PCPV: Pattern-based Cost-efficient Proactive VNF placement and chaining for value-added services in content delivery networks

Shohreh Ahvar, Jagruti Sahoo, Ehsan Ahvar, Mouhamad Dieye, Roch Glitho, Halima Elbiaze, Noel Crespi

Accepted in IEEE Netsoft

PCPV: Pattern-based Cost-efficient Proactive VNF placement and chaining for value-added services in content delivery networks

Shohreh Ahvar
Telecom SudParis
Evry, France
shohreh.ahvar@telecom-sudparis.eu

Jagruti Sahoo
South Carolina State University
Orangeburg, USA
jagrutiss@gmail.com

Ehsan Ahvar
Univ. Rennes, Inria, CNRS, IRISA
Rennes, France
ehsan.ahvar@inria.fr

Mouhamad Dieye
Université du Québec à Montréal
Montral, Canada
dieye.mouhamad@gmail.com

Roch Glitho
Concordia University
Montreal, Canada
glitho@ece.concordia.ca

Halima Elbiaze
Université du Québec à Montréal
Montreal, Canada
elbiaze.halima@uqam.ca

Noel Crespi
Telecom SudParis
Evry, France
noel.crespi@mines-telecom.fr

***Abstract*—Value-added services (VASs) are an integral part of todays Content Delivery Networks (CDNs). They can be implemented as a chain of Virtual Network Functions (VNFs). These chains need to be placed in an efficient way in CDNs in order to optimize quality of service (QoS) for end-users(EUs) while minimizing cost for providers. We formulate the problem as an Integer Linear Program (ILP) and propose a Pattern-based Cost-efficient Proactive VNF placement and chaining (PCPV) algorithm. The objective is to find the optimal number, location and chaining of VNFs in such a manner that the cost is minimized while QoS is met. Apart from cost minimization, the support for large-scale CDNs with a large number of physical machines (servers) and EUs is an important feature of the proposed algorithm. Through simulations, the algorithm behavior for small-scale to large scale CDN networks is analyzed.**

Keywords*—*Content Delivery Networks, Network Function Virtualization, Virtual Network Function, Cost, Placement.

I. Introduction

Content Delivery Networks (CDNs) are largely distributed infrastructures of surrogate servers (hereafter reffered to as server) placed in strategic locations [1] to serve end-users (EUs) with reduced latency. Value-added services (VASs) now play a very important role in CDNs. Now around 47% of Akamais revenues come from its VAS offerings [2]. Some examples are website/application acceleration (e.g., route optimization, TCP optimization, stream splitting) [3], analytics, content protection, advertisement overlays/tickers and content adaptation (e.g., transcoding, compression).

Network Function Virtualization (NFV) [4] could enable CDNs to provision VAS as chains of Virtual Network Functions (VNFs), which allows VASs to scale in an elastic manner. Besides, the updating of an existing VAS or the introduction of new VASs could be achieved with increased agility. We model the Placement and Chaining of VNFs (PC-VNF) for VAS as an optimization problem where the goal is to find the optimal number, location and efficient chaining of VNFs instances so that the CDN provider cost is minimized and QoS is satisfied.

This paper focuses on a proactive placement of VNFs where VNFs are deployed in an optimal manner before any request is received from the EUs to access the service. This type of deployment is triggered when a content provider requests the CDN provider to deploy a set of VASs. The rest of the paper is organized as follows: Section 2 presents PC-VNF problem in CDNs and ILP formulation. Section 3 describes the proposed algorithm. Section 4 portrays the simulation results. Section 5 concludes the paper.

