initialization step in advance ahead of VNRs' arrival. This is one advantage behind PaCoVNE's speed of performing VNE in real-time, given it mainly focuses on the actual mapping process itself. To facilitate recalling a specific list of SN paths by PaCoVNE algorithm whenever it receives a new VNR, these lists will be saved and categorized per path type in a data base repository, including number of nodes, edges, and connectivities for each path.

2) ***Segmentation and ranking****:* this is the differentiating aspect of PaCoVNE heuristic compared to others, mainly because it facilitates accommodating VNRs one by one and embed their nodes and edges in one step and in full coordination between VNM and VEM. First the heuristic formulates VNR^{r} segment Seg^{r}. Then, to formulate the candidate SN path segment Seg^{S}, it recalls the appropriate list of SN paths that has similar number of nodes and edges as that of VNR^{r}. Next, it ranks them according to their CPU utilization, and ends by formulating SN segment for the top ranked path.

3) ***Embedding decision****:* compares each element in the SN segment Seg^{S} to its counterpart in the Seg^{r}, one-by-one. Accordingly, if SN segment has enough resources to accommodate all demands of VNR^{r}, PaCoVNE selects the path of SN segment Seg^{S} to host VNR^{r}. Decision matrix for the embedding process is shown in eq.(17) bellow:

$$if\ cpu_{i}^{a} - cpu_{u}^{r} \geq 0\ and$$

$$if\ bw_{ij}^{a} - bw_{wx}^{r} \geq 0\ and$$

$$if\ d_{sd}^{a} \leq d_{uv}^{r} \tag{17}$$

*4) **Updating**:* once a successful embedding occurs, the heuristic updates all changed SN resources and moves to next VNR. However, in case that SN segment does not have enough resources to accommodate VNR demands, the heuristic jumps to the next ranked path, and follow on from step 5. This process keeps on going until no more VNRs to be handled.

Algorithm-1, PaCoVNE Pseudo-Code

1) Input: G^V.
2) **for** each VNR$^r \in R$ **do**
   -Formulate VNR^r parameters into segment Seg^r according to eq.(3).
3) For the set of all saved SN paths P^S:
   List all SN paths matching VNR^r size.
   Rank them in descending order based on μ_i according to eq.(2)
4) For top ranked SN path P_{sd}, formulate its segment Seg^s according to eq.(4).
5) **Compare** Seg^r against Seg^S
   Check for CPU, BW and Delay constraints according to eq.(17).
6) **If** satisfied,
   embed VNR^r on P_{sd}.
   else go to next ranked SN path, step-4.
7) **for** all SN nodes and edges **do**
   Update CPU and BW resources.
   Remove the embedded VNR^r from VNRs list.
8) **for** idle SN nodes **do**
   Turn-off to save power.
9) Evaluate Metrics.
10) **If** VNRs list not empty, **go** to next VNR step-2.

B. PaCoVNE Computational Time Complexity:

In this paper, regardless the number of VNRs and based on the adjacency matrix of SN, searching and listing all types of paths will consume $O(|N^S| + |E^S|)$ processing time, depending on total number of nodes N and edges E formulating the SN [17]. This step is performed and saved only once before the arrival of any VNR. Therefore, it will not have any impact on the real computational time complexity of the VNE process.

However, the actual VNE process starts when the first VNR arrives at the SN. Therefore, in order to evaluate computational time complexity of PaCoVNE at worst case, the focal computational component of the heuristic is determined based on the time consumed while sorting all listed SN paths that has the same number of nodes and edges as the VNR^r. The larger the number of listed paths, the more computational time is consumed by the working machine.

Accordingly, for each VNR^r, the PaCoVNE adopted (Bubble Sort) algorithm to rank all SN paths in descending order [17]. Thus, at the worst case, the PaCoVNE algorithm will have a quadratic computational time complexity in the order of $O(n^2)$, where n is number of paths.

