Cost-Efficient VNF Placement and Scheduling in Public Cloud Networks

Tao Gao, Xin Li, Yu Wu, Weixia Zou, Shanguo Huang, *Member, IEEE*, Massimo Tornatore, *Senior Member, IEEE*, and Biswanath Mukherjee, *Fellow, IEEE*

***Abstract*—Following successful adoption of cloud computing, many service providers (SPs) are now using high-performance Virtual Machines (VMs) located in large datacenters owned by public cloud infrastructure providers to deploy their virtual network functions (VNFs). Since using these VMs has a cost depending on utilization time, a complex problem of VNF placement and scheduling (VPS) must be addressed to achieve satisfactory network performance (e.g., latency) while minimizing the cost paid to lease VMs. In this study, a cost-efficient VPS scheme (CE-VPS) is proposed to address the VPS problem in public cloud networks considering dynamic requests of ordered sequences of VNFs. Our CE-VPS scheme goes beyond existing solutions as it models some important practical aspects such as an additional latency incurred by booting a VM and installing a VNF instance. Also, CE-VPS considers that VNFs can be multi-threaded or single-threaded, and that their throughput as a function of allocated computing resources must be modeled differently. CE-VPS is formulated as a mixed inter linear program (MILP) and also as an efficient heuristic algorithm. CE-VPS achieves lower cost and latency than conventional Best-Availability and Cost-Efficient Proactive VNF Placement schemes, and a better trade-off between resource consumption and latency performance than a conventional Low-Latency scheme.**

***Index Terms*—Network function virtualization, cost efficiency, VNF placement and scheduling, public cloud.**

AQ:1 Manuscript received November 2, 2019; revised March 29, 2020; accepted April 26, 2020. This work was supported in part by the National Natural Science Foundation of China (Nos. 61701039, 61331008, 61601054 and 61571058), and the National Science Foundation for Outstanding Youth Scholars of China (No.61622102). The work of Massimo Tornatore and Biswanath Mukherjee was supported by U.S. National Science Foundation Grant No. 1716945. The associate editor coordinating the review of this article and approving it for publication was T. He. *(Corresponding author: Shanguo Huang.)*

Tao Gao is with the State Key Laboratory of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications, Beijing 100876, China, and also with the University of California at Davis, Davis, CA 95616 USA (e-mail: taogao@bupt.edu.cn). AQ:2

Xin Li and Shanguo Huang are with the State Key Laboratory of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: xinli@bupt.edu.cn; shghuang@bupt.edu.cn).

Yu Wu and Biswanath Mukherjee are with the Department of Computer Science, University of California at Davis, Davis, CA 95616 USA (e-mail: yuuwu@ucdavis.edu; bmukherjee@ucdavis.edu).

Weixia Zou is with the Key Laboratory of Universal Wireless Communications, MOE, Beijing University of Posts and Telecommunications, Beijing 100876, China, and also with the School of Information and Telecommunications, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: zwx0218@bupt.edu.cn).

Massimo Tornatore is with the Department of Computer Science, University of California at Davis, Davis, CA 95616 USA, and also with the Department of Electronics and Information, Politecnico di Milano, 20133 Milan, Italy (e-mail: mtornatore@ucdavis.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2020.2992504

I. Introduction

Network function virtualization (NFV) promises to allow service providers (SPs) to reduce operational expenditures (OpEx) and capital expenditures (CapEx) [1]. Traditional network functions such as network address translator (NAT), firewall (FW), and intrusion detection system (IDS) are implemented in hardware middleboxes, which are expensive and complex to maintain and upgrade [2]. However, NFV enables to run virtualized instances of these network functions, i.e., virtual network functions (VNFs) [3], on generic commercial off-the-shelf (COTS) servers, making provisioning of service demands more flexible and efficient.

Service demands are often required to be steered through an ordered set of network functions, which is referred to as a service function chain (SFC) [4], e.g., traffic flow of a given demand may be required to first traverse a FW and then an IDS. Traditionally, traffic flows are routed through the required network functions implemented in hardware middleboxes with manually-configured routing tables. This process is complex, error-prone, and not optimal in terms of networking resource occupation. In contrast, SPs can deploy VNFs according to service demands flexibly and dynamically, and they can even be re-configured during runtime [5].

With development of cloud computing [6], cloud infrastructure providers (CIPs) such as Google cloud platform (GCP) and Amazon AWS offer on-demand computing in the form of virtual machines (VMs) with a pay-as-you-go pricing model [7], [8]. Hence, outsourcing VNFs and SFCs in public clouds provides a good alternative for the SPs, especially for those who might not have geographically-distributed datacenters, e.g., Altiostar, A10 Networks, etc. [9]. Since CIPs usually own several datacenters distributed across large geographical regions, the SP can customize the location of VMs that host VNFs to reduce operational cost and latency. Hence, how to minimize cost to lease computing and networking resources from CIPs is an important operational problem for SPs [10]. This makes the problem of VNF placement and scheduling in public cloud networks for dynamic traffic (VPS-CD) different compared to existing methods (e.g., [11]–[13]) whose objectives are primarily to decrease the latency.

Solving the VPS-CD problem in realistic settings requires one to account for several aspects which are often neglected in previous studies. First, the cost paid by SPs to CIPs is based on amount and duration of consumed cloud service. It is the primary concern for SPs to reduce cost and improve the quality of service (QoS) of demands. For instance, with more allocated resources, some VNFs can achieve higher throughput [14] and hence lower processing latency, but in turn it may lead to a cost increase. Second, service demands arrive in networks dynamically with different requirements (e.g., latency), which should be provisioned in an efficient and flexible manner. For instance, to serve a latency-sensitive demand, VNFs with higher throughput are desired, while for latency-insensitive demands, inexpensive VNFs with lower throughput are enough. Third, a VNF instance is usually installed in a VM or container. Booting a VM and installing a VNF instance will incur some latency [15], which should be taken into account when scheduling the VNF to serve multiple demands. Finally, a VNF can be single-threaded (ST), which can utilize one CPU core at most, or multi-threaded (MT), which can get higher throughput with more CPU cores allocated [16]. For instance, a ST VNF (e.g., Snort ST IDS) should avoid to be installed in a VM with multiple CPU cores, otherwise computing resources will be wasted since all but one CPU cores are idle.

In this study, we focus on the VPS-CD problem and propose a cost-efficient VPS scheme (CE-VPS) to minimize the cost paid by the SP to lease computing and networking resources. Our novel contributions can be summarized as follows:

1) The joint VPS problem is, for the first time to the best of our knowledge, studied for dynamic traffic in a public cloud scenario. Several factors including optimal location determination of VM and VNF, trade-off between computing resource consumption and latency guarantee, and cost-efficient data transmission scheme between different VNF instances, are considered. This study allows SPs to identify the best solution (in terms of VNF placement and scheduling) to deploy VNFs in a public cloud with reasonable cost;
2) We account for service demands with different latency requirements, i.e., fixed, variable, and unlimited. We also consider that, to reduce the latency caused by booting a VM and installing VNF instances, a VNF instance can remain in an idle state momentarily after it finishes previous data processing;
3) We consider VNF attributes and the relationship between VNF throughput and amount of allocated computing resources, which further improves resource utilization efficiency but makes the VPS-CD problem more complex;
4) We formulate the VPS-CD problem as a mixed integer linear program (MILP). Given a set of service demands with different parameters, the MILP aims to minimize the cost with latency constraints. As MILP is computationally prohibitive for large networks with many demands, we also develop an efficient heuristic algorithm.

The rest of this study is organized as follows. In Section II, we review related work. The VPS-CD problem statement is provided in Section III. In Sections IV and V, MILP formulation and heuristic approach to solve the problem are presented, respectively. Illustrative numerical results are discussed in Section VI. Section VII concludes this study.

II. Related Work

NFV promises to reduce operation cost, and improve the network efficiency and flexibility [17]. But it also increases the complexity of resource allocation. In [18], authors divided the NFV resource allocation problem into three parts: 1) VNF chain composition, i.e., how to obtain a specific SFC given a request since the order of VNFs may not be fixed; 2) VNF forwarding graph embedding, i.e., strategy of placing VNFs into physical network nodes; and 3) VNF scheduling, exploring how to schedule the execution of VNFs to reduce the latency of network services. The problem of VNF placement and/or scheduling has been well investigated over the past few years.

Authors in [19] first provided a mathematical formulation for the problem of VNF scheduling by resorting to the flexible job-shop problem. In [20], authors formulated the online VPS problem and proposed several algorithms considering service processing time, revenue, etc. Authors in [21] focused on the joint problem of VNF scheduling and traffic steering to minimize the total latency by proposing a MILP and a genetic algorithm-based method. In addition to minimizing the latency of service demands, other aspects should also be accounted for. An energy-aware VNF placement scheme for SFC in datacenters was proposed in [22] together with a power model in servers and switches. Authors in [23] proposed a MILP and a heuristic algorithm to reduce both end-to-end latency and resource consumption. The VPS problem with objective to minimize the operational cost incurred by deploying VNFs without violating service level agreements (SLAs) is studied in [24]. In [25], authors investigated two different types of cost when multiple chained VNFs share the CPU resource: upscaling cost and context-switching cost.

Many other challenges must be addressed to support deploying VNFs in VMs/containers in practice [26]. A virtualized software middlebox platform named ClickOS was introduced in [15]. While it is light-weight, VM booting latency cannot be avoided. Also, to evaluate the performance of VNFs with different thread attributes, authors in [16] conducted several experiments, which verify that a MT VNF can get higher throughput with more computing resources allocated.

Development of cloud computing has attracted attention for SPs to outsource VNFs to public clouds. Two architectures, APLOMB [27] and CloudNaaS [28], were proposed to outsource enterprise middlebox processing to cloud. In [29], authors studied the influence of NFV on CapEx of cloud-based networks. In [6], a support vector regression-based predictive model was used to minimize latency when deploying VNFs in a multi-cloud network. In [30], performance of deploying VNFs in an industry-relevant cloud platform (e.g., OpenStack) in terms of throughput was evaluated. Authors in [10] studied how to reduce cost when outsourcing the SFC to a multi-cloud network. Also, a cost-efficient service-provisioning scheme with QoS guarantee in a content-delivery network (CDN) was proposed in [31]. These two studies have similar objectives to ours; however, there are several differences: 1) We investigate the joint VPS problem in a dynamic traffic scenario, while both [10] and [31] studied the VNF placement problem for static traffic; 2) Different service demands with diverse latency

AQ:3

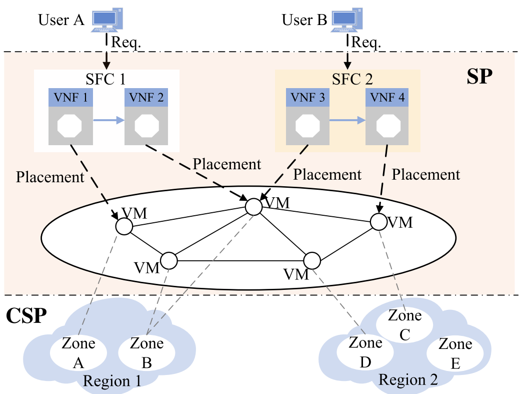


Fig. 1. SFC provisioning in the public cloud network.

requirements are generated in our study, while [31] focused on fixed QoS requirement; 3) We consider VNFs with different thread attributes, and computing resources can achieve different throughputs, making VNF scheduling more complex; and 4) Realistic settings, e.g., VM booting time (VBT), VNF installation time (VIT), etc., are accounted for in our study.

Moreover, the mechanism that a VNF instance can remain in an idle state for a period of time after it finishes any processing task, which is studied in our previous work [32], is also introduced to reduce the latency.

III. VPS-CD Problem Statement

In this section, we first introduce the network model and the metric to evaluate the incurred cost to lease computing and networking resources. Then, the VPS-CD problem is defined. To solve the problem, a conventional low-latency scheme whose objective is to reduce latency is reported and compared with our cost-efficient scheme. Finally, concept of idle state is introduced.

A. Network Model

1) Network Topology and Service Demand: The principle of SPs outsourcing SFCs into the public cloud is illustrated in Fig. 1. A CIP usually has several geographically-distributed datacenters, divided into different regions and zones. For instance, Google has five regions across the United States, in each of which there is one or more zones. Computing resources will be charged at different prices and a transmission fee will be incurred if data is transmitted between different zones. Table I shows pricing scheme of GCP for computing and networking resources in different regions. CIPs offer a pay-as-you-go pricing model. Thus, SPs can set up a VM where and when it is required. Hence, to provision a service demand of a user (e.g., Users A and B in Fig. 1), which requires a SFC consisting of a specific-ordered set of VNFs, the SP's objective is to place these VNFs into the VMs offered by the CIP, and also schedule these VNFs while minimizing cost and satisfying latency requirement of service demands.

Based on the pricing scheme, network topology is represented as $G(V, U, E)$, where V denotes set of VM-capable nodes, U denotes set of user nodes, and E denotes set of physical links. A service demand r is presented as

TABLE I
PRICING SCHEME OF GCP [7]

CPU Resource Pricing	
Region	Price (USD)
Iowa/Oregon/South Carolina	\$0.033174 / CPU hour
Los Angeles	\$0.03797 / CPU hour
Northern Virginia	\$0.037364 / CPU hour
General Network Pricing	
Traffic Type	Price (USD)
Egress between zones	\$0.01 (per GB)
Egress to the same zone	No charge

$r = < \mathbf{s_r}, a_r, d_r, l_r, \overline{s_r}, \overline{d_r} >$, where $\mathbf{s_r}$ is required SFC, a_r is arrival time, d_r is size of data to be processed in GB, l_r is latency requirement, $\overline{s_r}$ is source, and $\overline{d_r}$ is destination. We consider three types of latency requirements:

1) fixed, $l_r = l_r^f$, i.e., demand should be provisioned within deadline l_r^f, which means it is latency-sensitive, e.g., real-time gaming [33];
2) variable, $l_r = [l_r^{req}, l_r^{max}]$, i.e., it is desirable to provision demand within l_r^{req}, but it is acceptable to finish within l_r^{max}, e.g., video streaming [34];
3) unlimited, $l_r \rightarrow +\infty$, i.e., service demand is insensitive to latency, e.g., FTP service [35].

Assume SFC $\mathbf{s_r}$ consists of an ordered set of VNFs denoted as $F_{\mathbf{s_r}} = (f_{\mathbf{s_r},1}, f_{\mathbf{s_r},2}, \ldots, f_{\mathbf{s_r},k})$, where k is length of SFC, i.e., $k = |F_{\mathbf{s_r}}|$. To process the data of a demand, an instance of the required VNF must be installed into a VM with a certain amount of computing resources allocated, which are represented in number of CPU cores for simplicity. A VM can host multiple VNFs, and it will be shut down after all VNFs finish processing user data. The basic throughput of a VNF is P_f Gbps. If a MT VNF is installed in a VM with multiple allocated CPU cores, it can achieve a higher throughput while a ST VNF always has a basic throughput [16]. To simplify the problem, we assume the throughput of a MT VNF is linearly proportional to the amount of CPU cores allocated, i.e., if c CPU cores are allocated for the VM hosting the instance of MT VNF f, the throughput is $c \times P_f$ Gbps.