II. VNF Placement Problem

A. Problem Description

Assume that content providers request the CDN provider to deploy a set of VASs. The request specifies the service-related parameters (e.g., the description of functionalities that constitute the services) including the QoS threshold to be satisfied. Given a content X, the servers containing X, a set of EUs requesting the content, their workload and a set of services to be accessed by the EUs, the PC-VNF problem targeted in this paper is to find the number and location of VNF instances along with assigning the Service Function Chains (SCs) to the EUs for minimizing the cost of CDN provider and satisfying the QoS threshold of all services offered to the EUs. The reader should note that for each EU, the delay for viewing the video is the sum of the following delays: 1. Delay from the server (selected to stream the video) to the first VNF of SC and; 2. Delay for transmission and processing the video in SC and; 3. Delay from the last VNF to the EU. Our problem is a slight variation of the well-known Bin Packing problem [5] and the Hierarchical Facility Location-Allocation problem [6] which is an NP-Hard problem, calling for an efficient heuristic

B. ILP Formulation

Let us consider N as a set of servers and U as a set of EUs. The physical topology of the network is represented by a directed graph $G = (V, E)$, where $V = N \cup U$ is the set of nodes composed of the servers and EUs connected by

TABLE I. INPUT PARAMETERS AND VARIABLES

	Network Inputs		Service Inputes
N	Set of servers in the network, $N \subseteq V$	H	Set of services h_f requested by user $f \in U$
U	Set of EUs in the network, $U \subseteq V$	K	Set of VNF types that constitute all services $h_f \in H$
E	The set of edges (i.e., logical communication links) in the network	V^f_{NF}	$V_f \subseteq K$ is a set of VNFs that constitute service $h_f \in H, f \in U$
$BW_{(u,v)}$	The bandwidth capacity of edge $(u,v) \in E$	E^f_{NF}	Set of VNF edges of the VNF chain for service $h_f \in H$, $f \in U$,
$D_{(u,v)}$	Delay of unit load (1 Gbps) for edge $(u,v) \in E$	I_k	Set of VNF instances of type $k \in K$
$\sigma(u,v)$	Hop count of the edge $(u,v) \in E$	α_k	Software license cost of a VNF instance of type $k \in K$
B_n	Bandwidth cost of unit load (1 Gbps) per hop server n	$T_{k,n}$	Processing delay of VNF instance of type $k \in K$ on server $n \in N$ for unit load (1 Gbps)
$\beta^f_{(u,v)}$	Bandwidth cost incurred by sending load of EU f along edge $(u,v) \in E$	R_k	Resource requirement for VNF type $k \in K$
C_n	Capacity of server $n \in N$ in terms of resource units	P_k	Processing capacity (Gbps) of VNF type $k \in K$
γ	Site license cost	L_f	Load of EU $f \in U$
δ_n	Operational cost for unit resource (vCPU) for server n	D_{h_f}	QoS (i.e., Service Delay),threshold of service $h_f \in H$
		Variables	
$x_{k,n,j}$	1, if instance j of VNF type k is assigned to server $n \in N$ and 0, otherwise	$y^{f,p,q}_{u,v}$	1, if edge (u,v) hosts VNF edge (p,q) of SC for EU f and 0, otherwise
$\lambda^f_{k,n,j}$	1, if VNF type k belonging to SC of EU f is mapped to its instance j on server n and 0, otherwise	z_n	1, if server n is used and 0, otherwise

directional edges E. Let us also consider K as a set of VNFs of different types, such as Video Transcoder, Video Mixer and Video Compressor. A service request $h_f \in H$, generated by EU $f \in U$, is represented by a directed graph $h_f = G(V^f_{\text{NF}}, E^f_{\text{NF}})$, where V^f_{NF} is the set of VNFs that will be installed on nodes in N. E^f_{NF} is the set of virtual edges that dictate the head and tail of SC. Table I delineates the inputs and variables used in our ILP formulation. Our objective is to minimize the cost shown in (1) including software license cost per instance, α_k and site γ, the sum of operational costs for all deployed VNF instances, the communication cost (i.e., sum of the bandwidth costs amongst each pair of servers $u, v \in N$ hosting VNFs of SC in each EU service request $h_f \in H$,$f \in U$, $\beta^f_{(u,v)}$).