C. Illustrative Example:

A detailed example to explain the proposed heuristic is shown in fig.(2). It applies PaCoVNE on a SN of four nodes as shown in stage A, then it evaluates how to accommodate VNR^1, by sorting all listed SN paths based on the total sum of CPU utilizations 'μ' for each path. As shown in stage B, the PaCoVNE concludes by embedding VNR^1 on nodes 2 and 3, along path P_{23}, which had enough resources to accommodate its demands. In this case, the heuristic managed to save 21% of the total consumed power in the whole SN, by turning-off nodes 0 and 1, since they were idle. Stage C introduced VNR^2, the PaCoVNE decides that even though P_{23} is still the top ranked path, based on its CPU utilization, but since it does not have enough BW resources to accommodate the demanded BW by VNR^2, it jumps to next ranked SN path, P_{02}, which satisfies all demands of VNR^2.

TABLE I
SIMULATION SETTINGS FOR OFF-LINE HOMOGENEOUS

Parameter	SN	VNR
Nodes	50	15
$CPU\ max$	100	2.1
$BW\ max$	100	2.3
$Delay\ max$	250	$100 - 250$
PW^{Busy}	524	
pw^{idle}	$PW^{Busy} * 0.4$	
Loads	$0.2 - 0.9$	
Runs/load	50	
α	0.6	
β	0.23	
p_{wax}	0.2	

Therefore, PaCoVNE assigns P_{02} to accommodate VNR^2, then it keeps node 1 turned-off, since its the only idle node, resulting on saving 12% of the total consumed power by the whole SN.

VI. PERFORMANCE EVALUATION

In this paper, off-line version of PaCoVNE heuristic was tested using homogeneous and heterogeneous settings, once with end-to-end delay, and another time without it. The homogeneous version was compared to one of the most referenced heuristics, the energy aware relocating algorithm 'OCA/EA-RH' developed by [9]. Then, for the heterogeneous scenario, PaCoVNE was compared to its homogeneous version.

A. Simulation Settings:

For the off-line homogeneous scenario without delay, PaCoVNE was compared to OCA/EA-RH heuristic, which only used VNRs of 15 nodes; denoted as ($VNRs_{15}$), and therefore, the same simulation settings will be applied for PaCoVNE as well [9]. Specifically, the SN will handle a set of 80 VNRs, for different average loads, denoted by $\rho \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The value of each load ρ, reflects a ratio between VNRs demands to SN capacities, and therefore, loading the SN with X%, means this is the average of loading each node by X% load as well. The values of SN's nodes cpu_i^a and bw_{ij}^a resources were set to uniformly distributed values equal to 100. For each VNR, cpu_w^r and bw_{wx}^r values were estimated from the average embedding cost figure of [9], and are given as follows: $cpu_w^r = 2.1$ and $bw_{wx}^r = 2.3$. Finally, maximum power consumption by each SN node PW_i^{Busy} was set to 524 watts, while its idle power pw_i^{idle} was set as (PW_i^{Busy} * 0.4) [19]. For SN edges, end-to-end delay d_{ij}^a, was set equal to $250ms$ as a limit [20],[21]. While virtual network delays d_{wx}^r, was selected pseudorandomly between $100 - 250ms$. Table (1) summarizes all simulation settings to compare PaCoVNE against OCA/EA-RH for the off-line and Homogeneous scenario.

SN topologies were generated as directed graphs, through Waxman algorithm according to the following parameters: $\alpha = 0.6$, $\beta = 0.23$, and mean probability of creating an edge between any two SN nodes, denoted as p_{wax} was set equal to 0.2. Important to notice that, these parameters differ from what [9] used, since the aforementioned parameters will provide average edges at each SN node of 6, instead of 12 as used by [9]. This caused PaCoVNE heuristic to rank much less number of paths, yet, it produced better results compared to OCA/RA-EH. To overcome the probabilistic nature of Waxman topology generation, the set of 80 VNRs were run for 50 times per each ρ load value.