2) Cost Evaluation: Cost incurred by an SP depends on three components: 1) number of VMs set up; 2) duration a VM keeps running and amount of CPU cores allocated to it; and 3) amount of data transferred between different zones.

In general, our cost model is based on the usage of two types of resources, i.e., computing and networking resources. Note that, if relevant, other kinds of resources, e.g., storage and memory, could be added to our model without impacting the overall proposed scheme. Furthermore, if we consider that some CIPs might charge for the used link bandwidth (e.g., AWS), the cost model can be freely modified to include an additional item.

To quantitatively evaluate the total cost, Eq. (1) is introduced, where M is set of used VMs, c_m is number of CPU cores allocated for VM m, the sum in the brackets is runtime of VM m, P_m^{CPU} is price in dollars per CPU core per time unit, and d (resp. P^{net}) is total size (resp. transmission fee) of data in GB transmitted between different zones. Runtime of VM m is calculated by summing up VBT B_m, runtime of all installed VNF instances (whose set is denoted by $\overline{F_m}$), and time consumed to shut down VM D_m. Furthermore, runtime of VNF instance f is $W_f = t_f^{ins} + \sum_r t_{f,r}^{prs} + t^{idle}$, where t_f^{ins} denotes VIT of VNF instance f, $t_{f,r}^{prs}$ denotes duration that VNF instance f processes data of demand r, and t^{idle} denotes duration of idle state.

$$cost_{total} = \sum_{m \in M} c_m \times \left(B_m + \sum_{f \in \overline{F_m}} W_f + D_m \right) \times P_m^{CPU} + d \times P^{net} \quad (1)$$

B. Low-Latency Scheme vs. CE-VPS Scheme

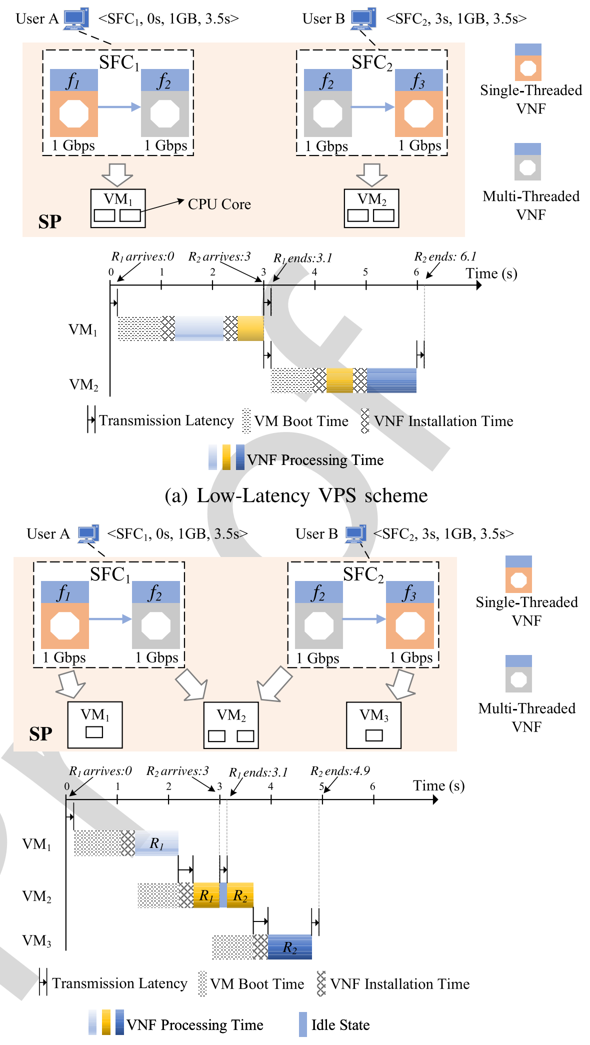
VPS-CD Problem Definition: Given network topology of public clouds with pricing scheme, the objective of placing and scheduling VNFs is to minimize cost incurred by the SP to lease computing and networking resources; also, latency requirements of service demands, which arrive dynamically, should be satisfied.

To solve the VPS-CD problem, we propose a CE-VPS scheme and compare it with a conventional low-latency scheme (C-VPS) whose objective is to minimize latency [22].

1) Comparison of Schemes: C-VPS scheme is shown in Fig. 2(a). There are two service demands R_1 and R_2, which have the same size of data to be processed (1GB) and latency requirement (3.5s), but require different SFCs (SFC_1 and SFC_2, respectively), and arrive at different moments (0s and 3s, respectively). SFC_1 (resp. SFC_2) consists of two VNFs: ST f_1 and MT f_2 (resp. MT f_2 and ST f_3). Basic throughput of all VNFs are assumed to be 1Gbps.

In C-VPS, different VNFs requested by a service demand are installed in a single VM to avoid data transmission latency. As shown in Fig. 2(a), a transmission latency of 0.1s is initially incurred (capacity of connection between user node and datacenter in public cloud is assumed to be 10Gbps in this example). To provision service demand R_1, we first set up a VM with two allocated CPU cores, which incurs a VM booting latency (1s in the example) and a VNF installation latency (0.2s). Since f_1 is ST and basic throughput is 1 Gbps, processing latency of f_1 is 1s. After that, instance of f_2 is installed in same VM, whose throughput is doubled with 2 CPU cores, i.e., 2Gbps, and processing latency is 0.5s. Finally, data is transferred from VM_1 to the destination with a transmission latency of 0.1s. In conclusion, total latency of demand R_1 is 3.1s. However, as f_1 is ST, one CPU core of VM_1 is idle, leading to a waste of computing resources. The example is analogous for demand R_2.

However, the proposed CE-VPS scheme can place and schedule VNFs based on their attributes, as shown in Fig. 2(b). For R_1, another VM with two CPU cores allocated is set up for the instance of f_2 to achieve higher throughput. Also, since we can boot VM_2 in advance before the data processed



(a) Low-Latency VPS scheme

(b) CE-VPS scheme

Fig. 2. Comparison of different schemes.

by the instance of f_1 arrives, latency can be reduced. Basic bandwidth of connection from VM_1 to VM_2 is assumed to be 5Gbps (actually, bandwidth can be customized), hence transmission latency incurred is 0.2s. For the instance of f_2 installed in VM_2, it remains in idle state for a period of time. During this period, demand R_2 arrives, and the instance can start to process its data immediately. Hence, total latency of R_2 can be decreased from 3.1s in C-VPS to 1.9s. Assume price of per-CPU core is \$$P$/s, then total costs of C-VPS and CE-VPS can be calculated by $2.9 \times 2 \times 2 \times P = 11.6P$ dollars and $(2.2 \times 2 + 2.3 \times 2) \times P = 9P$ dollars, respectively. Thus, CE-VPS achieves significantly-lower cost (24%) compared to C-VPS.

2) Idle State: In this subsection, we recall the concept of the idle state through an example. Fig. 3(a) shows two service demands R_1 and R_2, requiring the same SFC, that arrive in the network at different instants. To provision R_1, VM_1 is booted and a new instance of VNF f_1 is installed, incurring some latency. In a conventional scheme, after f_1 finishes processing the data of R_1, data will be transmitted to the instance of VNF f_2; meanwhile, the instance of f_1 will be removed to save computing resources. When R_2 arrives, the same procedure

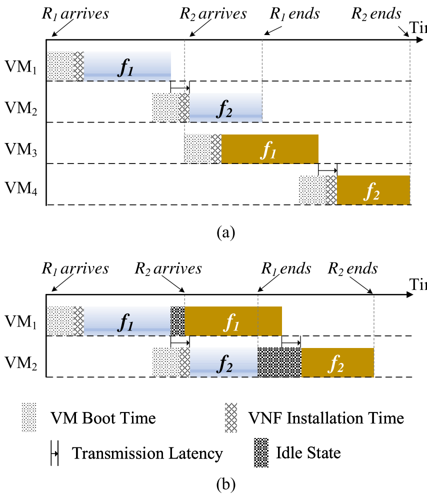


Fig. 3. Different VPS strategies: (a) without idle state and (b) with idle state.

is executed, unnecessarily increasing latency (i.e., intuitively, it would have been preferable to maintain f_1 and f_2 active, avoiding the re-booting).

In our proposed scheme, VNF instances (e.g., VNF f_1 and f_2 as shown in Fig. 3(b)) can remain in idle state after they finish the previous task. Thus, when R_2 arrives, a VNF instance can start working immediately without the need for booting a new VM and re-installing a VNF instance. When the load of service demands is high, idle state can improve the network performance remarkably in terms of additional computing resources. Details about how idle state can affect network performance in terms of latency, resource utilization, etc., can be found in [32]. In this study, a fixed duration of idle state is assumed for VNF instances.

IV. MILP Formulation

In this section, the CE-VPS scheme is formulated as a MILP, which tries to minimize the total cost spent by the SP on computing and networking resources provided by the CIP.

Notations	Description
$G(V, U, E)$	Cloud network topology, where V is set of VM-capable nodes, U is set of user nodes, and E is set of links, $(i, j) \in E$.
$L^{\kappa}_{(i,j)}$	Length of κ^{th} shortest path between nodes i and j, $\kappa \in K$.
H^z_i	1 if node i belongs to zone $z, z \in Z$, and Z is set of zones.
P^{CPU}_z	Price of a CPU core per hour in zone z.
P^{net}	Price of data when traffic transferred between different zones (per GB).
Φ	Speed of light in fiber, 200 km/ms.
Ω	Transmission rate from a user node to a VM-capable datacenter node.
Ψ	A large integer constant.
C	Maximum number of available CPU cores, $c \in [1, C]$.
N	Highest level of egress network capacity that a VM can have, $n \in [1, N]$. Basic network capacity is Θ Gbps.
F, M	Set of VNFs, $f \in F$, and set of VMs, $m \in M$, respectively.
Δ_f	Indicator denoting whether VNF f is MT, i.e., $\Delta_f = 1$, or ST, i.e., $\Delta_f = 0$.
P_f	Basic processing capacity of VNF f.
I, B, D	VIT, VBT, and time consumed to remove a VM, respectively.
S	Set of SFCs, $\mathbf{s} \in S$.
$F_{\mathbf{s}}$	Set of VNFs in SFC $\mathbf{s}$, $F_{\mathbf{s}} \subseteq F$. Let $f_{\mathbf{s},k}$ denote the k^{th} VNF in SFC $\mathbf{s}$, $f_{\mathbf{s},k} \in F_{\mathbf{s}}$.
R	Set of service demands, $r \in R$.

Variables	Description
φ	Float variable denoting total cost.
x_m	Integer variable denoting time when VM m is initialized.
y_m	Integer variable denoting time when VM m is removed.
$l^i_{m,c}$	Integer variable denoting runtime of VM m in node i with c CPU cores.
$q^c_m \in \{0, 1\}$	1 if VM m is allocated with c CPU cores.
$g^i_m \in \{0, 1\}$	1 if VM m is initialized in node i.
$h^{f,m}_r \in \{0, 1\}$	1 if an instance of VNF f requested by demand r, is installed in VM m.
$h^{f,z}_r \in \{0, 1\}$	1 if VNF f is installed in a VM that belongs to zone z.
p^f_r	Integer variable denoting the moment when VNF f requested by demand r starts to process user data.
$p^{f,f'}_{r,r'} \in \{0, 1\}$	1 if VNF f requested by r starts to process data before VNF f' requested by r' does.
w^f_r	Integer variable denoting processing latency of VNF f requested by demand r.
w^k_r	Integer variable denoting transmission latency between VMs hosting k^{th} and $(k+1)^{th}$ VNF instances.
$u^k_r \in \{0, 1\}$	1 if k^{th} and $(k+1)^{th}$ VNF instances are installed in different VMs.
$a^n_m \in \{0, 1\}$	1 if egress network capacity level allocated for VM m is n.
o^k_r	Integer variable denoting propagation latency between VMs hosting k^{th} and $(k+1)^{th}$ VNF instances.
$z^r_k \in \{0, 1\}$	1 if locations of VMs hosting k^{th} and $(k+1)^{th}$ VNF instances belong to different zones.
$e^{r,\kappa}_{(i,j)} \in \{0, 1\}$	1 if the κ^{th} shortest path between nodes i and j is established for demand r.

A. Objective Function

$$Minimize(\varphi) \tag{2}$$

The MILP objective is to minimize the total cost as in Eq. (3).

$$\varphi = \sum_{z \in Z} \sum_{i \in V} \sum_{c \in [1,C]} \sum_{m \in M} c \times l^i_{m,c} \times H^z_i \times P^{CPU}_z + \sum_{r \in R} \sum_{k \in [1, |F_{\mathbf{s_r}}|-1]} d_r \times z^r_k \times P^{net} \tag{3}$$

B. Latency Constraints

$$a_r + l_r \geq p_r^{f_{\mathbf{s_r},k}} + w_r^{f_{\mathbf{s_r},k}} + \frac{d_r}{\Omega} + \frac{\sum\limits_{i \in V} e_{(i,\overline{d_r})}^{r,\kappa} \times L_{(i,\overline{d_r})}^{\kappa}}{\Phi}, \quad \forall r \in R,\ f_{\mathbf{s_r},k} \in F_{\mathbf{s_r}},\ \kappa \in K \quad (4)$$

Eq. (4) ensures the latency requirement of demand r. First two items on right side ensure the last required VNF finishes processing all data before the deadline. Transmission and propagation latency are also considered.

$$p_r^{f_{\mathbf{s_r},1}} \geq a_r + \frac{d_r}{\Omega} + \frac{\sum\limits_{i \in V} e_{(\overline{s_r},i)}^{r,\kappa} \times L_{(\overline{s_r},i)}^{\kappa}}{\Psi}, \quad \forall r \in R,\ f_{\mathbf{s_r},1} \in F_{\mathbf{s_r}},\ \kappa \in K \quad (5)$$

Eq. (5) ensures the first VNF instance of the SFC can start to process data only after the data has been transferred from the user node to the node where the VM is hosting the first VNF through the κ^{th} shortest path, which induces some transmission and propagation latency.

$$p_r^{f_{\mathbf{s_r},k}} + w_r^{f_{\mathbf{s_r},k}} + w_r^k + o_r^k \leq p_r^{f_{\mathbf{s_r},k+1}}, \quad \forall r \in R,\ k \in [1, |F_{\mathbf{s_r}}| - 1],\ f_{\mathbf{s_r},k} \in F_{\mathbf{s_r}} \quad (6)$$

Eq. (6) ensures that processing at VNF $f_{\mathbf{s_r},k+1}$ should not start until the data has been processed by the previous VNF and transferred to the VM hosting VNF $f_{\mathbf{s_r},k+1}$.

$$w_r^f \geq \frac{d_r}{P_f} \times (\Delta_f \times \frac{q_m^c}{c} + 1 - \Delta_f) + \Psi \times (h_r^{f,m} - 1), \quad \forall r \in R,\quad f \in F_{\mathbf{s_r}},\ c \in [1, C],\ m \in M \quad (7)$$

Eq. (7) calculates the VNF processing latency. Specifically, if the VNF is MT, that is $\Delta_f = 1$, the latency is calculated through multiplying basic processing capacity P_f by the number of CPU cores allocated. Otherwise, the latency is calculated only in terms of basic processing capacity.

$$w_r^k \geq \frac{d_r}{\Theta \times n} \times a_m^n + \Psi \times (h_r^{f_{\mathbf{s_r},k},m} + u_r^k - 2), \quad \forall r \in R,\ k \in [1, |F_{\mathbf{s_r}}| - 1],\ f_{\mathbf{s_r},k} \in F_{\mathbf{s_r}},\ n \in [1, N],\ m \in M \quad (8)$$

Eq. (8) calculates that transmission latency between VNFs $f_{\mathbf{s_r},k}$ and $f_{\mathbf{s_r},k+1}$, which applies only when they are deployed in different VMs, i.e., both $h_r^{f_{\mathbf{s_r},k},m}$ and u_r^k equal one. The latency is calculated in terms of the egress network capacity level n allocated to the VM that hosts $f_{\mathbf{s_r},k}$, where a higher level means the latency can be reduced.

$$o_r^k \geq (h_r^{f_{\mathbf{s_r},k},m} + g_m^i + h_r^{f_{\mathbf{s_r},k+1},m'} + g_{m'}^j - 3) \times \frac{L_{(i,j)}^{\kappa}}{\Psi} + (e_{(i,j)}^{r,\kappa} - 1) \times \Psi, \quad \forall r \in R,\ k \in [1, |F_{\mathbf{s_r}}| - 1], \ f_{\mathbf{s_r},k}, f_{\mathbf{s_r},k+1} \in F_{\mathbf{s_r}}, m, m' \in M, i, j \in V,\ \kappa \in K \quad (9)$$

$$1 \geq \sum_{\kappa \in K} e_{(i,j)}^{r,\kappa} \geq h_r^{f,m} + g_m^i + h_r^{f',m'} + g_{m'}^j - 3, \quad \forall r \in R, f,\ f' \in F_{\mathbf{s_r}},\ m, m' \in M,\ i, j \in V \quad (10)$$

Eq. (9) calculates propagation latency of the κ^{th} shortest path between the two VMs hosting two consecutive VNFs in a service chain. This applies only when two VNFs are deployed in different nodes, i.e., when the sum of all variables within the brace equals one. Eq. (10) ensures at most one among K-shortest paths is selected.