Furthermore, it includes the bandwidth costs between the servers $u \in N$ hosting the tail VNFs of SC in each EU service request $h_f \in H$, and EU $f \in U$, $\beta^f_{(u,v)}$. Where, $\beta^f_{(u,v)} = L_f \cdot \sigma_{(u,v)} \cdot B_u$ is the cost of using bandwidth between $u \in V$ and $v \in V$, with hop count $\sigma_{(u,v)}$, for load L_f for EU $f \in U$.

$$\min \begin{cases} \left(\sum_{k\in K,n\in N,j\in I_k} x_{k,n,j} \times \alpha_k\right) + \\ \left(\sum_{f\in U,(p,q)\in E^f_{\text{NF}},(u,v)\in E} \beta^f_{(u,v)} \cdot y^{f,p,q}_{u,v}\right) + \\ \left(\sum_{f\in U,k\in V^f_{\text{NF}} \mid \forall i\in V^f_{\text{NF}} \nexists (k,i)\in E^f_{\text{NF}},u\in N,j\in I_k,(u,f)\in E} \beta^f_{(u,f)} \cdot \lambda^f_{k,n,j}\right) \\ + \left(z_n \times \sum_{n\in N}(\gamma + \sum_{k\in K,j\in I_k}(R_k \cdot \delta_n \cdot x_{k,n,j}))\right) \end{cases} \tag{1}$$

VNF Placement:

$$\sum_{n\in N}\sum_{j\in I_k} \lambda^f_{k,n,j} = 1, \quad \forall f \in U, k \in K \tag{2}$$

$$\sum_{f\in U} L_f \cdot \lambda^f_{k,n,j} \le P_k \cdot x_{k,n,j} \quad \forall k \in K, n \in N, j \in I_k \tag{3}$$

$$\sum_{k\in K}\sum_{j\in I_k} R_k \cdot x_{k,n,j} \le C_n \cdot z_n \quad \forall n \in N \tag{4}$$

VNFs chain mapping:

$$\sum_{j\in I_p} \lambda^f_{k,u,j} \cdot \sum_{j'\in I_q} \lambda^f_{k,v,j'} = y^{f,p,q}_{u,v} \tag{5}$$
$$\forall f \in U, (p,q) \in E^f_{\text{NF}}, (u,v) \in E$$

$$\sum_{(u,v)\in E} y^{f,p,q}_{u,v} = 1 \quad \forall f \in U, (p,q) \in E^f_{\text{NF}} \tag{6}$$

$$\sum_{f\in U}\sum_{(p,q)\in E^f_{\text{NF}}} L_f \cdot y^{f,p,q}_{u,v} \le BW_{(u,v)} \quad \forall (u,v) \in E \tag{7}$$

QoS guarantee:

$$\begin{aligned} &\sum_{(p,q)\in E_f,(u,v)\in E,\forall f\in U} D_{(u,v)} \cdot L_f \cdot y^{f,p,q}_{u,v} \\ &+ \sum_{k\in V_f \,/\, \forall i\in V_f \exists (k,i)\in E_f,u\in N,j\in I_k,\forall f\in U} T_{k,u} \cdot L_f \cdot \lambda^f_{k,u,j} \\ &+ \sum_{\substack{k\in V_f \,/\, \forall i\in V_f \nexists (k,i)\in E_f, \\ u\in N,j\in I_k,(u,f)\in E,\forall f\in U}} (T_{k,u} + D_{(u,f)}) \cdot L_f \cdot \lambda^f_{k,u,j} \le D_{h_f} \end{aligned} \tag{8}$$

Variable $x_{k,n,j}$ $\forall k \in K, n \in N, j \in I_k$ is used to identify unique instance $j \in I_k$ of VNF type $k \in K$ on server $n \in N$. Variable z_n $\forall n \in N$ is used to record servers hosting VNFs.

We ensure in (2) that for each EU service request $h_f \in H$, $f \in U$, an instance of the requested VNF type must be assigned to a server. However, an instance of a VNF type can cater to multiple users. (3), (4) and (6) ensure that the VNF instances, the host servers and links are not overloaded respectively.

The edge (p,q) between two consecutive VNFs in each EU service request must be assigned to a physical edge (u,v) between two servers u and v, in (5). As ensured in (6), the VNFs and their respective ordered edges are mapped to only one pair of physical servers and their edge. (8) is QoS constraint which includes the VNF processing delay on the surrogates, the network communication delay between the VNFs and the delay from the EUs to the last VNFs of SCs.