B. Heuristic work-flow:

Initialization: based on the SN adjacency matrix, the heuristic lists all SN paths of 15 nodes, denoted as $P_{15} \in P^S$, this is only performed once and saved at the beginning. These paths can then be used for any number of $VNRs_{15}$. This is important, since

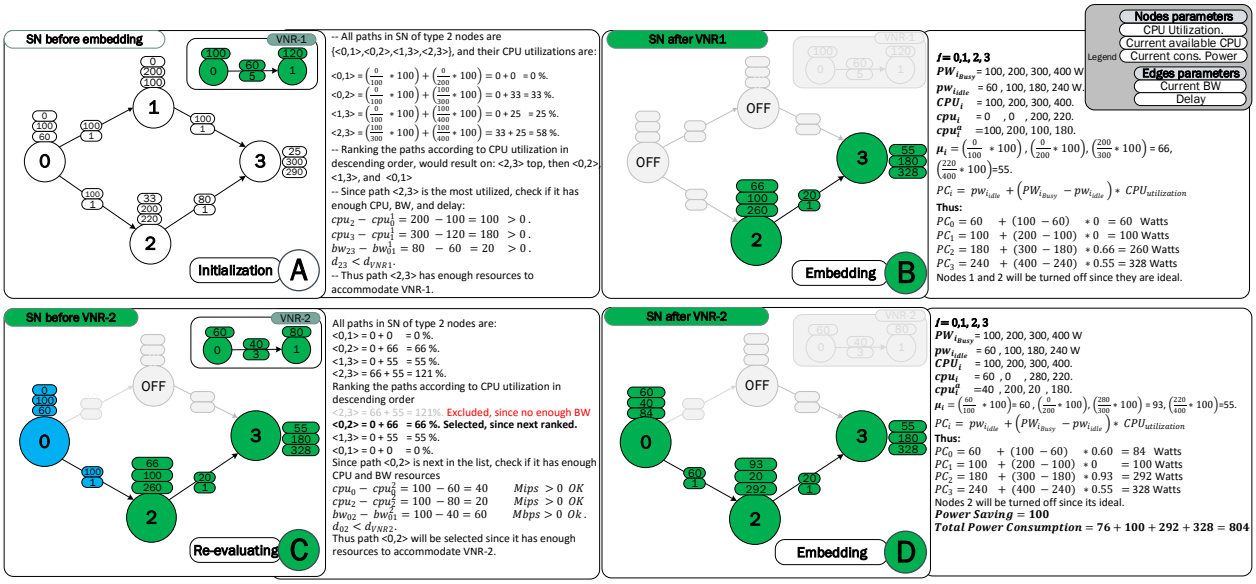

Fig. 2. Numerical example showing basics of PaCoVNE

the heuristic will focus on the actual embedding process itself, and not on searching for the best path at the arrival of each new VNR, which saved PaCoVNE's run time considerably. Moreover, its important to mention that PaCoVNE heuristic can generate all types of SN paths at the beginning, and accordingly, it can handle any type of VNRs regardless of how many nodes they may contain.

VNR segment formulation: Since OCA/EA-RH heuristic used VNRs$_{15}$, then PaCoVNE formulates Seg^{r15} as defined in eq.(3).

SN path segment: the heuristic selects the path of highest μ, denoted as $P^a \in P_{15}$, and formulates its segment Seg^{a15} as defined in eq.(4).

Ranking: For each path $P^a \in P_{15}$, PaCoVNE calculates and sums its μs, then it ranks the paths based on the value of its μ from highest to lowest.

Embedding: The algorithm compares both segments according to eq.(17), if the conditions are satisfied, then it embeds VNR^{r15} on SN path P^a.

Turning off idle SN nodes: Once VNR^{r15} is embedded successfully, PaCoVNE identifies all idle SN nodes and turns them off to save power consumption. Next it updates all SN elements based on that.

C. Evaluation Metrics:

The PaCoVNE heuristic will be evaluated according to following metrics:

- *Average Total power consumption, PW:* defined as the total power consumed by all SN nodes after each VNR embedding, and averaged over the total number of VNRs R [9],

$\forall \rho \in$ *Loads*,

$$PW = \frac{1}{R}\Big(\sum_{\forall r \in R}\sum_{\forall i \in N^S} PC_i\Big) \tag{18}$$

- *Average Saved Power, PS:* the amount of saved power after embedding each VNR, using the proposed power reduction strategy. Calculated by subtracting total power consumed by all SN nodes without power reduction strategy PW^-, from total power consumed by all active SN nodes after applying power reduction strategy PW^+. The results will be averaged over the total number of VNRs R.