C. VNF Placement Constraints

$$\sum_{m \in M} h_r^{f,m} = 1, \quad \forall r \in R,\ f \in F_{\mathbf{s_r}} \quad (11)$$

Eq. (11) ensures that each VNF of the requested SFC should be installed in only one VM.

$$\sum_{r \in R} \sum_{f \in F_{\mathbf{s_r}}} h_r^{f,m} \geq \sum_{i \in V} g_m^i \geq \sum_{r \in R} \sum_{f \in F_{\mathbf{s_r}}} h_r^{f,m} / \Psi, \quad \forall m \in M \quad (12)$$

Eq. (12) ensures that, if a VM is responsible to process user data, it should be mapped into a VM-capable node.

$$\sum_{\kappa \in K} e_{(\overline{s_r},i)}^{\kappa} \geq h_r^{f_{\mathbf{s_r},1},m} + g_m^i - 1, \quad \forall r \in R,\ f_{\mathbf{s_r},1} \in F_{\mathbf{s_r}},\ m \in M,\ i \in V \quad (13)$$

$$\sum_{\kappa \in K} e_{(i,\overline{d_r})}^{\kappa} \geq h_r^{f_{\mathbf{s_r},|F_{\mathbf{s_r}}|},m} + g_m^i - 1, \quad \forall r \in R,\ f_{\mathbf{s_r},|F_{\mathbf{s_r}}|} \in F_{\mathbf{s_r}},\ m \in M,\ i \in V \quad (14)$$

Eqs. (13)-(14) ensure a connection is established from the source to the node where the first VNF is installed, and from the node where the last VNF is installed to the destination.

D. VNF Scheduling Constraints

$$x_m + B + I \leq p_r^f + \Psi \times (1 - h_r^{f,m}), \quad \forall m \in M,\ r \in R,\ f \in F_{\mathbf{s_r}} \quad (15)$$

Eq. (15) ensures the VM should boot before any VNF installed in it begins to process data.

$$y_m \geq D + p_r^f + \Psi \times (h_r^{f,m} - 1) + w_r^f, \quad \forall m \in M,\ r \in R,\ f \in F_{\mathbf{s_r}} \quad (16)$$

Eq. (16) ensures the VM can be shut down after all hosting VNFs have finished their tasks.

$$2 - h_m^{f_{\mathbf{s_r},k}} - h_m^{f_{\mathbf{s_r},k+1}} \geq u_r^k \geq h_m^{f_{\mathbf{s_r},k}} + h_{m'}^{f_{\mathbf{s_r},k+1}} - 1, \quad \forall r \in R,\ f_{\mathbf{s_r},k}, f_{\mathbf{s_r},k+1} \in F_{\mathbf{s_r}},\ m, m' \in M,\ m \neq m' \quad (17)$$

Eq. (17) determines value of u_r^k, which is used to denote whether two consecutive VNFs in a SFC are installed in two different VMs. u_r^k equals 1 iff the VNFs are deployed in two different VMs m and m', where both variables $h_m^{f_{\mathbf{s_r},k}}$ and $h_{m'}^{f_{\mathbf{s_r},k+1}}$ equal 1.

$$z_k^r \geq h_r^{f_{\mathbf{s_r},k},z} + h_r^{f_{\mathbf{s_r},k+1},z'} - 1, \quad \forall r \in R,\ f_{\mathbf{s_r},k},\ f_{\mathbf{s_r},k+1} \in F_{\mathbf{s_r}}, z,\ z' \in Z,\ z \neq z' \quad (18)$$

$$h_r^{f,z} \geq (h_r^{f,m} + g_m^i - 1) \times H_i^z, \quad \forall r \in R,\ f \in F_{\mathbf{s_r}},\ i \in V,\ z \in Z,\ m \in M \quad (19)$$

Eqs. (18)-(19) determine whether VNFs $f_{\mathbf{s_r},k}$ and $f_{\mathbf{s_r},k+1}$ are located in two nodes that belong to different zones.

$$y_m - x_m + (q_m^c + g_m^i - 2) \times \Psi \leq l_{m,c}^i \leq (2 - q_m^c - g_m^i) \times \Psi, \quad \forall m \in M,\ c \in [1, C],\ i \in V \quad (20)$$

Eq. (20) determines runtime of VM m with c CPU cores.

$$1 \geq p_{r,r'}^{f,f'} + p_{r',r}^{f',f} \geq h_r^{f,m} + h_{r'}^{f',m} - 1, \quad \forall r, r' \in R,\ f \in F_{\mathbf{s_r}},\ f' \in F_{\mathbf{s_{r'}}},\ m \in M \quad (21)$$

$$p_r^f + w_r^f + I - p_{r'}^{f'} \leq (1 - p_{r,r'}^{f,f'}),$$
$$\forall r,\ r' \in R,\ f \in F_{\mathbf{s_r}},\ f' \in F_{\mathbf{s_{r'}}} \quad (22)$$

Eqs. (21)-(22) ensure that, if two VNFs f and f' requested by different demands are installed in the same VM, which means both $h_r^{f,m}$ and $h_{r'}^{f',m}$ equal 1, the VM cannot process the two requests at the same time. Hence, the processing order is determined by Eq. (22), where if $p_{r,r'}^{f,f'}$ equals one, meaning that, if VNF f first processes data, VNF f' cannot work until VNF f finishes and an instance is installed.

E. Resource-Allocation Constraints

$$1 \geq \sum_{c \in [1,C]} q_m^c \geq \sum_{r \in R} \sum_{f \in F_{\mathbf{s_r}}} h_r^{f,m}/\Psi, \quad \forall m \in M \quad (23)$$

Eq. (23) calculates the number of CPU cores allocated to a VM.

In the MILP, dominant number of variables is among $l_{m,c}^i$, $h_r^{f,m}$, $h_r^{f,z}$, and $p_{r,r'}^{f,f'}$, which are O($|V| \times |M| \times C$), O($|F| \times |M| \times |R|$), O($|F| \times |Z| \times |R|$), and O($|F|^2 \times |R|^2$), respectively. $|V|$ is size of VM-capable node set, $|M|$ is size of VM set, $|F|$ is size of VNF set, $|R|$ is size of service demand set, and $|Z|$ is number of zones. About constraints, the dominant number is among (9) and (21), which are of complexity O($|F| \times |R| \times |M|^2 \times |V|^2$) and O($|F|^2 \times |R|^2 \times |M|$), respectively.

V. Heuristic Approach

The MILP is computationally prohibitive for large networks. Hence, an efficient heuristic is developed to achieve near-optimal performance for dynamic service demands in large networks. The heuristic for CE-VPS consists of three sub-algorithms, i.e., optimal zone determination (OZD), latency requirement verification (LRV), and service demand provisioning (SDP).

A. OZD Algorithm

OZD is responsible to find the optimal zone to host as many instances of required VNFs as possible, where transmission cost is minimized. We construct a $|Z| \times |F_{\mathbf{s}}|$ matrix M, which is represented as follows. Element m_{f_k,z_j} equals 1 if there is at least one instance of VNF f_k in zone z_j, and 0 otherwise. Next, for each row, we do *AND* operation between any two adjacent elements and sum the results up to get vector V, where the largest value v_j denotes that zone z_j hosts most qualified VNFs so data transmission fee can be decreased.

$$V = AND(M) = \begin{bmatrix} \sum_{k \in [1,|F_{\mathbf{s}}|-1]} m_{f_k,z_1} \,\&\, m_{f_{k+1},z_1} \\ \sum_{k \in [1,|F_{\mathbf{s}}|-1]} m_{f_k,z_2} \,\&\, m_{f_{k+1},z_2} \\ \vdots \\ \sum_{k \in [1,|F_{\mathbf{s}}|-1]} m_{f_k,z_{|Z|}} \,\&\, m_{f_{k+1},z_{|Z|}} \end{bmatrix}$$

$$= \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{|Z|} \end{bmatrix}, \quad M = \begin{bmatrix} m_{f_1,z_1} & \cdots & m_{f_{|F_{\mathbf{s}}|},z_1} \\ \vdots & \ddots & \vdots \\ m_{f_1,z_{|Z|}} & \cdots & m_{f_{|F_{\mathbf{s}}|},z_{|Z|}} \end{bmatrix}$$

The pseudo-code of OZD (**Algorithm 1**) is stated as follows. In **Algorithm 1**, we first determine the VMs that host the instances of required VNFs. In line 2, relevant parameters are initialized, where time indicator T is used to estimate the time at which each VNF in $F_{\mathbf{s}}$ should start to process the data. Candidate sets $I_{f_1}, I_{f_2}, \ldots, I_{f_k}$ are used to store the qualified instances for each VNF. Note that, for the first VNF in SFC $\mathbf{s}$, processing should start after the data is transmitted from user node $\overline{s}$ to the datacenter with a latency of d/C_{in}, where d is data size and C_{in} is ingress network capacity from the user to the public cloud. From line 4 to 6, if the demand must finish before deadline DDL, each VNF instance is checked whether it is available at a certain moment, and time indicator T is updated with the estimated processing time according to the basic throughput of the VNF. Otherwise, all instances are selected as candidates since the demand is insensitive to latency as stated in line 8. In line 9, to find the optimal zone, the matrix-based method presented above is employed, and then the results are returned.

Algorithm 1: OZD Algorithm

Input: Service demand r and its deadline DDL
Output: Optimal zone z, and set of qualified VNFs Q in z

1 Find set of VMs hosting VNFs required by demand r, i.e., $F_{\mathbf{s}} = (f_1, f_2, \ldots, f_{|F_{\mathbf{s}}|})$;
2 Initialize time indicator $T = a + d/C_{in}$ and candidate instance sets, i.e., $I_{f_1}, I_{f_2}, \ldots I_{f_k} = \varnothing$;
3 **if** $DDL \neq +\infty$ **then**
4   **for** *each* $f \in F_{\mathbf{s}}$ **do**
5   Add the instance of VNF f to I_f, if it is available at time T;
6   Update $T = T + d/P_f$;
7 **else**
8   Add all existing instances for each VNF to sets $I_{f_1}, I_{f_2}, \ldots, I_{f_k}$ correspondingly;
9 Employ the matrix-based method to find optimal zone z and VNF set Q, and return;

Complexity: In line 1, complexity of obtaining VMs is O(V_{max}), where V_{max} denotes maximum number of VMs. From line 4 to 6, it requires O($V_{max}F_{max}$) to check the availability of each instance of each required VNF, where F_{max} is maximum number of the VNFs in a SFC. Matrix operation to find the optimal zone in line 9 requires O($V_{max}|Z|$). Taking all steps into consideration, time complexity of **Algorithm 1** is O($V_{max}(F_{max} + |Z|)$).

B. LRV Algorithm

LRV is responsible to check whether user data can be processed by candidate instance set Λ within deadline DDL.

Pseudo-code of LRV (**Algorithm 2**) can be summarized as follows. Time indicator T is initialized in line 1. Next, in line 3, propagation and transmission latency is calculated for the path from source of the demand to the datacenter hosting the first VNF instance. Specifically, Yen's algorithm [36] is employed to calculate K-shortest paths, and the one with

Algorithm 2: LRV Algorithm

Input: Arrival time a of service demand r, candidate instance set Λ, deadline DDL, and size of data to be processed d
Output: **true**, if DDL can be met; **false**, otherwise
1 Initialize time indicator $T = 0$;
2 Calculate K-shortest path from source $\overline{s}$ of r to location of first VNF instance i_{f_1} in Λ and select the one with least latency $lat(\overline{s}, i_{f_1})$;
3 Set $T = a + d/C_{in} + lat(\overline{s}, i_{f_1})$;
4 **for** *each* $k \in |\Lambda|$ **do**
5   Get its throughput $P^*_{f_k}$, available time TS_{f_k}, and egress network capacity C_{f_k};
6   $T = max(TS_{f_k}, T) + d/P^*_{f_k}$;
7   **if** $k \leq |\Lambda| - 1$ **then**
8     $T = T + d/C_{f_k} + lat(i_{f_k}, i_{f_{k+1}})$;
9   **else**
10     $T = T + d/C_{eg} + lat(i_{f_k}, \overline{d})$;
11 **return** $T \leq DDL$? **true**: **false**;

Algorithm 3: SDP Algorithm

Input: Service demand r
1 Initialize set of VNF instances to serve r, i.e., $\Lambda = \varnothing$, deadline for r, i.e., $DDL = l_r$;
2 **if** *r has a variable latency requirement* **then**
3   Set $DDL = l_r^{req}$;
4 Call **Algorithm 1** with $< r, DDL >$ to find optimal zone z and qualified VNF set Q;
5 **for** *each* $f_k \in F_{\mathbf{s}}, k \leq |F_{\mathbf{s}}|$ **do**
6   **if** $f_k \in Q$ **then**
7   Find qualified instance i_f in zone z, $\Lambda = \Lambda \cup i_f$;
8   **else**
9   **if** *$f_{k-1} \in Q$, and meantime, f_{k-1} and f_k have the same attribute* **then**
10   Install an instance i_{f_k} in f_{k-1}' VM;
11   **else**
12   Check all instances of f_k in other zones;
13   **if** *there exist qualified instances* **then**
14   Select the one i_{f_k} in the most inexpensive zone;
15   **else**
16   Set up a new VM in zone z with N CPU cores, calculated by Eq. (24);
17   Install an instance i_{f_k} in the VM;
18     $\Lambda = \Lambda \cup i_{f_k}, Q = Q \cup f_k$;
19 Call **Algorithm 2** with args $< a, \Lambda, DDL, d >$ and get the returned result $flag$;
20 **if** $flag ==$ ***true*** **then**
21   Serve the demand with Λ;
22 **else if** $DDL == l_r^{req}$ **then**
23   Set $DDL = l_r^{max}$, **go to Step** 4;

least latency is selected. From lines 4-10, T is updated after each VNF instance processes the data. Specifically, in line 5, relevant parameters are obtained, where $P^*_{f_k}$ is throughput of VNF instance f_k, TS_{f_k} is the time that f_k can actually start to process the data, and C_{f_k} is egress network capacity of the VM hosting f_k. Egress network capacity is flexible and can be customized. In line 6, T is updated according to the actual start processing time. Then, in lines 7-10, transmission latency and propagation latency are considered. Finally, the result is returned.