III. PATTERN-BASED COST-EFFICIENT PROACTIVE VNF PLACEMENT (PCPV)

PCPV is run once for each SC which is accessed by a set of EUs. VNF placements should satisfy QoS and prevent VNF and server from overloading. PCPV has two initial placement and modification phases. Phase 1 simplifies PC-VNF by focusing on QoS satisfaction and relaxing the VNF overloading constraint. To simplify the PC-VNF, two assumptions are considered: (i) users can be located everywhere in the network (ii) VNFs do not have capacity limitations. Considering the simplified problem, phase 1 finds the minimum number of instances of each VNF to cover the entire network with promised service delay, selects appropriate servers to deploy the instances and chains the deployed VNFs. Phase 2, if necessary, modifies the initial PC-VNF (i.e., adding/removing the selected servers and VNFs) based on extra information such as the VNF's capacity, the predicted EU's load and location information to prevent VNF (and server) from overloading.

A. Phase 1: Initial VNF Placement and chaining

To initially place the VNFs, we propose partitioning and patterning techniques. Due to the limited capacity of servers, it may possible a SC cannot be allocated on a server. Therefore, the phase 1 first divides SC to some partitions (or PTs) composed of one or more VNFs and is located on one of the servers. Phase 1 then finds the minimum number of needed instances of each PT to cover the network with promised QoS and selects appropriate servers for placing these PT instances on them. We do both of them at the same time using patterning technique. To this end, for each PT, phase 1 creates a pattern.

By mapping the patterns into the network, the minimum number of needed instances to cover the whole network is found and also location (servers) of these PT instances is estimated at the same time. This pattern-based method provides a scalable method for any size of network where the source of the chain is unknown in advance. The five parts of phase 1 are described as follows.

1) Chain Partitioning: PCPV divides a chain h into several PTs, *PartNum*. The maximum possible size of PTs equals to the minimum capacity, C, of the servers. To accomplish partitioning, PCPV creates and fills the first PT. If still some VNFs of the chain remain, PCPV fills the second PT. This procedure continues until all VNFs of the chain are selected.

In case of VNF fragmentation during filling a PT, the fragmented VNF is migrated to the next PT. After partitioning, the capacity of the smallest VNF of PT i is defined as $PartCap[i]$ and the total requirement of the VNFs of PT is considered as PT requirement $Re[i]$. Also, PTs numbered in ascending order from the chain head to the EU. Each PT i receives the content as input and yields the processed output content to the next PT i+1 in SC. The output of the last PT in SC is delivered to the EU. Thus, the EUs are directly connected to the server hosting the last PT of SC. We define the ones (i.e., EUs or another PTs of the chain) who receive the content from an instance of the PT as the customers of that PT instance.

2) Pattern designing: This section first presents the proposed patterns. Then, it describes why and how a pattern size is adapted based on the service characteristics (e.g., the maximum Delay Threshold (DT) and the number of created PTs (chain length)).

Proposed patterns– After finding the number of PTs for SC, PCPV defines a pattern for each created PT. In order to design a pattern, for the sake of QoS fairness, the best idea is to deploy an instance in the center of an area to reduce the number of needed instances for satisfying a requested QoS. However, finding a server exactly in the center of an area is not always possible (due to equipment limitations). To this end, PCPV considers a square shape zone (instead of a point) in the center of the area to deploy an instance (second part of this section explains how we set the zone size to make sure at least one server located inside it). For the sake of unity, we design our patterns based on this square shape zone unit with edge of d. PCPV also takes into account location of PT customers to define pattern for each PT. We present our propose patterns with a simple example. Assume a chain including three PTs. Where the EUs are customers of PT 3 instances, PT 3 instances themselves are customers of PT 2 instances and, finally, PT 2 instances are customers of PT 1 instances. In order to design a pattern for each PT, we first need to know location of its customers. As the locations of PTs 1 and 2 customers are not clear prior to placement of PT 3 instances, we start from the last PT where its customers are EUs.

For the last PT, as EUs (i.e., customers of the last PT) are assumed to be located anywhere in the network in the phase 1, we propose a pattern to cover anywhere in the network with the minimum number of needed last PT instances. To this end, we propose a nine same-size zone pattern for the last PT to find the central zone in the area. As it is a symmetric pattern, deploying the last PT instance in a server in central zone minimizes the variance of the distance (i.e., delay) between the last PT and the EUs located in any place of the nine-zone pattern. Therefore, a fairness in QoS is provided for the PT instance customers. Here, the maximum distance between each instance of PT 3 and its customers, X_3, is $2\sqrt{2}$.