$\forall \rho \in$ *Loads*,

$$PS = \frac{1}{R}\sum_{\forall r \in R}\Big(\sum_{\forall i \in N^S} PW^- - \sum_{\forall i \in N^S} PW^+\Big) \tag{19}$$

- *Average Acceptance Ratio, AR:* is a ratio to represent how PaCoVNE algorithm is performing, calculated for each load value ρ, by dividing number of successfully embedded VNRs by total number of VNRs R [7],[9].

$\forall \rho \in$ *Loads*,

$$AR = \frac{1}{R}\ Total\ Number\ of\ Embdded\ VNRs * 100 \tag{20}$$

- *Average Cost of embedding VNRs, EC:* is the sum of total consumed SN resources CPU and BW while embedding each VNR. Tuning parameters to represent relative costs per each SN resource, denoted as α for SN nodes' cost, and β for SN edges, were both set equal to one [7],[9].

$\forall \rho \in$ *Loads*,

$$EC = \frac{1}{R}\sum_{\forall r \in R}\Big(\sum_{\forall (i,j) \in E^S}(\beta * bw_{ij}) + \sum_{\forall i \in N^S}(\alpha * cpu_i)\Big) \tag{21}$$

- *Average CPU utilization, CPU_{util}:* it represents SN nodes' utilization trend after all simulation iterations. Its defined as ratio between consumed CPU cpu_i, and maximum CPU resources, averaged overall VNRs for each load ρ [7].

$\forall \rho \in$ *Loads*,

$$CPU_{util} = \frac{1}{R}\sum_{\forall r \in R}\Big(\sum_{\forall i \in N^S}\frac{(CPU_i - cpu_i^a)}{CPU_i} * 100\Big) \tag{22}$$

- *Average BW utilization, BW_{util}:* it represents utilization of SN edges after all simulation iterations. And is defined as ratio between consumed bw_{ij}, and the maximum BW, averaged overall VNRs for each load ρ [7].

$\forall \rho \in$ *Loads*,

$$BW_{util} = \frac{1}{R}\sum_{\forall r \in R}\Big(\sum_{\forall (i,j) \in E^S}\frac{(BW_{ij} - bw_{ij}^a)}{BW_{ij}} * 100\Big) \tag{23}$$