Complexity: In line 2, complexity of Yen's algorithm is O($K|V|(|E| + |V| \log |V|)$). Complexity of the for loop from line 4 to 10 is O($F_{max}K|V|(|E| + |V| \log |V|)$). In conclusion, complexity of **Algorithm 2** is O($F_{max}K|V|(|E| + |V| \log |V|)$).

C. SDP Algorithm

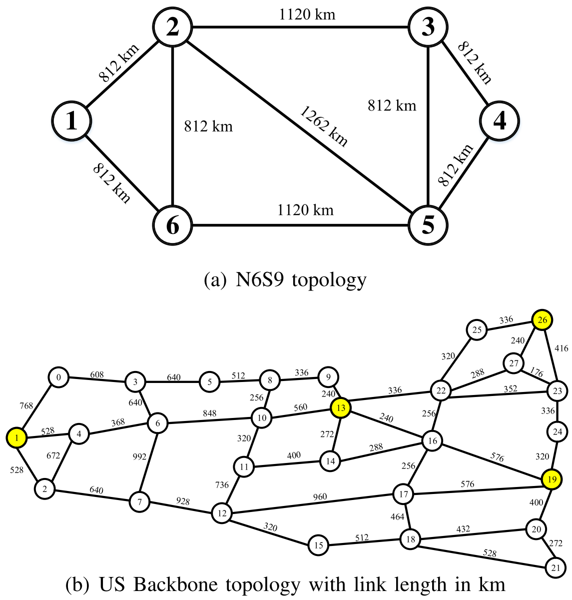
SDP is responsible to serve a single demand that arrives dynamically, and pseudo-code of SDP is reported in **Algorithm 3**. In lines 1-3, deadline DDL is determined based on type of latency requirement of r. **Algorithm 1** is called to find optimal zone z and corresponding VNFs in line 4. Lines 5-18 employ existing or newly-installed VNF instances to serve r. Specifically, in lines 6-7, an existeing instance of the required VNF in zone z is selected. If there is no available instances, the VNF prior to it is checked to see whether they are both ST or MT, in lines 9-10. If they have the same attribute, a new VNF instance is installed in the VM hosting the prior VNF instance. In lines 12-14, if previous procedures fail, other zones are checked to determine whether there are qualified instances. In lines 15-17, a new VM will be booted, for which number of required CPU cores is calculated in Eq. (24).

In Eq. (24), if VNF f is ST, number of required CPU core is one. Otherwise, it is calculated according to deadline DDL and processing latency of other VNFs, i.e., $\sum_{f' \in F_{\mathbf{s}}/f} \frac{d}{P_{f'}}$. Note that processing latencies of other VNFs are estimated in terms of their basic throughput; hence, actual processing latency can be smaller than the estimated value.

$$N = \begin{cases} 1, & f \text{ is ST} \\ \left\lceil d \Big/ \left(\left(DDL - \sum_{f' \in F_{\mathbf{s}}/f} d/P_{f'} \right) \times P_f \right) \right\rceil, & f \text{ is MT} \end{cases} \tag{24}$$

In line 19, **Algorithm 2** is called to verify whether the latency requirement is met. From line 22 to 23, if LRV fails, we check whether the latency requirement can be relaxed.

Complexity: In line 4, **Algorithm 1** is called and its complexity has been analyzed. Complexity of the for loop in lines 5-18 is O($F_{max}V_{max}$). Complexity of **Algorithm 2** has



(a) N6S9 topology

(b) US Backbone topology with link length in km

Fig. 4. Network topologies used in simulation.

also been analyzed. Taking all steps into consideration, complexity of **Algorithm 3** is $O(F_{max}(K|V|(|E|+|V|\log|V|)+V_{max})+V_{max}|Z|)$, which runs in polynomial time.

VI. Performance Evaluation

In this section, we first evaluate the performance of CE-VPS through the MILP in a small-scale network. Then, the heuristic algorithms of CE-VPS and three conventional VPS schemes are compared in large-scale networks.

A. Simulation Setup

The MILP is implemented using ILOG CPLEX v12.5, and heuristic algorithms are coded in Python. All simulations run on a personal computer with Intel i7-7600 2.9 GHz CPU, 16 GB RAM, and Windows 10 operating system.

For MILP, network topology N6S9 shown in Fig. 4(a) is employed, which includes two NFV-capable datacenters belonging to different zones. Prices of CPU core in each zone are \$0.03/hour and \$0.04/hour, respectively. Networking price for data transmission between zones is \$0.01/GB. Data sizes and latency requirements (including fixed and variable) of demands are uniformly distributed in the range [0.1GB, 2GB] and [0.1s, 15s], respectively, according to different types of applications [37]. Further, value of latency requirement is set as infinite for latency-insensitive demands. We assume three VNFs, whose throughputs and attributes are (1Gbps, MT), (2Gbps, MT), and (4Gbps, ST). Also, VBT and VIT are assumed to be 20ms and 10ms, respectively. Basic network capacity Θ is 5 Gbps, and if a VM is allocated with c CPU cores, its egress network capacity is $c \times \Theta$ Gbps [7]. Performance of MILP is compared with CE-VPS heuristic by giving as input the same set of static service demands. Besides, three baseline schemes whose main procedures are as follows.

1) *CPVNF [31]:* For each demand, servers (replaced by VMs in this study for fair comparison) with higher importance rank metric (SIR) are selected for required VNFs. SIR is originally defined according to the remaining computing capacity of a server, bandwidth capacity of links, and whether VNF instances preexist. Since in our study we are considering a public cloud, and computing resource and bandwidth in public cloud can be regarded as unlimited (e.g., the link bandwidth between datacenters can reach over 1 Pbps in Google datacenter network [38]), we modify SIR definition by using the available time and number of CPU cores of a VM instead of remaining computing and bandwidth capacity.
2) *Best-Availability [20]:* For each required VNF of a demand, the scheme attempts to place it into a VM whose current demand queue has the earliest finish time (i.e., best availability).
3) *Low-Latency [22]:* This scheme sets up a new VM to host all VNF instances for each demand. Number of CPU cores allocated to the VM is calculated according to Eq. (24).

The heuristic approach is conducted on US Backbone topology [39], as shown in Fig. 4(b), and there are four datacenters belonging to four different zones. Prices of CPU core in datacenters 1, 13, 19, and 26 are \$0.034/h, \$0.038/h, \$0.04/h, and \$0.035/h, respectively, based on the pricing scheme of GCP [7] as stated before. Traffic arrives dynamically according to a Poisson distribution with λ demands per second. Four different SFCs and six optional VNFs, i.e., NAT, FW, traffic monitor (TM), WAN optimization controller (WOC), intrusion detection and prevention system (IDPS), and video optimization controller (VOC), are considered [25], [40]. Four SFCs are: Web Service (NAT-FW-TM-WOC-IDPS), VoIP (NAT-FW-TM-FW-NAT), Video Streaming (NAT-FW-TM-VOC-IDPS), and Online Gaming (NAT-FW-VOC-WOC-IDPS). We assume throughputs and attributes of optional VNFs are (2Gbps, ST), (1Gbps, MT), (1Gbps, ST), (2Gbps, MT), (4Gbps, ST), and (4Gbps, MT) [40], [41], respectively. The duration of idle state is to 2 seconds according to our previous work. Other parameters are the same as that in the MILP. To obtain good statistical confidence of results, the simulation is run 20 times for each traffic load and we take the average. In each simulation run, 10,000 demands are generated.

B. Performance Comparison: MILP and Heuristics

Fig. 5 shows the cost of different schemes, which is normalized to the largest value achieved by Best-Availability. We observe that MILP can reduce the cost by over 10%, 11%, and 14% on average compared with Low-Latency, CPVNF, and Best-Availability Algorithms, respectively. Moreover, CE-VPS achieves close-to-optimal results, where average gap is 4.7%. Best-Availability has the worst performance, as it prefers to select instances with earliest finish time, even it is in a different zone incurring data transmission cost.

Table II compares the running time of different schemes. We find that, with increasing number of demands, time consumed by MILP increases significantly. It spends over 5 hours on 10 demands, which becomes impractical to be employed. However, all heuristic approaches obtain results with around 100ms.

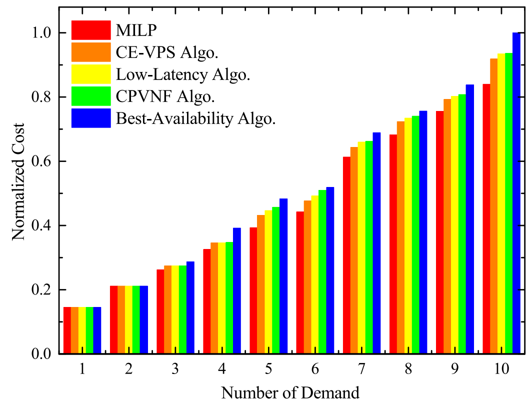


Fig. 5. Normalized cost.

TABLE II
AVG. RUNNING TIME OF DIFFERENT SCHEMES (s)

Number of demands	2	4	6	8	10	12
MILP	5	65	431	4412	20431	-
CE-VPS	0.099	0.102	0.112	0.119	0.119	0.117
Low-Latency	0.083	0.083	0.083	0.086	0.088	0.091
CPVNF	0.100	0.098	0.109	0.113	0.112	0.111
Best-Availability	0.096	0.109	0.110	0.121	0.978	0.112

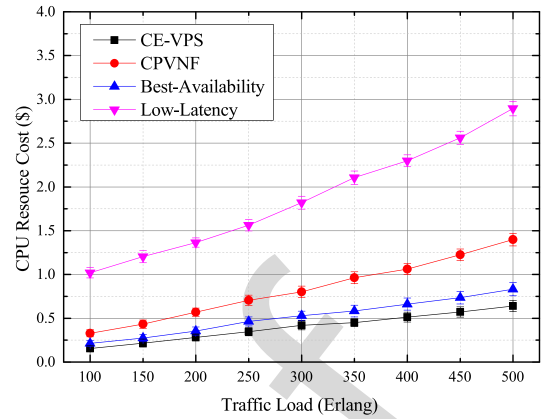
C. Performance Comparison of Different Heuristics

The performance of our proposed CE-VPS heuristic and three baseline algorithms are evaluated according to CPU resource cost, data transmission cost, average latency, and average number of used VMs per service demand. These results are plotted with a confidence level of 95%.

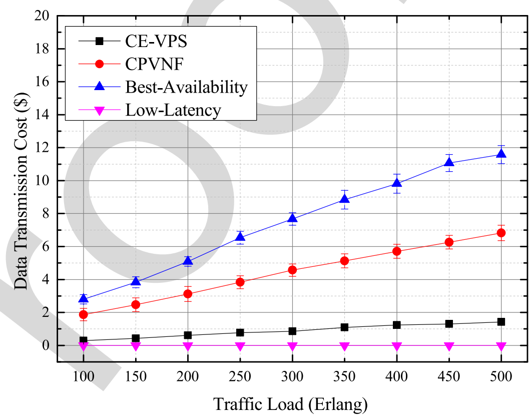
In Fig. 6(a), results show that CPU resource cost increases almost linearly with traffic load for all schemes. Compared to Best-Availability, CPVNF, and Low-Latency schemes, CE-VPS can reduce CPU resource cost by about 23%, 48%, and 78%, respectively, at traffic load of 500 Erlang. The benefits come from the fact that, in CE-VPS, an optimal zone with low price of CPU resource is found to serve the demand. Moreover, in Low-Latency, a VM is established for each demand to host all required VNFs, achieving a highest cost of CPU resources.

Data transmission cost is compared for different schemes in Fig. 6(b). Note that data transmission costs of CE-VPS, CPVNF, and Best-Availability schemes are much higher than costs of CPU resources. However, CE-VPS achieves much lower transmission cost than CPVNF and Best-Availability schemes, and the reduction can reach as high as 76% and 88%, respectively, at traffic load of 500 Erlang. For Low-Latency, there is no transmission cost incurred, but total cost of CE-VPS is still lower than that of Low-Latency.

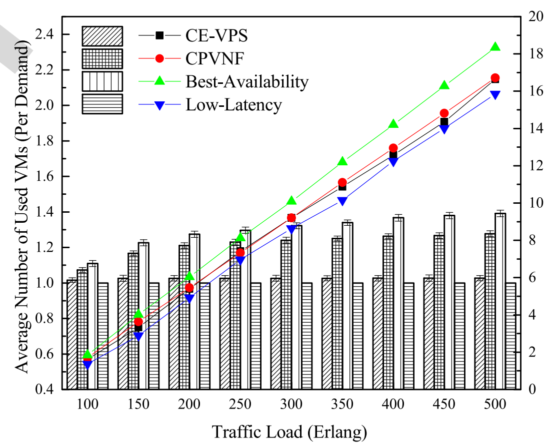
Next, we evaluate the performance in terms of average number of used VMs per service demand for different schemes in Fig. 6(c). In Low-Latency, one VM is set up for each demand, hence the value always equals to one. CE-VPS also achieves a low VM usage, meaning that the frequency at which VMs hosting required VNF instances are reused by multiple demands is much higher than in CPVNF and Best-Availability schemes, which contributes to decreasing the cost of booting new VMs and installing new VNF instances. The frequent reuse benefits from the fact that: 1) scheduling of VNFs can be conducted more efficiently when VNF attributes are considered; and 2) idle state promotes the reuse of VNF instances among multiple demands.



(a) CPU resource cost



(b) Data transmission cost



(c) Average number of used VMs and average latency of different schemes

Fig. 6. Simulation results of different schemes.

Average latencies of different schemes are also compared, where Low-Latency achieves the best performance as the transmission latency between different VMs is avoided. However, latency reduction between Low-Latency and CE-VPS

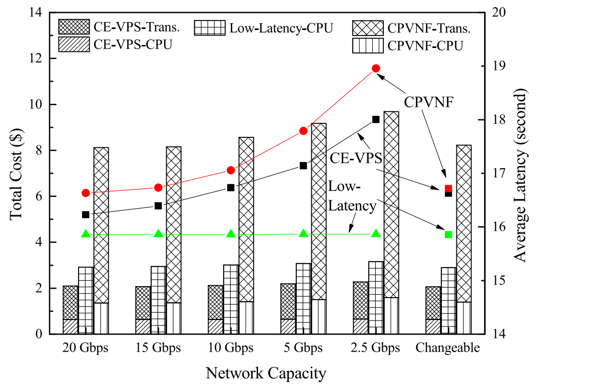


Fig. 7. Total cost (bars) and average latency (curves) for different network capacities.