Likewise, pattern 2 is created based on merging four copies of pattern 3 together. An instance of PT 2 is placed in a zone in the center of the merged squared-shape area (i.e., the sticking point of these four copies of pattern 3) to offer a minimized variance of delay to its customers (i.e., instances of PT 1). The maximum distance between each instance of PT 2 and its customers, X_2, is $2.5\sqrt{2}$. We have the same situation for PT 3 to offer a minimized variance of the delay to its customers (i.e., instances of PT 2). The maximum distance between each instance of PT 1 and its customers, X_1, is $4\sqrt{2}$. This procedure can be continued in situation of exiting more PTs. Finally, the maximum chain length can be calculated based on the three defined patterns (i.e., *MaxChainLength*= X_3+ X_2+ X_1=$8.5\sqrt{2}d$).

We can generalize the example as follows: for a PT i of a SC with $PartNum$ PTs, its pattern size is 4 times larger than pattern size of PT i+1 (i.e., its edge is two times longer). Therefore, as edge size of the nine-zone pattern of the last PT is $3 * d$, PCPV sets a general formula to find edge PW_i (in unit of zone with edge d) for PT i pattern using equation (9):

$$PW_i = (3 \times d) \times 2^{PartNum-i} \tag{9}$$

Instance of pattern i is then deployed in the central zone of the pattern. Finally, based on the defined patterns, for a SC with $PartNum$ PTs, its maximum length is calculated in terms of zone size d by 10:

$$MaxChainLength = \sum_{i=1}^{PartNum} X_i = k_{PartNum} \cdot d \tag{10}$$

Where X_i is the maximum possible distance between PT i instances (anywhere in the central zone) and their customers (anywhere in the pattern). As the chain length is in the unit of d, we can also consider it as coefficient of d. Next part describes how d is calculated.

After defining the patterns for PTs of a SC, PCPV identifies the appropriate pattern size for placement to minimize the number of used servers while satisfying the service DT. To identify optimal patterns size, as patterns are designed based on the same-size zones, we identify the optimal zone size. Zone size has an important effect on both service delay and cost. Increasing size of the zone decreases the operational cost, (due to deploying fewer instances and using fewer servers) and increases both communication cost and service delay (due to increasing the chain length). As the operational cost plays greater role on total cost in comparison to the communication cost, PCPV attempts to increase zone size.

However, PCPV cannot increase size of the zone (which leads to increasing chain length) in a way that violate the requested DT. Hence, PCPV increases the zone size (which leads to increasing the chain length) up to the point that still satisfying requested DT (and communication cost). To find the *optimal* zone size, PCPV calculates the maximum permitted

SC length based on the service DT using (11), inspired by [7].

$$dist = \frac{(\phi - \delta) \cdot W}{1000} \tag{11}$$

Where $dist$ is the distance (in mi) between two nodes, W is the propagation speed of the network medium (in $\frac{mi}{sec}$) and ϕ is the end to end delay (in ms). We reserve $(\frac{1}{k})$ of total end to end delay for delay from content to the first VNF where k is the number of VNFs in the chain and we call it δ. Using (11), the maximum chain length ($dist$) is calculated by considering ϕ as the service DT. Finally, putting the maximum permitted SC length calculated from (11) in equation (10), PCPV finds the optimal zone size d_{opt}.

Nevertheless, the low values of service delay (ϕ) may result in a relatively small optimal zone size. Hence, some of the selected zones may have no server to be selected. To overcome this issue, we use the algorithm proposed in [8] for solving the Largest Empty Circle (LEC) problem [9]. It computes the largest empty circle (d_0) such that in any other circle with a greater diameter in the same topology, we will have at least one node. We use this notion to ensure that there will be at least one server inside a zone. The final optimal zone size is therefore determined as the maximum of d_{opt} and d_0.