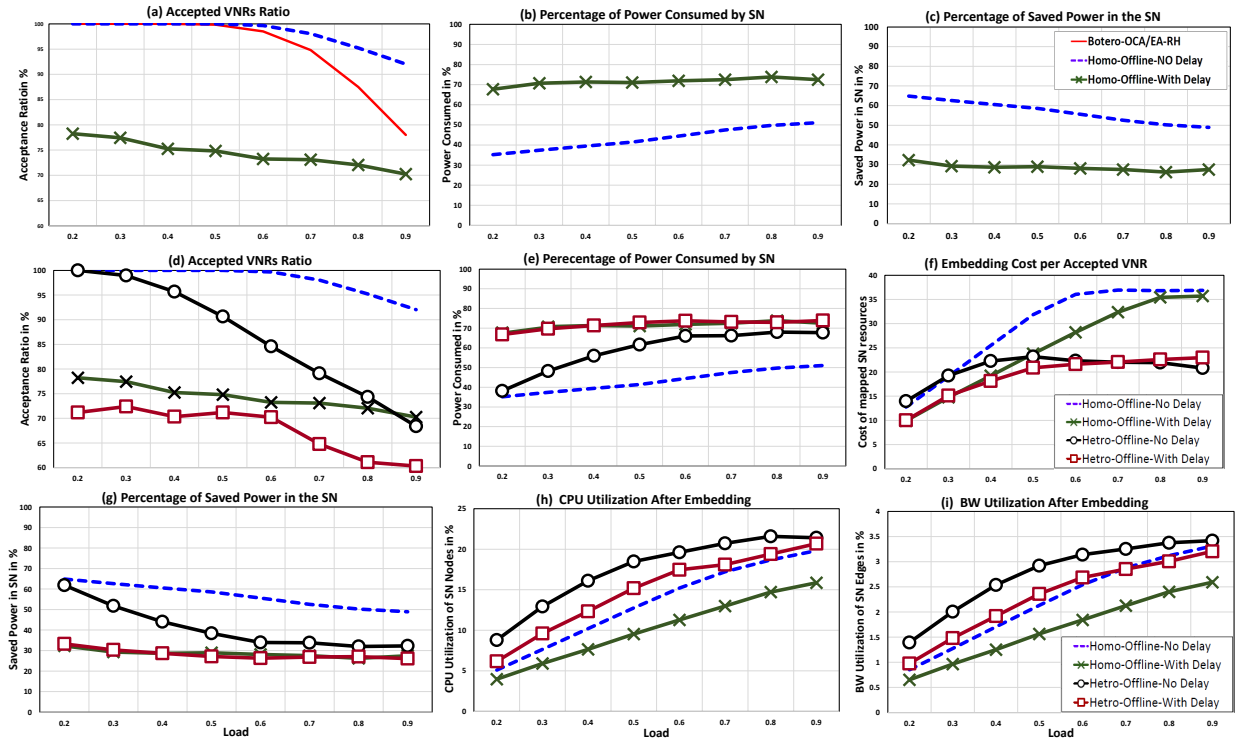

Fig. 3. Comparison Results of PaCoVNE Homogeneous against OCA/EA-RH and PaCoVNE Heterogeneous

VII. RESULTS AND DISCUSSION

A. *Off-line Homogeneous scenario:*

Simulation results in fig.(3a) shows that PaCoVNE performed similar or better than EA-RH in terms of acceptance ratio for lower loads, and is much better for higher loads, thanks to the one stage, full coordinated embedding and segment formulation of PaCoVNE, which makes sure to allocate virtual nodes and their associated edges together, and at the same time, thus, increasing the acceptance ratio. In comparison, the OCA/EA-RH relocates least stressed virtual nodes and their associated edges to other suitable active SN nodes that are more stressed, using the cost-based VNE approach of [7]. Then in a separate phase, OCA/EA-RH relocates least stressed edges to shortest energy path.

However, in terms of power consumption at SN nodes, PaCoVNE can not be compared to OCA/EA-RH, since both algorithms used different formulas to calculate the consumed power per each SN node. Nevertheless, fig.(3b) shows that SN's power consumption is still high, giving that PaCoVNE model includes idle power in addition to power consumption when SN nodes were loaded according to their CPU utilization. This entails the importance of considering idle power as a main component for increasing power consumption of SN's nodes, even if they do not process any data.

Moreover, regarding saved power results shown in fig.(3c) clarifies that, when the load was 0.2 PaCoVNE managed to save 65% of SN's total power, and when the load was much increased to 0.9 it saved 49%, implying that, in a range of loads between 0.2 to 0.9, PaCoVNE would save in average 57% of SN's total power consumption, by putting idle nodes into sleeping mode, while maintaining high VNE acceptance ratios across almost all loads. These results highlights the benefits of using PaCoVNE's new segmentation strategy to fully coordinate VNE, also pinpoints the obvious impact of idle power consumption on the overall SN's power consumption, thus, reducing it would ultimately reduce SN costs. Indeed, important to point out that in real life conditions, putting idle nodes into sleeping mode as a power reduction strategy, could affect service maintainability of SN, especially considering on-line scenarios. Therefore, other strategies could be explored as well.

TABLE II
SIMULATION SETTINGS FOR OFF-LINE HETEROGENEOUS SCENARIO

Parameter	SN	VNR
Nodes	50	15
CPU_{max}	$Random$ 40 – 100	$Random$ 1.5 - 2.1
BW_{max}	$Random$ 40 – 100	$Random$ 1.6 - 2.3
$Delay_{max}$	Random 100 – 250	Random 100 – 250
PW_{Busy}	524	
pw_{Idle}	PW_{Busy} * 0.4	

In the case of including end-to-end delay, fig.(3a, 3b, and 3c) shows the obvious impact of end-to-end delay. In comparison to PaCoVNE homogeneous without delay, acceptance ratio was degraded by 24% in average for all loads, increased power consumption by 57%, and reduced saved power by 51%. These results implies the significance of including end-to-end delay as a main constraint to embed VNRs, and how negatively it would impact the whole VNE process.