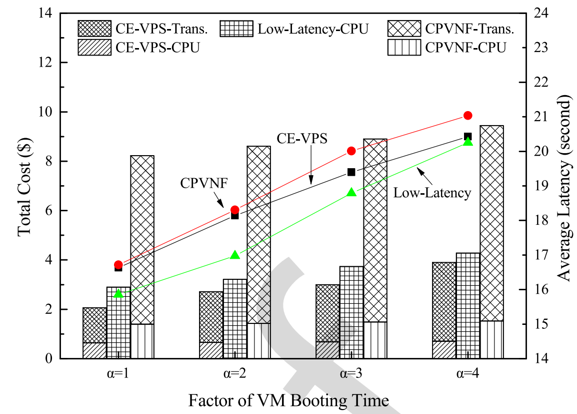
is only about 3% on average, since for CE-VPS, latency requirement will be checked before the demand is finally served. Even compared with CPVNF and Best-Availability, Low-Latency scheme only reduces latency by about 4%, at traffic load of 500 Erlang. With increasing traffic load, average latency of each scheme increases almost linearly. This is because, as average data size of service demands increases, a proportional increment of both processing and transmission latencies is incurred.

D. Performance Comparison for Different Network Capacities

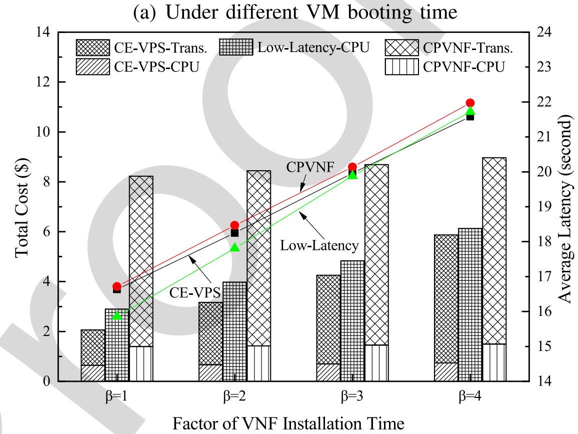
Higher network capacity may reduce the transmission latency and improve reuse of VNFs by multiple demands. Hence, we evaluate performance of different schemes for different network capacities in terms of cost and latency in Fig. 7. In GCP, with more CPU cores, VM can have a higher egress network capacity (see Section VI-A), and such scheme is denoted as "Changeable" in results. Other fixed network capacities, i.e., 20, 15, 10, 5, and 2.5 Gbps, are also considered.

From the results, we find that, with higher network capacity, total cost can be reduced for CE-VPS and CPVNF. Specifically, for CPVNF, CPU resource cost can be reduced by about 13% when network capacity increases from 2.5 Gbps to 5 Gbps, while for CE-VPS, reduction is about 6%. Moreover, cost decreases much slowly when network capacity increases from 10 Gbps to 20 Gbps, which implies that a network capacity of 10 Gbps is enough to guarantee quality of service. Note that CPVNF and CE-VPS with changeable network capacity achieve comparable (or even better) performance compared to that with fixed network capacity of 15 Gbps. This indicates the importance of adjusting network capacity flexibly on reducing transmission cost.

With respect to average latency, we find that Low-Latency remains on the same level with different network capacities. But latency can be reduced by about 11% for CPVNF and by about 6% for CE-VPS when network capacity increases from 2.5 Gbps to 5 Gbps. Performance improvement becomes unremarkable when network capacity is greater than 10 Gbps. The phenomenon indicates that it becomes a bottleneck when network capacity is very small, where service demand can suffer a similar magnitude of transmission latency to VNF processing latency. In this case, performance of both latency and CPU resource cost will deteriorate significantly.



(a) Under different VM booting time



(b) Under different VNF installation time

Fig. 8. Total cost (bars) and average latency (curves) of different schemes.

E. Performance Evaluation Under Different VM Booting Time and VNF Installation Time

To evaluate the effect of VBT and VIT on performance in terms of cost and latency, we run simulation under different parameters. Factors α and β lead to different times of initial VBT and VIT, respectively, e.g., $\alpha = 2$ represents VBT is 40ms (initial time is 20ms). The results are shown in Figs. 8(a) and 8(b).

We find that all schemes consume more CPU resources for increasing VBT, and the increment is more significant for Low-Latency. For CE-VPS, data transmission cost increases more remarkably for a longer booting time. This is because, to provision a demand with a strict latency requirement, an active VNF instance (even in a different zone) is preferred to be selected as booting a new VM will induce a significant latency. But, in CPVNF, available time of VMs and computing resource consumption are both considered, leading to a slight increase of data transmission cost, which is similar to CE-VPS.

It can also be found from Fig. 8(a) that service demands suffer a longer average latency when VBT increases, and performance of Low-Latency is significantly affected by VBT. Specifically, when $\alpha = 4$, CE-VPS achieves average latency close to Low-Latency.

Effect of VIT on performance for different schemes is shown in Fig. 8(b). We find that VIT has a more notable influence on performance than VBT. For CE-VPS, data transmission cost rises a lot when VIT becomes longer. Specifically, when $\beta = 4$, total cost of CE-VPS becomes very close to Low-Latency. In CPVNF, CPU resource cost almost remains the same while data transmission cost increases slightly with VIT being longer, since it tries to achieve a trade-off between CPU resource consumption and latency performance. Moreover, the effect of VIT on CPU resource cost is more vital for Low-Latency.

Average latency for the three schemes increases when β factor becomes larger, because for demands that are latency-insensitive, required VNFs are more likely to be executed in a VM with fewer CPU resource allocated. It should be noted that the low-latency advantage of Low-Latency over CE-VPS disappears when β equals 4.

Thus, we conclude that both VBT and VIT have significant impact on the performance in terms of CPU resource cost, data transmission cost, and average latency. Specifically, Low-Latency, for each service demand, sets up a new VM and initializes required VNF instances, and this affects negatively its performance, both in terms of cost and latency (VIT has more impact than VBT). CE-VPS, instead, minimizes the costs of CPU resource and data transmission by attempting to re-use existing VNF instances and to avoid data transmission among different zones. As CE-VPS also ensures that latency requirement is satisfied, a superior trade-off between latency performance and cost can be achieved.

As a whole, we show that re-using existing VM/VNF instances allows to more effectively satisfy latency requirements, especially for latency-sensitive applications, e.g., the emerging VR gaming. In turn, this indicates that it is desirable to have technologies for rapid VNF booting and to deploy VNFs in public clouds with short VBT.

VII. Conclusion

Cloud computing allows SPs to deploy VNFs into high-performance VMs in public cloud datacenters operated by CIPs. When deploying VNFs in cloud, the SP aims to minimize the cost paid to lease computing and networking resources, while satisfying diverse latency requirements of different service demands. The optimization problem addressed in this study, namely the "VNF placement and scheduling in public cloud networks (VPS-CD)", is different from other conventional versions of the VPS problem. In VPS-CD, we incorporate the impact of several realistic factors which are typically neglected in existing VPS solutions, e.g., VNF threading attributes, VM booting time, and VNF installation time. A solution to the VPS-CD problem has not been investigated before until now. In this study, a cost-efficient VPS-CD scheme is proposed, and to formulate the VPS-CD problem, a MILP and an efficient heuristic are designed for small-scale and large-scale networks, respectively. Our results confirm the importance of developing VNFs with short installation time and of using algorithms (such as VPS-CD) which promote the reutilization of existing VM/VNF instances. Compared to two baseline schemes, Best-Availability and Cost-Efficient Proactive VNF Placement (CPVNF), both total cost and latency can be reduced by CE-VPS. Also, a better trade-off between resource consumption and latency performance is achieved by CE-VPS when compared to a conventional Low-Latency scheme.

References

[1] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Comput. Commun.*, vol. 102, pp. 1–16, Apr. 2017.

[2] C. Donley, J. Berg, and M. Kloberdans, "Network function virtualization (NFV)," U.S. Patent 14 788 684, Jan. 7, 2016,

[3] G. ETSI, "Network functions virtualisation (NFV); Use cases," *V1*, vol. 1, p. 10, Dec. 2013.

[4] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-Art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[5] R. Yu, G. Xue, V. T. Kilari, and X. Zhang, "Network function virtualization in the multi-tenant cloud," *IEEE Netw.*, vol. 29, no. 3, pp. 42–47, May 2015.

[6] L. Gupta, M. Samaka, R. Jain, A. Erbad, D. Bhamare, and C. Metz, "COLAP: A predictive framework for service function chain placement in a multi-cloud environment," in *Proc. IEEE 7th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2017, pp. 1–9.

[7] *Google Cloud Platform*. [Online]. Available: https://cloud.google.com/44 AQ:4

[8] *Amazon EC2*. [Online]. Available: https://aws.amazon.com/ec2/

[9] (Jul. 2018). *2018 NFV Report Series Part 3: State of the VNF Ecosystem*.https://noteya.com/telco-systems-sdxcentral-2018-nfv-report-series-part%-3/

[10] H. Chen *et al.*, "MOSC: A method to assign the outsourcing of service function chain across multiple clouds," *Comput. Netw.*, vol. 133, pp. 166–182, Mar. 2018.

[11] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2014, pp. 7–13.

[12] T. Lin, Z. Zhou, M. Tornatore, and B. Mukherjee, "Demand-aware network function placement," *J. Lightw. Technol.*, vol. 34, no. 11, pp. 2590–2600, Jun. 1, 2016.

[13] R. Mijumbi, "Placement and scheduling of functions in network function virtualization," 2015, *arXiv:1512.00217*. [Online]. Available: http://arxiv.org/abs/1512.00217

[14] A. Gupta, M. Farhan Habib, U. Mandal, P. Chowdhury, M. Tornatore, and B. Mukherjee, "On service-chaining strategies using virtual network functions in operator networks," *Comput. Netw.*, vol. 133, pp. 1–16, Mar. 2018.

[15] J. Martins *et al.*, "ClickOS and the art of network function virtualization," in *Proc. 11th USENIX Conf. Networked Syst. Design Implement.*, 2014, pp. 459–473.

[16] A. Sheoran, X. Bu, L. Cao, P. Sharma, and S. Fahmy, "An empirical case for container-driven fine-grained VNF resource flexing," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2016, pp. 121–127.

[17] G. ETSI, "Network functions virtualisation (NFV): Architectural framework," *ETsI Gs NFV*, vol. 2, no. 2, p. 1, 2013.

[18] J. Gil Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[19] J. F. Riera, E. Escalona, J. Batalle, E. Grasa, and J. A. Garcia-Espin, "Virtual network function scheduling: Concept and challenges," in *Proc. Int. Conf. Smart Commun. Netw. Technol. (SaCoNeT)*, Jun. 2014, pp. 1–5.

[20] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Proc. 1st IEEE Conf. Netw. Softw. (NetSoft)*, Apr. 2015, pp. 1–9.

[21] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3746–3758, Sep. 2016.

[22] K. Yang, H. Zhang, and P. Hong, "Energy-aware service function placement for service function chaining in data centers," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[23] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 98–106.
[24] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.
[25] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2015, pp. 191–197.
[26] J. Anderson, H. Hu, U. Agarwal, C. Lowery, H. Li, and A. Apon, "Performance considerations of network functions virtualization using containers," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2016, pp. 1–7.
[27] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 13–24, Sep. 2012.
[28] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "CloudNaaS: A cloud networking platform for enterprise applications," in *Proc. 2nd ACM Symp. Cloud Comput. (SOCC)*, 2011, pp. 1–8.
[29] M. D. Ananth and R. Sharma, "Cost and performance analysis of network function virtualization based cloud systems," in *Proc. IEEE 7th Int. Advance Comput. Conf. (IACC)*, Jan. 2017, pp. 70–74.
[30] P. Bellavista *et al.*, "Virtual network function embedding in real cloud environments," *Comput. Netw.*, vol. 93, pp. 506–517, Dec. 2015.
[31] M. Dieye *et al.*, "CPVNF: Cost-efficient proactive VNF placement and chaining for value-added services in content delivery networks," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 2, pp. 774–786, Jun. 2018.
[32] T. Gao *et al.*, "Demand-adaptive VNF placement and scheduling with low latency in optical datacenter networks," in *Proc. 17th Int. Conf. Opt. Commun. Netw. (ICOCN)*, Feb. 2019, Art. no. 110480.
[33] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei, "Measuring the latency of cloud gaming systems," in *Proc. 19th ACM Int. Conf. Multimedia*, 2011, pp. 1269–1272.
[34] E. Steinbach, N. Farber, and B. Girod, "Adaptive playout for low latency video streaming," in *Proc. Int. Conf. Image Process.*, 2001, pp. 962–965.
[35] J. D. Touch and D. J. Farber, "An experiment in latency reduction," in *Proc. Conf. Comput. Commun.*, Jun. 1994, pp. 175–181.
[36] J. Y. Yen, "An algorithm for finding shortest routes from all source nodes to a given destination in general networks," *Quart. Appl. Math.*, vol. 27, no. 4, pp. 526–530, Jan. 1970.
[37] M. Kuzlu, M. Pipattanasomporn, and S. Rahman, "Communication network requirements for major smart grid applications in HAN, NAN and WAN," *Comput. Netw.*, vol. 67, pp. 74–88, Jul. 2014.
[38] *Google's Data Centers Grow So Fast it Has to Build its Own Networks*. [Online]. Available: https://www.computerworld.com/article/2937831/googles-data-centers-grow%-so-fast-it-has-to-build-its-own-networks.html
[39] T. Gao, W. Zou, X. Li, B. Guo, S. Huang, and B. Mukherjee, "Distributed sub-light-tree based multicast provisioning with shared protection in elastic optical datacenter networks," *Opt. Switching Netw.*, vol. 31, pp. 39–51, Jan. 2019.
[40] A. Gupta, B. Jaumard, M. Tornatore, and B. Mukherjee, "A scalable approach for service chain mapping with multiple SC instances in a wide-area network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 529–541, Mar. 2018.
[41] K. Argyraki *et al.*, "Can software routers scale?" in *Proc. ACM Workshop Program. Routers Extensible Services Tomorrow*, 2008, pp. 21–26.

Tao Gao received the B.S. degree in communication engineering from the Dalian University of Technology (DLUT) in 2010, and the Ph.D. degree from the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications (BUPT), in 2019. He was a Visiting Ph.D. Student with the Network Laboratory, University of California at Davis, Davis, CA, USA, from 2017 to 2018. His research interests include the survivability of optical networks and the virtualization of network functions.

Xin Li was born in Shandong, China, in 1986. He received the B.S. degree in communication engineering from the Shandong University of Science and Technology in 2008 and the Ph.D. degree in communication and information system from the Beijing University of Posts and Telecommunications (BUPT) in 2014. He is currently a Lecturer at BUPT. His research focuses on software-defined optical networking, elastic optical networks, datacenter networking, and optical network survivability.