3) Patterns mapping: By mapping the defined patterns (as described in section III-A2) to the network, a set of zones are determined, $\mathcal{Z}_{Sel}^{tot} = \left\{\mathcal{Z}_{Sel}^{i} / 1 \le i \le PartNum\right\}$. Where, $\mathcal{Z}_{Sel}^{i}$ is a set of selected zones for instances of PT i, $\mathcal{Z}_{Sel}^{i} = \left\{\mathcal{Z}_{j}^{i} / 1 \le j \le |\mathcal{Z}_{Sel}^{i}|\right\}$. $\mathcal{Z}_{j}^{i}$ refers to the selected zone for the jth instance of PT i.

Two special cases may arise. The first one consists of having only one copy of a PT in the network. Thus, the succeeding PT is placed in the location closest to the previous one. In the second case, the pattern size is greater than the network size. Accordingly, some patterns cannot be entirely located in the network. Thus, if a server of a pattern is located outside the network, the pattern is shifted into the network until a server is found.

4) Server Selection: For each selected zone $\mathcal{Z}_{j}^{i}$, a server with the least cost (e.g., most energy efficient) is selected to place the instance j of PT i. In this case, $\mathcal{S}_{Sel}^{i}$ is a set of selected servers for instances of PT i, $\mathcal{S}_{Sel}^{i} = \left\{\mathcal{S}_{j}^{i} / 1 \le j \le |\mathcal{S}_{Sel}^{i}|\right\}$. We consider $\mathcal{S}_{Sel}^{tot} = \left\{\mathcal{S}_{Sel}^{i} / 1 \le i \le PartNum\right\}$ as a set of all selected servers for all PTs. As a result, for each PT i, a set of instances $\mathcal{I}_{dep}^{i} = \left\{I_{j}^{i} / 1 \le j \le |I_{dep}^{i}|\right\}$, are deployed in the network (I_{j}^{i} refers to the j^{th} instance of PT i).

5) Chaining: For each instance (e.g., I_{j}^{i}), its customers in its cover area $\mathcal{A}_{j}^{i}$ are assigned to it to receive content. $\mathcal{A}_{j}^{i}$ is the indicated area for I_{j}^{i} defined by the pattern of PT i. However, EU assignment to last PT will be done in phase 2 (remind that we do not have EUs location information in phase 1).

B. Phase 2: VNF Placement Modification

Phase 2 includes two following steps.

1) Placement Modification: As phase 2 has extra information of EUs location, it first assigns the EUs to the instances of the last PT based on the cover area of last PT instances. Afterwards, phase 2 can calculate load of all instances based on the assigned costumers to them, where $\rho(I_{j}^{i})$ is load of instance I_{j}^{i}. Phase 2 then removes all instances without customers and it finds overloaded instances (i.e., instances with overloaded VNFs) starting from the first instance of the last PT to the last instance of the first PT of the chain. When the load of an instance I_{j}^{i} (i.e., $\rho(I_{j}^{i})$) exceeds its capacity (i.e., $PartCap[i]$), a new instance of PT i has to be deployed.

If there is enough capacity on the server $\mathcal{S}_{j}^{i}$ where instance I_{j}^{i} is deployed, the new instance is deployed on that server as well. Otherwise, we select the next lowest cost server (called $\mathcal{S}_{new}^{i}$) in zone $\mathcal{Z}_{j}^{i}$ to deploy the new instance.

If no server in zone $\mathcal{Z}_{j}^{i}$ with enough capacity is available, we expand our search area to the closest zone of the intersection region of $\mathcal{A}_{j}^{i}$ and $\mathcal{A}_{j'}^{i-1}$ (i.e., $C\mathcal{L}_{j}^{i}$). j' is the index of PT $i-1$ instance which I_{j}^{i} is already assigned to it as its customer.

Deploying the new instance inside $C\mathcal{L}_{j}^{i}$ guarantees that the distance between the new instance of PT i and instance $I_{j'}^{i-1}$ does not violate the maximum possible distance (X_{i-1}) and service DT. However, the new instance is not located in the center of the $\mathcal{A}_{j}^{i}$. As the new instance is closer to instance $I_{j'}^{i-1}$ than to already deployed I_{j}^{i}, SC length becomes shorter. The reduction of SC gives possibility of supporting customers farther than predefined cover area of the new instance in phase 1 and, therefore, can cover whole $\mathcal{A}_{j}^{i}$.