B. *Off-line Homogeneous against Heterogeneous:*

The rational behind comparing PaCoVNE using homogeneous to heterogeneous configuration is to give some insights about how PaCoVNE would behave on semi-real life conditions, where SN resources usually differ in size and capacity, in addition to including end-to-end delay. Table-2 summarizes the heterogeneous simulation settings.

Fig.(3d, 3g, 3h, and 3i) shows simulation results considering heterogeneous conditions, indicating the out-performance of homogeneous-PaCoVNE in terms of acceptance ratio, saved power, CPU and BW utilizations with and without end-to-end delay. In terms of power consumption and embedding cost as shown in fig.(3e and 3f), heterogeneous-PaCoVNE performed

much worse than homogeneous in both of them, mainly due to PaCoVNE's rapid tendency to utilize the SN resources. This is clearly translated into less accepted VNRs as loads increases. In addition to that, almost same conclusion can be deduced when end-to-end delay was applied, showing that the resultant metrics for the heterogeneous performed even much worse than the homogeneous across all metrics and loads, doubling down the significance of including end-to-end delay as a main VNE constraint.

VIII. CONCLUSIONS

This paper introduced the PaCoVNE heuristic, which performed VNE in a more efficient methodology than other algorithms in the literature. Performance of the heuristic was evaluated using homogeneous and heterogeneous configurations, once without considering end-to-end delay, as a constraint, and also when delay was included. Simulation results showed that, for the homogeneous scenario and when end-to-end delay was not included, the new heuristic managed to save considerable amount of substrate network's power consumption by 57% in average, for a range of loads between 0.2 to 0.9, through putting idle nodes into sleeping mode, while maintaining high VNE acceptance ratios, thanks to the new coordinated VNE approach. However, when end-to-end delay was factored in, PaCoVNE performance resulted on both, less saved power and acceptance ratio in comparison to homogeneous without delay. Suggesting that, introducing end-to-end delay, as in the real world and as a major constraint, had clear impact on the whole VNE process. On the other hand, when PaCoVNE in homogeneous setting was compared to heterogeneous version, the heuristic's performance degraded across all evaluation metrics, and specifically when end-to-end delay was included. Thus, doubling on the critical importance of considering delay as a major guiding principle to perform the VNE process in acceptable levels that could be applicable to real world applications.

The following points are the main outcomes of this paper:

1) PaCoVNE provided a new and better strategy to fully coordinated VNM and VEM simultaneously and in one step, thanks to the segmentation design concept.
2) The new strategy resulted on clear enhancements on VNE acceptance ratio in comparison to literature, fundamentally due to the advantageous of one stage embedding.
3) Most significant, was PaCoVNE's capabilities to save a very considerable amount of total power consumption of SN elements. Mainly due to the very precise embeddings, which allowed for efficiently distributing VNRs on the most powerful SN nodes, and consequentially, enabled a better identification methodology for the more idle SN nodes to turn them off.
4) However, when end-to-end delay was included, it significantly impacted VNE process, as reflected by lower acceptance ratios. Suggesting the importance of including end-to-end delay as a major VNE constraint.
5) Depending on the size of the VNR, the time consumed by the PaCoVNE heuristic to embed the VNR successfully varies significantly. The larger the number of nodes per a VNR, the more time it takes to embed it. This suggests that for large networks, the PaCoVNE should partition the VNRs and physical paths into smaller portions to speed up the embedding time. Thus, even thought the solution for some partitions may be sufficient, but it may not be as sufficient when aggregating the solutions for all partitions of the selected substrate network path.