Yu Wu received the B.E. degree from the Department of Information and Communication Engineering, Zhejiang University, China, in 2014, and the Ph.D. degree from the Department of Computer Science, University of California at Davis, Davis, CA, USA, in 2018. He currently works with Google Cloud focusing on software-defined network (SDN) research and development. During his Ph.D. study, he has authored 14 peer-reviewed conference papers and journal articles. His research interests include cloud and optical network resource management/optimization, green energy communication, and 5G emerging technologies. He was a recipient of the Best Paper Award granted by the IEEE Transmission, Access, and Optical Systems (TAOS) Technical Committee in IEEE GLOBECOM 2017.

Weixia Zou received the bachelor's degree from Tongji University in 1994, the master's degree from Shandong University in 2002, and the Ph.D. degree from the Beijing University of Posts Telecommunications (BUPT), China, in 2006. She is currently an Associate Professor with BUPT. Her current research focused on new technologies of short-range wireless communications and the electromagnetic compatibility (EMC).

Shanguo Huang (Member, IEEE) received the Ph.D. degree from the Beijing University of Posts Telecommunications (BUPT), Beijing, China, in 2006. He is currently a Professor with the State Key Laboratory of Information Photonics and Optical Communications (IPOC), and the Dean of the School of Sciences, BUPT. He has been actively undertaking several national projects, published three books and more than 150 journals and refereed conferences, and authorized 14 patents. His current research interests include the networks designing, planning, the traffic control and resource allocations, routing algorithms, and performance analysis. He was awarded the Beijing Higher Education Young Elite Teacher, the Beijing Nova Program, and the Program for New Century Excellent Talents in University from the Ministry of Education, from 2011 to 2013. AQ:5

Massimo Tornatore (Senior Member, IEEE) received the Ph.D. degree in information engineering from the Department of Electronics, Information and Bioengineering, Politecnico di Milano, Italy, in 2006. He is currently an Associate Professor with the Department of Electronics, Information and Bioengineering, Politecnico di Milano. He also holds an appointment as an Adjunct Professor with the Department of Computer Science, University of California at Davis, Davis, CA, USA. He has authored over 300 peer-reviewed conference papers and journal articles. His research interests include performance evaluation, optimization and design of communication networks (with an emphasis on the application of optical networking technologies), cloud computing, and energy-efficient networking. He was a co-recipient of 11 best-paper awards. He is a member of the Editorial Board of *Photonic Network Communications* (Springer), *Optical Switching and Networking* (Elsevier), IEEE COMMUNICATION SURVEYS AND TUTORIALS, and IEEE COMMUNICATION LETTERS.

Biswanath (Bis) Mukherjee (Fellow, IEEE) received the B.Tech. degree (Hons.) in electronics from the Indian Institute of Technology, Kharagpur, in 1980, and the Ph.D. degree in electrical engineering from the University of Washington, Seattle, in 1987.

He presented the first proposal/prototype for a network intrusion detection system in 1990; today it is a 10B/year industry. Also, in 1990, he proposed and prototyped the first filtering firewall, which is 8B/year business today. He led the proposal/prototype for the first dynamic bandwidth allocation algorithm in Ethernet-based fiber-to-the-home networks in 2002; today over 100 million units are deployed worldwide, including AT&T's Gigabit Fiber offering. In 2007, he provided the first proposal to integrate optical and wireless networks, forming the basis for the fronthaul of the upcoming 5G industry. He served a five-year term as a Founding Member of the Board of Directors of IPLocks, a Silicon Valley startup company (acquired by Fortinet). He was also a Founding Director of an optical startup, Optella, Inc. (acquired by Cosemi, Inc.) He has served on the Technical Advisory Board of over a dozen startup companies, including Teknovus (acquired by Broadcom). He is the Founder and the President of Ennetix, Inc., a Davis-based, SBIR-funded startup company specializing in AI-powered, application-centric, and network analytics for optimal user experience. He is also a Distinguished Professor of computer science with the University of California at Davis, Davis, CA, USA, where he has been for the past three decades. He was the author of the graduate-level textbook *Optical WDM Networks* (Springer, January 2006). He was the General Co-Chair of the IEEE/OSA Optical Fiber Communications (OFC) Conference in 2011, the TPC Co-Chair of OFC2009, and the TPC Chair of IEEE INFOCOM'96. He has supervised 77 Ph.D. students to completion. He is the Winner of the 2004 Distinguished Graduate Mentoring Award, the 2009 College of Engineering Outstanding Senior Faculty Award, and the 2016 UC International Community Building Award at UC Davis. He was the Co-Winner of the 2018 Charles Kao Award (named after Nobel Laureate and Fiber-Optic Pioneer Charles Kao) for the Best Paper in IEEE/OSA Journal of Optical Communications and Networks. He is also the Co-Winner of 12 other Best Paper Awards, mainly from IEEE conferences and journals; and also from the National Computer Security Conference. He was also the Winner of the IEEE Communications Society's Inaugural (2015) Outstanding Technical Achievement Award for pioneering work on shaping the optical networking area. He is an Editor of *Optical Networks* (Springer). He has served on eight journal editorial boards, most notably IEEE/ACM Transactions on Networking and IEEE Network; and also as a Guest Editor for IEEE Journal on Selected Areas in Communications, IEEE/OSA Journal of Lightwave Technology, Proceedings of the IEEE, and IEEE Communications.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your article. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

Carefully check the page proofs (and coordinate with all authors); additional changes or updates WILL NOT be accepted after the article is published online/print in its final form. Please check author names and affiliations, funding, as well as the overall article for any errors prior to sending in your author proof corrections. Your article has been peer reviewed, accepted as final, and sent in to IEEE. No text changes have been made to the main part of the article as dictated by the editorial level of service for your publication.

AQ:1 = Please confirm or add details for any funding or financial support for the research of this article.

AQ:2 = Please provide the department name for University of California at Davis where the author Tao Gao is currently affiliated with.

AQ:3 = Note that if you require corrections/changes to tables or figures, you must supply the revised files, as these items are not edited for you.

AQ:4 = Please provide the accessed date for Refs. [7], [8], and [38].

AQ:5 = Please provide a better/higher quality image for Shanguo Huang.

Cost-Efficient VNF Placement and Scheduling in Public Cloud Networks

Tao Gao, Xin Li, Yu Wu, Weixia Zou, Shanguo Huang, *Member, IEEE*, Massimo Tornatore, *Senior Member, IEEE*, and Biswanath Mukherjee, *Fellow, IEEE*

***Abstract*— Following successful adoption of cloud computing, many service providers (SPs) are now using high-performance Virtual Machines (VMs) located in large datacenters owned by public cloud infrastructure providers to deploy their virtual network functions (VNFs). Since using these VMs has a cost depending on utilization time, a complex problem of VNF placement and scheduling (VPS) must be addressed to achieve satisfactory network performance (e.g., latency) while minimizing the cost paid to lease VMs. In this study, a cost-efficient VPS scheme (CE-VPS) is proposed to address the VPS problem in public cloud networks considering dynamic requests of ordered sequences of VNFs. Our CE-VPS scheme goes beyond existing solutions as it models some important practical aspects such as an additional latency incurred by booting a VM and installing a VNF instance. Also, CE-VPS considers that VNFs can be multi-threaded or single-threaded, and that their throughput as a function of allocated computing resources must be modeled differently. CE-VPS is formulated as a mixed inter linear program (MILP) and also as an efficient heuristic algorithm. CE-VPS achieves lower cost and latency than conventional Best-Availability and Cost-Efficient Proactive VNF Placement schemes, and a better trade-off between resource consumption and latency performance than a conventional Low-Latency scheme.**

***Index Terms*— Network function virtualization, cost efficiency, VNF placement and scheduling, public cloud.**

AQ:1 Manuscript received November 2, 2019; revised March 29, 2020; accepted April 26, 2020. This work was supported in part by the National Natural Science Foundation of China (Nos. 61701039, 61331008, 61601054 and 61571058), and the National Science Foundation for Outstanding Youth Scholars of China (No.61622102). The work of Massimo Tornatore and Biswanath Mukherjee was supported by U.S. National Science Foundation Grant No. 1716945. The associate editor coordinating the review of this article and approving it for publication was T. He. *(Corresponding author: Shanguo Huang.)*

Tao Gao is with the State Key Laboratory of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications, Beijing 100876, China, and also with the University of California at Davis, Davis, CA 95616 USA (e-mail: taogao@bupt.edu.cn). AQ:2

Xin Li and Shanguo Huang are with the State Key Laboratory of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: xinli@bupt.edu.cn; shghuang@bupt.edu.cn).

Yu Wu and Biswanath Mukherjee are with the Department of Computer Science, University of California at Davis, Davis, CA 95616 USA (e-mail: yuuwu@ucdavis.edu; bmukherjee@ucdavis.edu).

Weixia Zou is with the Key Laboratory of Universal Wireless Communications, MOE, Beijing University of Posts and Telecommunications, Beijing 100876, China, and also with the School of Information and Telecommunications, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: zwx0218@bupt.edu.cn).

Massimo Tornatore is with the Department of Computer Science, University of California at Davis, Davis, CA 95616 USA, and also with the Department of Electronics and Information, Politecnico di Milano, 20133 Milan, Italy (e-mail: mtornatore@ucdavis.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2020.2992504

I. Introduction

NETWORK function virtualization (NFV) promises to allow service providers (SPs) to reduce operational expenditures (OpEx) and capital expenditures (CapEx) [1]. Traditional network functions such as network address translator (NAT), firewall (FW), and intrusion detection system (IDS) are implemented in hardware middleboxes, which are expensive and complex to maintain and upgrade [2]. However, NFV enables to run virtualized instances of these network functions, i.e., virtual network functions (VNFs) [3], on generic commercial off-the-shelf (COTS) servers, making provisioning of service demands more flexible and efficient.

Service demands are often required to be steered through an ordered set of network functions, which is referred to as a service function chain (SFC) [4], e.g., traffic flow of a given demand may be required to first traverse a FW and then an IDS. Traditionally, traffic flows are routed through the required network functions implemented in hardware middleboxes with manually-configured routing tables. This process is complex, error-prone, and not optimal in terms of networking resource occupation. In contrast, SPs can deploy VNFs according to service demands flexibly and dynamically, and they can even be re-configured during runtime [5].

With development of cloud computing [6], cloud infrastructure providers (CIPs) such as Google cloud platform (GCP) and Amazon AWS offer on-demand computing in the form of virtual machines (VMs) with a pay-as-you-go pricing model [7], [8]. Hence, outsourcing VNFs and SFCs in public clouds provides a good alternative for the SPs, especially for those who might not have geographically-distributed datacenters, e.g., Altiostar, A10 Networks, etc. [9]. Since CIPs usually own several datacenters distributed across large geographical regions, the SP can customize the location of VMs that host VNFs to reduce operational cost and latency. Hence, how to minimize cost to lease computing and networking resources from CIPs is an important operational problem for SPs [10]. This makes the problem of VNF placement and scheduling in public cloud networks for dynamic traffic (VPS-CD) different compared to existing methods (e.g., [11]–[13]) whose objectives are primarily to decrease the latency.

Solving the VPS-CD problem in realistic settings requires one to account for several aspects which are often neglected in previous studies. First, the cost paid by SPs to CIPs is based on amount and duration of consumed cloud service. It is the primary concern for SPs to reduce cost and improve the quality of service (QoS) of demands. For instance, with more allocated resources, some VNFs can achieve higher throughput [14] and hence lower processing latency, but in turn it may lead to a cost increase. Second, service demands arrive in networks dynamically with different requirements (e.g., latency), which should be provisioned in an efficient and flexible manner. For instance, to serve a latency-sensitive demand, VNFs with higher throughput are desired, while for latency-insensitive demands, inexpensive VNFs with lower throughput are enough. Third, a VNF instance is usually installed in a VM or container. Booting a VM and installing a VNF instance will incur some latency [15], which should be taken into account when scheduling the VNF to serve multiple demands. Finally, a VNF can be single-threaded (ST), which can utilize one CPU core at most, or multi-threaded (MT), which can get higher throughput with more CPU cores allocated [16]. For instance, a ST VNF (e.g., Snort ST IDS) should avoid to be installed in a VM with multiple CPU cores, otherwise computing resources will be wasted since all but one CPU cores are idle.

In this study, we focus on the VPS-CD problem and propose a cost-efficient VPS scheme (CE-VPS) to minimize the cost paid by the SP to lease computing and networking resources. Our novel contributions can be summarized as follows:

1) The joint VPS problem is, for the first time to the best of our knowledge, studied for dynamic traffic in a public cloud scenario. Several factors including optimal location determination of VM and VNF, trade-off between computing resource consumption and latency guarantee, and cost-efficient data transmission scheme between different VNF instances, are considered. This study allows SPs to identify the best solution (in terms of VNF placement and scheduling) to deploy VNFs in a public cloud with reasonable cost;
2) We account for service demands with different latency requirements, i.e., fixed, variable, and unlimited. We also consider that, to reduce the latency caused by booting a VM and installing VNF instances, a VNF instance can remain in an idle state momentarily after it finishes previous data processing;
3) We consider VNF attributes and the relationship between VNF throughput and amount of allocated computing resources, which further improves resource utilization efficiency but makes the VPS-CD problem more complex;
4) We formulate the VPS-CD problem as a mixed integer linear program (MILP). Given a set of service demands with different parameters, the MILP aims to minimize the cost with latency constraints. As MILP is computationally prohibitive for large networks with many demands, we also develop an efficient heuristic algorithm.

The rest of this study is organized as follows. In Section II, we review related work. The VPS-CD problem statement is provided in Section III. In Sections IV and V, MILP formulation and heuristic approach to solve the problem are presented, respectively. Illustrative numerical results are discussed in Section VI. Section VII concludes this study.

II. Related Work

NFV promises to reduce operation cost, and improve the network efficiency and flexibility [17]. But it also increases the complexity of resource allocation. In [18], authors divided the NFV resource allocation problem into three parts: 1) VNF chain composition, i.e., how to obtain a specific SFC given a request since the order of VNFs may not be fixed; 2) VNF forwarding graph embedding, i.e., strategy of placing VNFs into physical network nodes; and 3) VNF scheduling, exploring how to schedule the execution of VNFs to reduce the latency of network services. The problem of VNF placement and/or scheduling has been well investigated over the past few years.

Authors in [19] first provided a mathematical formulation for the problem of VNF scheduling by resorting to the flexible job-shop problem. In [20], authors formulated the online VPS problem and proposed several algorithms considering service processing time, revenue, etc. Authors in [21] focused on the joint problem of VNF scheduling and traffic steering to minimize the total latency by proposing a MILP and a genetic algorithm-based method. In addition to minimizing the latency of service demands, other aspects should also be accounted for. An energy-aware VNF placement scheme for SFC in datacenters was proposed in [22] together with a power model in servers and switches. Authors in [23] proposed a MILP and a heuristic algorithm to reduce both end-to-end latency and resource consumption. The VPS problem with objective to minimize the operational cost incurred by deploying VNFs without violating service level agreements (SLAs) is studied in [24]. In [25], authors investigated two different types of cost when multiple chained VNFs share the CPU resource: upscaling cost and context-switching cost.