Considering the mentioned example in section III-A2, assume instance I_{1}^{3} is overloaded and there is no server with enough capacity in zone $\mathcal{Z}_{1}^{3}$ where instance I_{1}^{3} has been already deployed. As I_{1}^{3} has been assigned to I_{1}^{2}, PCPV places the new instance of PT 3 in the closest zone to instance I_{1}^{2}. While the distance between instance I_{1}^{3} and instance I_{1}^{2} is $2.5\sqrt{2}d$, the distance between the new deployed instance of PT 3 and I_{1}^{2} is reduced to $1.5\sqrt{2}d$. This reduction allows the new instance of PT 3 to cover its customers located farther than the pre-defined distance of $2\sqrt{2}d$ (up to $3\sqrt{2}d$).

It is noted that, after the placement modification, the initial number of instances for each PT obtained in phase 1 based on the pattern mapping, may be changed.

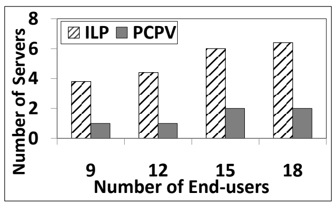
2) Customer Assignment: After modification, if necessary, phase 2 should chain/re-chain a PT instance to a next-level PT instance and/or assign EUs to the instances of the last PT.

IV. PERFORMANCE EVALUATION

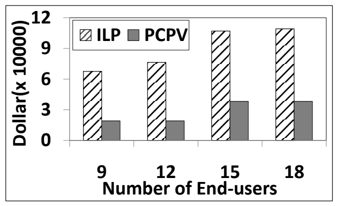
As processing delay and link bandwidths is not considered in the simulations, PCPV is compared with the modified version of ILP implemented in CPLEX. In particular, the constraint (7) is relaxed and in constraint (8), the processing delay of VNFs is ignored and the links delay is traffic independence. We run simulations for 5 different random mesh topologies and report the average of 5 simulations. Three scenarios are designed for small and large environments.

A. Experimental Set up

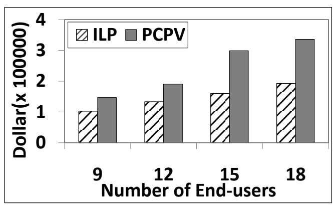
Scenario I compares PCPV to ILP. A CDN provider spans on 300*300 mi^2 geographical area including nine states, each



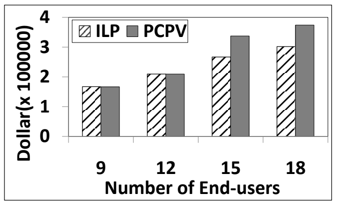
(a) Number of servers



(b) Operational cost



(c) Communication cost



(d) Total cost

Fig. 1. 3 VNFs per chain - 9 states (Scenario I)

state of size 100*100 mi^2. We consider that the CDN provider owns a server in each state and the servers are placed randomly in the states. For this configuration, we consider 9, 12, 15 and 18 EUs number and 3 VNFs per SC with the Service Delay Threshold (D_h^{th}) of 1.5 ms.

Scenario II shows the scalability of PCPV. Twenty various configurations including a different number of states and EUs are defined. We simulate different CDN networks environments with the size of 600*600 mi^2 including 36 states, 700*700 mi^2 including 49 states, 800*800 mi^2 with 64 states, 1200*1200 mi^2 including 144 states and, finally, an environment with a size of 2500*2500 mi^2 including 625 states. Four various numbers of EUs including 50, 100, 150 and 200 are considered. Scenario III shows the impact of the number of VNFs in a SC. We investigate the performance of the proposed algorithm by considering SCs that consist of 3,6 and 9 VNFs in the environment including 625 states. We also assume that the CDN provider owns at least one server in each state of size 100*100 mi^2. The number of servers in each state is chosen randomly between 1 and 5. Bandwidth Cost (B), 10 Dollar/Gbps, EU Load (L_f), 1 Gbps, servers capacity (C_n) 32 vCPU, site license cost (γ) 1000 Dollar, servers Operational cost (δ_n) randomly between 5 and 10 Dollar/vCPU and VNF license cost (α_k), 1000 Dollar/vCPU are the other set up setting in this experiment.