As a future work, the authors are planning to extend the application of PaCoVNE to work for the online scenarios. Also, in addition to CPU utilizations to rank SN paths, other criterion can be studied, such as: edges utilizations, or their propagation delay. Moreover, other non linear parameters can be considered to evaluate the performance of the PaCoVNE, namely, what would be the impact of jitter, packet-loss, and grade of service on the VNE process given the segmentation strategy used by the PaCoVNE.

IX. ACKNOWLEDGMENT

This work has been partially supported by the Ministerio de Economía y Competitividad of the Spanish Government under project TEC2016-76795- C6-1-R and AEI/FEDER, UE.

REFERENCES

[1] M. Dayarathna, Y. Wen and R. Fan, "Data Center Energy Consumption Modeling: A Survey," in IEEE Communications Surveys and Tutorials, vol. 18, no. 1, pp. 732-794, Firstquarter 2016.
[2] ESTI, Network Functions Virtualisation, Introductory White Paper, October,2012.
[3] 5G PPP Architecture Working Group, "View on 5G Architecture," Version 1.0, 2016.
[4] Rachid El Hattachi, and Javan Erfanian, NGMN 5G White Paper, 2015.
[5] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer and X. Hesselbach, "Virtual Network Embedding: A Survey," in IEEE Communications Surveys and Tutorials, vol. 15, no. 4, pp. 1888-1906, 2013.
[6] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," ACM SIGCOMM CCR, vol. 38, noi. 2, pp. 17-29, 2008.
[7] M. Chowdhury, M. R. Rahman and R. Boutaba, "ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping," in IEEE/ACM Transactions on Networking, vol. 20, no. 1, pp. 206-219, Feb. 2012.
[8] J. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. de Meer, "Energy efficient virtual network embedding," Communications Letters, IEEE, vol. 16, no. 5, pp. 756-759, 2012.
[9] J. Botero, X. Hesselbach, "Greener networking in a network virtualization environment," Computer Networks, vol 57, issue 9, pp. 20121-2039, 2013.
[10] Sen Su, Zhongbao Zhang, Alex X. Liu, Xiang Cheng, Yiwen Wang, and Xinchao Zhao, "Energy-Aware Virtual Network Embedding," IEEE/ACM Transactions on Networking, vol. 22, no. 5, pp. 1607-1620, 2014.
[11] Nizar Triki, Nadjia Kara, May El Barachi, Souad Hadjres, "A green energy-aware hybrid virtual network embedding," Computer networks, Vol. 91, pp. 712-737, 2015.
[12] Leonard Nonde, Taisir E. H. El-Gorashi, and Jaafar M. H. Elmirghani, "Energy Efficient Virtual Network Embedding for Cloud Networks," Journal of Lightwave Technology, Vol. 33, No. 9, pp. 1828-1849, 2015.
[13] Xiaohua Chen, Chunzhi Li, and Yunliang Jiang, "A feedback control approach for energy efficient virtual network embedding," Computer Communications, Vol. 80, pp. 16-32, 2016.
[14] X. Fan, W. D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in Pro. 34th Annu. ISCA, pp. 13-23, 2007.
[15] Bradley, Hax and Magnanti, "Applied Mathematical Programming," Chapters-8 and 9, Addison-Wesley, 1977.
[16] S. G. Kolliopoulos and C. Stein, "Improved approximation algorithms for unsplittable flow problems," Proceedings 38th Annual Symposium on Foundations of Computer Science, Miami Beach, FL, pp. 426-436, 1997.
[17] J. Kleinberg and E. Tardos, "Algorithms Design," Addison-Wesley, 2009.
[18] Ilhem Fajjari, "Resource Allocation Algorithms for Virtual networks within Cloud Backbone Network," PhD Thesis, Pierre et Marie Curie University, France, 2012.
[19] Telecommunications Infrastructure Standard for Data Centers. http://www.tia-942.org/.
[20] ITU, Draft new Report ITU-R M. [IMT-2020.TECH PERF REQ], Minimum requirements related to technical performance for IMT-2020 radio interface(s)", ITU, Document 5.40-E, 22 February, 2017.
[21] G. Almes, S. Kalidindi, M. Zekauskas, amd A. Morton, A One-Way Delay Metric for IP Performance Metric (IPPM), IETF, RFC-7679, 2016.