Many other challenges must be addressed to support deploying VNFs in VMs/containers in practice [26]. A virtualized software middlebox platform named ClickOS was introduced in [15]. While it is light-weight, VM booting latency cannot be avoided. Also, to evaluate the performance of VNFs with different thread attributes, authors in [16] conducted several experiments, which verify that a MT VNF can get higher throughput with more computing resources allocated.

Development of cloud computing has attracted attention for SPs to outsource VNFs to public clouds. Two architectures, APLOMB [27] and CloudNaaS [28], were proposed to outsource enterprise middlebox processing to cloud. In [29], authors studied the influence of NFV on CapEx of cloud-based networks. In [6], a support vector regression-based predictive model was used to minimize latency when deploying VNFs in a multi-cloud network. In [30], performance of deploying VNFs in an industry-relevant cloud platform (e.g., OpenStack) in terms of throughput was evaluated. Authors in [10] studied how to reduce cost when outsourcing the SFC to a multi-cloud network. Also, a cost-efficient service-provisioning scheme with QoS guarantee in a content-delivery network (CDN) was proposed in [31]. These two studies have similar objectives to ours; however, there are several differences: 1) We investigate the joint VPS problem in a dynamic traffic scenario, while both [10] and [31] studied the VNF placement problem for static traffic; 2) Different service demands with diverse latency

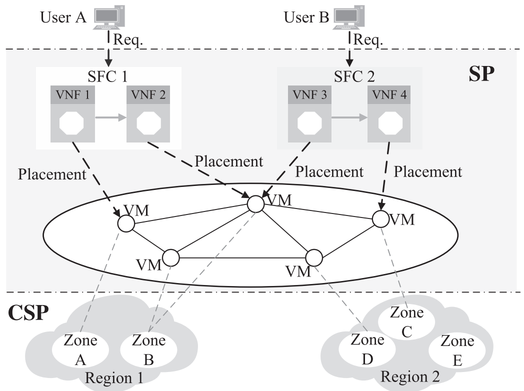


Fig. 1. SFC provisioning in the public cloud network.

requirements are generated in our study, while [31] focused on fixed QoS requirement; 3) We consider VNFs with different thread attributes, and computing resources can achieve different throughputs, making VNF scheduling more complex; and 4) Realistic settings, e.g., VM booting time (VBT), VNF installation time (VIT), etc., are accounted for in our study.

Moreover, the mechanism that a VNF instance can remain in an idle state for a period of time after it finishes any processing task, which is studied in our previous work [32], is also introduced to reduce the latency.

III. VPS-CD Problem Statement

In this section, we first introduce the network model and the metric to evaluate the incurred cost to lease computing and networking resources. Then, the VPS-CD problem is defined. To solve the problem, a conventional low-latency scheme whose objective is to reduce latency is reported and compared with our cost-efficient scheme. Finally, concept of idle state is introduced.

A. Network Model

1) Network Topology and Service Demand: The principle of SPs outsourcing SFCs into the public cloud is illustrated in Fig. 1. A CIP usually has several geographically-distributed datacenters, divided into different regions and zones. For instance, Google has five regions across the United States, in each of which there is one or more zones. Computing resources will be charged at different prices and a transmission fee will be incurred if data is transmitted between different zones. Table I shows pricing scheme of GCP for computing and networking resources in different regions. CIPs offer a pay-as-you-go pricing model. Thus, SPs can set up a VM where and when it is required. Hence, to provision a service demand of a user (e.g., Users A and B in Fig. 1), which requires a SFC consisting of a specific-ordered set of VNFs, the SP's objective is to place these VNFs into the VMs offered by the CIP, and also schedule these VNFs while minimizing cost and satisfying latency requirement of service demands.

Based on the pricing scheme, network topology is represented as $G(V, U, E)$, where V denotes set of VM-capable nodes, U denotes set of user nodes, and E denotes set of physical links. A service demand r is presented as

TABLE I
PRICING SCHEME OF GCP [7]

CPU Resource Pricing	
Region	Price (USD)
Iowa/Oregon/South Carolina	\$0.033174 / CPU hour
Los Angeles	\$0.03797 / CPU hour
Northern Virginia	\$0.037364 / CPU hour
General Network Pricing	
Traffic Type	Price (USD)
Egress between zones	\$0.01 (per GB)
Egress to the same zone	No charge

$r = < \mathbf{s_r}, a_r, d_r, l_r, \overline{s_r}, \overline{d_r} >$, where $\mathbf{s_r}$ is required SFC, a_r is arrival time, d_r is size of data to be processed in GB, l_r is latency requirement, $\overline{s_r}$ is source, and $\overline{d_r}$ is destination. We consider three types of latency requirements:

1) fixed, $l_r = l_r^f$, i.e., demand should be provisioned within deadline l_r^f, which means it is latency-sensitive, e.g., real-time gaming [33];
2) variable, $l_r = [l_r^{req}, l_r^{max}]$, i.e., it is desirable to provision demand within l_r^{req}, but it is acceptable to finish within l_r^{max}, e.g., video streaming [34];
3) unlimited, $l_r \rightarrow +\infty$, i.e., service demand is insensitive to latency, e.g., FTP service [35].

Assume SFC $\mathbf{s_r}$ consists of an ordered set of VNFs denoted as $F_{\mathbf{s_r}} = (f_{\mathbf{s_r},1}, f_{\mathbf{s_r},2}, \ldots, f_{\mathbf{s_r},k})$, where k is length of SFC, i.e., $k = |F_{\mathbf{s_r}}|$. To process the data of a demand, an instance of the required VNF must be installed into a VM with a certain amount of computing resources allocated, which are represented in number of CPU cores for simplicity. A VM can host multiple VNFs, and it will be shut down after all VNFs finish processing user data. The basic throughput of a VNF is P_f Gbps. If a MT VNF is installed in a VM with multiple allocated CPU cores, it can achieve a higher throughput while a ST VNF always has a basic throughput [16]. To simplify the problem, we assume the throughput of a MT VNF is linearly proportional to the amount of CPU cores allocated, i.e., if c CPU cores are allocated for the VM hosting the instance of MT VNF f, the throughput is $c \times P_f$ Gbps.

2) Cost Evaluation: Cost incurred by an SP depends on three components: 1) number of VMs set up; 2) duration a VM keeps running and amount of CPU cores allocated to it; and 3) amount of data transferred between different zones.

In general, our cost model is based on the usage of two types of resources, i.e., computing and networking resources. Note that, if relevant, other kinds of resources, e.g., storage and memory, could be added to our model without impacting the overall proposed scheme. Furthermore, if we consider that some CIPs might charge for the used link bandwidth (e.g., AWS), the cost model can be freely modified to include an additional item.

To quantitatively evaluate the total cost, Eq. (1) is introduced, where M is set of used VMs, c_m is number of CPU cores allocated for VM m, the sum in the brackets is runtime of VM m, P_m^{CPU} is price in dollars per CPU core per time unit, and d (resp. P^{net}) is total size (resp. transmission fee) of data in GB transmitted between different zones. Runtime of VM m is calculated by summing up VBT B_m, runtime of all installed VNF instances (whose set is denoted by $\overline{F_m}$), and time consumed to shut down VM D_m. Furthermore, runtime of VNF instance f is $W_f = t_f^{ins} + \sum_r t_{f,r}^{prs} + t^{idle}$, where t_f^{ins} denotes VIT of VNF instance f, $t_{f,r}^{prs}$ denotes duration that VNF instance f processes data of demand r, and t^{idle} denotes duration of idle state.

$$cost_{total} = \sum_{m \in M} c_m \times \left(B_m + \sum_{f \in \overline{F_m}} W_f + D_m \right) \times P_m^{CPU} + d \times P^{net} \quad (1)$$

B. Low-Latency Scheme vs. CE-VPS Scheme

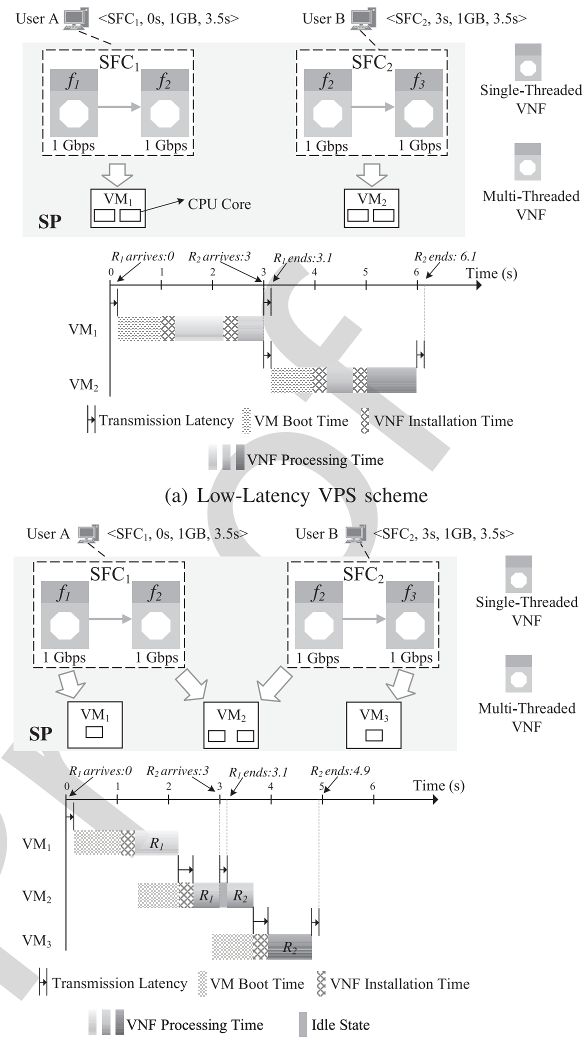
VPS-CD Problem Definition: Given network topology of public clouds with pricing scheme, the objective of placing and scheduling VNFs is to minimize cost incurred by the SP to lease computing and networking resources; also, latency requirements of service demands, which arrive dynamically, should be satisfied.

To solve the VPS-CD problem, we propose a CE-VPS scheme and compare it with a conventional low-latency scheme (C-VPS) whose objective is to minimize latency [22].

1) Comparison of Schemes: C-VPS scheme is shown in Fig. 2(a). There are two service demands R_1 and R_2, which have the same size of data to be processed (1GB) and latency requirement (3.5s), but require different SFCs (SFC_1 and SFC_2, respectively), and arrive at different moments (0s and 3s, respectively). SFC_1 (resp. SFC_2) consists of two VNFs: ST f_1 and MT f_2 (resp. MT f_2 and ST f_3). Basic throughput of all VNFs are assumed to be 1Gbps.

In C-VPS, different VNFs requested by a service demand are installed in a single VM to avoid data transmission latency. As shown in Fig. 2(a), a transmission latency of 0.1s is initially incurred (capacity of connection between user node and datacenter in public cloud is assumed to be 10Gbps in this example). To provision service demand R_1, we first set up a VM with two allocated CPU cores, which incurs a VM booting latency (1s in the example) and a VNF installation latency (0.2s). Since f_1 is ST and basic throughput is 1 Gbps, processing latency of f_1 is 1s. After that, instance of f_2 is installed in same VM, whose throughput is doubled with 2 CPU cores, i.e., 2Gbps, and processing latency is 0.5s. Finally, data is transferred from VM_1 to the destination with a transmission latency of 0.1s. In conclusion, total latency of demand R_1 is 3.1s. However, as f_1 is ST, one CPU core of VM_1 is idle, leading to a waste of computing resources. The example is analogous for demand R_2.

However, the proposed CE-VPS scheme can place and schedule VNFs based on their attributes, as shown in Fig. 2(b). For R_1, another VM with two CPU cores allocated is set up for the instance of f_2 to achieve higher throughput. Also, since we can boot VM_2 in advance before the data processed



(a) Low-Latency VPS scheme

(b) CE-VPS scheme

Fig. 2. Comparison of different schemes.

by the instance of f_1 arrives, latency can be reduced. Basic bandwidth of connection from VM_1 to VM_2 is assumed to be 5Gbps (actually, bandwidth can be customized), hence transmission latency incurred is 0.2s. For the instance of f_2 installed in VM_2, it remains in idle state for a period of time. During this period, demand R_2 arrives, and the instance can start to process its data immediately. Hence, total latency of R_2 can be decreased from 3.1s in C-VPS to 1.9s. Assume price of per-CPU core is \$$P$/s, then total costs of C-VPS and CE-VPS can be calculated by $2.9 \times 2 \times 2 \times P = 11.6P$ dollars and $(2.2 \times 2 + 2.3 \times 2) \times P = 9P$ dollars, respectively. Thus, CE-VPS achieves significantly-lower cost (24%) compared to C-VPS.

2) Idle State: In this subsection, we recall the concept of the idle state through an example. Fig. 3(a) shows two service demands R_1 and R_2, requiring the same SFC, that arrive in the network at different instants. To provision R_1, VM_1 is booted and a new instance of VNF f_1 is installed, incurring some latency. In a conventional scheme, after f_1 finishes processing the data of R_1, data will be transmitted to the instance of VNF f_2; meanwhile, the instance of f_1 will be removed to save computing resources. When R_2 arrives, the same procedure

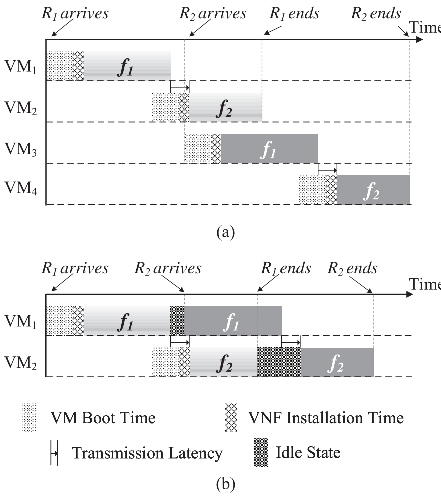


Fig. 3. Different VPS strategies: (a) without idle state and (b) with idle state.

is executed, unnecessarily increasing latency (i.e., intuitively, it would have been preferable to maintain f_1 and f_2 active, avoiding the re-booting).

In our proposed scheme, VNF instances (e.g., VNF f_1 and f_2 as shown in Fig. 3(b)) can remain in idle state after they finish the previous task. Thus, when R_2 arrives, a VNF instance can start working immediately without the need for booting a new VM and re-installing a VNF instance. When the load of service demands is high, idle state can improve the network performance remarkably in terms of additional computing resources. Details about how idle state can affect network performance in terms of latency, resource utilization, etc., can be found in [32]. In this study, a fixed duration of idle state is assumed for VNF instances.

IV. MILP Formulation

In this section, the CE-VPS scheme is formulated as a MILP, which tries to minimize the total cost spent by the SP on computing and networking resources provided by the CIP.