B. Results and Discussions

To evaluate the effectiveness of PCPV, execution time and cost components are calculated.

As mentioned in section III-A, PCPV focuses on reducing operational cost to achieve total cost reduction. However, our method is designed for provisioning the services with the QoS constraint which needs to be satisfied. Placing SCs in such a manner that satisfies the service DT for the EUs leads to keep the communication cost lower than a threshold. Therefore, PCPV is not causing a high communication cost while trying to reduce the operational cost sharply. As a result the performance of PCPV is increased where the impact of operational cost is more than communication cost and total cost is determined by almost the operational cost. This tendency is strongly affected by the cost parameters such as VNF license cost, site lic cost and communication cost per unit of load.

Fig. 1 shows PCPV performance in the worst case where the communication cost is almost 10 times the operational cost. Fig. 1(d) shows the total cost of PCPV and ILP for different numbers of EUs. The results prove that PCPV performance is close to ILP. However, ILP outperforms PCPV in communication cost by improving 30-46% (Fig.1(c)). Whereas PCPV utilizes fewer severs (Fig.1(a)) and improves operational cost by 66-77% in comparison to ILP (Fig. 1(b)) to serve different numbers of EU requests. Therefore, PCPV could keep its performance close to ILP by utilizing fewer servers and having less operational cost. Utilizing fewer servers in PCPV is due to its placement policy. Using partitioning and patterning techniques as well as detecting optimal zone enable PCPV to optimize number of VNFs and find appropriate servers.

As to execution time, in a small size environment with 9 states and different numbers of Eus, PCPV takes around 0.02 second to run whereas ILP needs almost a day. Also, in the scenario II PCPV has less than a minute execution time even for a highest EUs number and states (29.1522 seconds for 625 states and 200 EUs). In addition, by increasing the number of VNFs per chain from 3, to 6 and 9 in scenario III, the execution time remains low (29.4656 and 28.9628 respectively).

V. Conclusion and Future Work

This paper proposed an ILP model and a pattern-based proactive VNF placement algorithm to guarantee the service DT in offering VNF based VASs to EUs. The goal was to place VNFs in a way that it leads to the optimum number of VNFs to reduce the cost while still satisfying the service DT. We showed PCPV is scalable and its performance is close to ILP. As future work, PCPV will be integrated into a scaling algorithm to handle fluctuations in EU workload over time.

References

[1] M. Pathan, R. Buyya, and A. Vakali, *Content Delivery Networks: State of the Art, Insights, and Imperatives*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 3–32.

[2] “Akamai q2 earnings preview: Value-added services will drive growth, but currency headwinds likely played the minor spoilsport,” http://www.trefis.com/, accessed 3 December 2015.

[3] M. Pathan, R. K. Sitaraman, and D. Robinson, *Advanced Content Delivery, Streaming, and Cloud Services*. Wiley Publishing, 2014.

[4] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, “Network function virtualization: State-of-the-art and research challenges,” *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[5] M. Monaci and P. Toth, “A set-covering-based heuristic approach for bin-packing problems,” *INFORMS Journal on Computing*, vol. 18, no. 1, pp. 71–85, 2006.

[6] R. Z. Farahani, M. Hekmatfar, B. Fahimnia, and N. Kazemzadeh, “Survey: Hierarchical facility location problem: Models, classifications, techniques, and applications,” *Comput. Ind. Eng.*, vol. 68, 2014.

[7] K. Wehrle, M. Gnes, and J. Gross, *Modeling and Tools for Network Simulation*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[8] G. T. Toussaint, “Computing largest empty circles with location constraints,” *International Journal of Computer & Information Sciences*, vol. 12, no. 5, pp. 347–358, 1983.

[9] L. P. Chew and R. L. S. Drysdale, “Finding Largest Empty Circles with Location Constraints,” Dartmouth College, Computer Science, Hanover, NH, Tech. Rep. PCS-TR86-130, 1986.