Notations	**Description**
$G(V, U, E)$	Cloud network topology, where V is set of VM-capable nodes, U is set of user nodes, and E is set of links, $(i, j) \in E$.
$L^{\kappa}_{(i,j)}$	Length of κ^{th} shortest path between nodes i and j, $\kappa \in K$.
H^z_i	1 if node i belongs to zone $z, z \in Z$, and Z is set of zones.
P^{CPU}_z	Price of a CPU core per hour in zone z.
P^{net}	Price of data when traffic transferred between different zones (per GB).
Φ	Speed of light in fiber, 200 km/ms.
Ω	Transmission rate from a user node to a VM-capable datacenter node.
Ψ	A large integer constant.
C	Maximum number of available CPU cores, $c \in [1, C]$.
N	Highest level of egress network capacity that a VM can have, $n \in [1, N]$. Basic network capacity is Θ Gbps.
F, M	Set of VNFs, $f \in F$, and set of VMs, $m \in M$, respectively.
Δ_f	Indicator denoting whether VNF f is MT, i.e., $\Delta_f = 1$, or ST, i.e., $\Delta_f = 0$.
P_f	Basic processing capacity of VNF f.
I, B, D	VIT, VBT, and time consumed to remove a VM, respectively.
S	Set of SFCs, $\mathbf{s} \in S$.
$F_{\mathbf{s}}$	Set of VNFs in SFC $\mathbf{s}$, $F_{\mathbf{s}} \subseteq F$. Let $f_{\mathbf{s},k}$ denote the k^{th} VNF in SFC $\mathbf{s}$, $f_{\mathbf{s},k} \in F_{\mathbf{s}}$.
R	Set of service demands, $r \in R$.

Variables	**Description**
φ	Float variable denoting total cost.
x_m	Integer variable denoting time when VM m is initialized.
y_m	Integer variable denoting time when VM m is removed.
$l^i_{m,c}$	Integer variable denoting runtime of VM m in node i with c CPU cores.
$q^c_m \in \{0, 1\}$	1 if VM m is allocated with c CPU cores.
$g^i_m \in \{0, 1\}$	1 if VM m is initialized in node i.
$h^{f,m}_r \in \{0, 1\}$	1 if an instance of VNF f requested by demand r, is installed in VM m.
$h^{f,z}_r \in \{0, 1\}$	1 if VNF f is installed in a VM that belongs to zone z.
p^f_r	Integer variable denoting the moment when VNF f requested by demand r starts to process user data.
$p^{f,f'}_{r,r'} \in \{0, 1\}$	1 if VNF f requested by r starts to process data before VNF f' requested by r' does.
w^f_r	Integer variable denoting processing latency of VNF f requested by demand r.
w^k_r	Integer variable denoting transmission latency between VMs hosting k^{th} and $(k+1)^{th}$ VNF instances.
$u^k_r \in \{0, 1\}$	1 if k^{th} and $(k+1)^{th}$ VNF instances are installed in different VMs.
$a^n_m \in \{0, 1\}$	1 if egress network capacity level allocated for VM m is n.
o^k_r	Integer variable denoting propagation latency between VMs hosting k^{th} and $(k+1)^{th}$ VNF instances.
$z^r_k \in \{0, 1\}$	1 if locations of VMs hosting k^{th} and $(k+1)^{th}$ VNF instances belong to different zones.
$e^{r,\kappa}_{(i,j)} \in \{0, 1\}$	1 if the κ^{th} shortest path between nodes i and j is established for demand r.

A. Objective Function

$$Minimize(\varphi) \tag{2}$$

The MILP objective is to minimize the total cost as in Eq. (3).

$$\varphi = \sum_{z \in Z} \sum_{i \in V} \sum_{c \in [1,C]} \sum_{m \in M} c \times l^i_{m,c} \times H^z_i \times P^{CPU}_z + \sum_{r \in R} \sum_{k \in [1, |F_{\mathbf{s_r}}|-1]} d_r \times z^r_k \times P^{net} \tag{3}$$

B. Latency Constraints

$$a_r + l_r \geq p_r^{f_{\mathbf{s_r},k}} + w_r^{f_{\mathbf{s_r},k}} + \frac{d_r}{\Omega} + \frac{\sum_{i\in V} e_{(i,\overline{d_r})}^{r,\kappa} \times L_{(i,\overline{d_r})}^{\kappa}}{\Phi}, \quad \forall r\in R,\ f_{\mathbf{s_r},k}\in F_{\mathbf{s_r}},\ \kappa\in K \tag{4}$$

Eq. (4) ensures the latency requirement of demand r. First two items on right side ensure the last required VNF finishes processing all data before the deadline. Transmission and propagation latency are also considered.

$$p_r^{f_{\mathbf{s_r},1}} \geq a_r + \frac{d_r}{\Omega} + \frac{\sum_{i\in V} e_{(\overline{s_r},i)}^{r,\kappa} \times L_{(\overline{s_r},i)}^{\kappa}}{\Psi}, \quad \forall r\in R,\ f_{\mathbf{s_r},1}\in F_{\mathbf{s_r}},\ \kappa\in K \tag{5}$$

Eq. (5) ensures the first VNF instance of the SFC can start to process data only after the data has been transferred from the user node to the node where the VM is hosting the first VNF through the κ^{th} shortest path, which induces some transmission and propagation latency.

$$p_r^{f_{\mathbf{s_r},k}} + w_r^{f_{\mathbf{s_r},k}} + w_r^k + o_r^k \leq p_r^{f_{\mathbf{s_r},k+1}}, \quad \forall r\in R,\ k\in[1,|F_{\mathbf{s_r}}|-1],\ f_{\mathbf{s_r},k}\in F_{\mathbf{s_r}} \tag{6}$$

Eq. (6) ensures that processing at VNF $f_{\mathbf{s_r},k+1}$ should not start until the data has been processed by the previous VNF and transferred to the VM hosting VNF $f_{\mathbf{s_r},k+1}$.

$$w_r^f \geq \frac{d_r}{P_f} \times (\Delta_f \times \frac{q_m^c}{c} + 1 - \Delta_f) + \Psi \times (h_r^{f,m} - 1), \quad \forall r\in R,\quad f\in F_{\mathbf{s_r}},\ c\in[1,C],\ m\in M \tag{7}$$

Eq. (7) calculates the VNF processing latency. Specifically, if the VNF is MT, that is $\Delta_f = 1$, the latency is calculated through multiplying basic processing capacity P_f by the number of CPU cores allocated. Otherwise, the latency is calculated only in terms of basic processing capacity.

$$w_r^k \geq \frac{d_r}{\Theta \times n} \times a_m^n + \Psi \times (h_r^{f_{\mathbf{s_r},k},m} + u_r^k - 2), \quad \forall r\in R,\ k\in[1,|F_{\mathbf{s_r}}|-1],\ f_{\mathbf{s_r},k}\in F_{\mathbf{s_r}},\ n\in[1,N],\ m\in M \tag{8}$$

Eq. (8) calculates that transmission latency between VNFs $f_{\mathbf{s_r},k}$ and $f_{\mathbf{s_r},k+1}$, which applies only when they are deployed in different VMs, i.e., both $h_r^{f_{\mathbf{s_r},k},m}$ and u_r^k equal one. The latency is calculated in terms of the egress network capacity level n allocated to the VM that hosts $f_{\mathbf{s_r},k}$, where a higher level means the latency can be reduced.

$$o_r^k \geq (h_r^{f_{\mathbf{s_r},k},m} + g_m^i + h_r^{f_{\mathbf{s_r},k+1},m'} + g_{m'}^j - 3) \times \frac{L_{(i,j)}^{\kappa}}{\Psi} + (e_{(i,j)}^{r,\kappa} - 1) \times \Psi, \quad \forall r\in R,\ k\in[1,|F_{\mathbf{s_r}}|-1],\ f_{\mathbf{s_r},k}, f_{\mathbf{s_r},k+1}\in F_{\mathbf{s_r}}, m, m'\in M, i,j\in V,\ \kappa\in K \tag{9}$$

$$1 \geq \sum_{\kappa\in K} e_{(i,j)}^{r,\kappa} \geq h_r^{f,m} + g_m^i + h_r^{f',m'} + g_{m'}^j - 3, \quad \forall r\in R, f,\ f'\in F_{\mathbf{s_r}},\ m,m'\in M,\ i,j\in V \tag{10}$$

Eq. (9) calculates propagation latency of the κ^{th} shortest path between the two VMs hosting two consecutive VNFs in a service chain. This applies only when two VNFs are deployed in different nodes, i.e., when the sum of all variables within the brace equals one. Eq. (10) ensures at most one among K-shortest paths is selected.

C. VNF Placement Constraints

$$\sum_{m\in M} h_r^{f,m} = 1, \quad \forall r\in R,\ f\in F_{\mathbf{s_r}} \tag{11}$$

Eq. (11) ensures that each VNF of the requested SFC should be installed in only one VM.

$$\sum_{r\in R}\sum_{f\in F_{\mathbf{s_r}}} h_r^{f,m} \geq \sum_{i\in V} g_m^i \geq \sum_{r\in R}\sum_{f\in F_{\mathbf{s_r}}} h_r^{f,m}/\Psi, \quad \forall m\in M \tag{12}$$

Eq. (12) ensures that, if a VM is responsible to process user data, it should be mapped into a VM-capable node.

$$\sum_{\kappa\in K} e_{(\overline{s_r},i)}^{\kappa} \geq h_r^{f_{\mathbf{s_r},1},m} + g_m^i - 1, \quad \forall r\in R,\ f_{\mathbf{s_r},1}\in F_{\mathbf{s_r}},\ m\in M,\ i\in V \tag{13}$$

$$\sum_{\kappa\in K} e_{(i,\overline{d_r})}^{\kappa} \geq h_r^{f_{\mathbf{s_r},|F_{\mathbf{s_r}}|},m} + g_m^i - 1, \quad \forall r\in R,\ f_{\mathbf{s_r},|F_{\mathbf{s_r}}|}\in F_{\mathbf{s_r}},\ m\in M,\ i\in V \tag{14}$$

Eqs. (13)-(14) ensure a connection is established from the source to the node where the first VNF is installed, and from the node where the last VNF is installed to the destination.

D. VNF Scheduling Constraints

$$x_m + B + I \leq p_r^f + \Psi \times (1 - h_r^{f,m}), \quad \forall m\in M,\ r\in R,\ f\in F_{\mathbf{s_r}} \tag{15}$$

Eq. (15) ensures the VM should boot before any VNF installed in it begins to process data.

$$y_m \geq D + p_r^f + \Psi \times (h_r^{f,m} - 1) + w_r^f, \quad \forall m\in M,\ r\in R,\ f\in F_{\mathbf{s_r}} \tag{16}$$

Eq. (16) ensures the VM can be shut down after all hosting VNFs have finished their tasks.

$$2 - h_m^{f_{\mathbf{s_r},k}} - h_m^{f_{\mathbf{s_r},k+1}} \geq u_r^k \geq h_m^{f_{\mathbf{s_r},k}} + h_{m'}^{f_{\mathbf{s_r},k+1}} - 1, \quad \forall r\in R,\ f_{\mathbf{s_r},k}, f_{\mathbf{s_r},k+1}\in F_{\mathbf{s_r}},\ m,m'\in M,\ m\neq m' \tag{17}$$

Eq. (17) determines value of u_r^k, which is used to denote whether two consecutive VNFs in a SFC are installed in two different VMs. u_r^k equals 1 iff the VNFs are deployed in two different VMs m and m', where both variables $h_m^{f_{\mathbf{s_r},k}}$ and $h_{m'}^{f_{\mathbf{s_r},k+1}}$ equal 1.

$$z_k^r \geq h_r^{f_{\mathbf{s_r},k},z} + h_r^{f_{\mathbf{s_r},k+1},z'} - 1, \quad \forall r\in R,\ f_{\mathbf{s_r},k},\ f_{\mathbf{s_r},k+1}\in F_{\mathbf{s_r}}, z,\ z'\in Z,\ z\neq z' \tag{18}$$

$$h_r^{f,z} \geq (h_r^{f,m} + g_m^i - 1) \times H_i^z, \quad \forall r\in R,\ f\in F_{\mathbf{s_r}},\ i\in V,\ z\in Z,\ m\in M \tag{19}$$

Eqs. (18)-(19) determine whether VNFs $f_{\mathbf{s_r},k}$ and $f_{\mathbf{s_r},k+1}$ are located in two nodes that belong to different zones.

$$y_m - x_m + (q_m^c + g_m^i - 2) \times \Psi \leq l_{m,c}^i \leq (2 - q_m^c - g_m^i) \times \Psi, \quad \forall m\in M,\ c\in[1,C],\ i\in V \tag{20}$$

Eq. (20) determines runtime of VM m with c CPU cores.

$$1 \geq p_{r,r'}^{f,f'} + p_{r',r}^{f',f} \geq h_r^{f,m} + h_{r'}^{f',m} - 1, \quad \forall r,r'\in R,\ f\in F_{\mathbf{s_r}},\ f'\in F_{\mathbf{s_{r'}}},\ m\in M \tag{21}$$

$$p_r^f + w_r^f + I - p_{r'}^{f'} \leq (1 - p_{r,r'}^{f,f'}),$$
$$\forall r,\ r' \in R,\ f \in F_{\mathbf{s_r}},\ f' \in F_{\mathbf{s_{r'}}} \quad (22)$$

Eqs. (21)-(22) ensure that, if two VNFs f and f' requested by different demands are installed in the same VM, which means both $h_r^{f,m}$ and $h_{r'}^{f',m}$ equal 1, the VM cannot process the two requests at the same time. Hence, the processing order is determined by Eq. (22), where if $p_{r,r'}^{f,f'}$ equals one, meaning that, if VNF f first processes data, VNF f' cannot work until VNF f finishes and an instance is installed.

E. Resource-Allocation Constraints

$$1 \geq \sum_{c \in [1,C]} q_m^c \geq \sum_{r \in R} \sum_{f \in F_{\mathbf{s_r}}} h_r^{f,m}/\Psi, \quad \forall m \in M \quad (23)$$

Eq. (23) calculates the number of CPU cores allocated to a VM.

In the MILP, dominant number of variables is among $l_{m,c}^i$, $h_r^{f,m}$, $h_r^{f,z}$, and $p_{r,r'}^{f,f'}$, which are O($|V| \times |M| \times C$), O($|F| \times |M| \times |R|$), O($|F| \times |Z| \times |R|$), and O($|F|^2 \times |R|^2$), respectively. $|V|$ is size of VM-capable node set, $|M|$ is size of VM set, $|F|$ is size of VNF set, $|R|$ is size of service demand set, and $|Z|$ is number of zones. About constraints, the dominant number is among (9) and (21), which are of complexity O($|F| \times |R| \times |M|^2 \times |V|^2$) and O($|F|^2 \times |R|^2 \times |M|$), respectively.

V. Heuristic Approach

The MILP is computationally prohibitive for large networks. Hence, an efficient heuristic is developed to achieve near-optimal performance for dynamic service demands in large networks. The heuristic for CE-VPS consists of three sub-algorithms, i.e., optimal zone determination (OZD), latency requirement verification (LRV), and service demand provisioning (SDP).

A. OZD Algorithm

OZD is responsible to find the optimal zone to host as many instances of required VNFs as possible, where transmission cost is minimized. We construct a $|Z| \times |F_{\mathbf{s}}|$ matrix M, which is represented as follows. Element m_{f_k,z_j} equals 1 if there is at least one instance of VNF f_k in zone z_j, and 0 otherwise. Next, for each row, we do *AND* operation between any two adjacent elements and sum the results up to get vector V, where the largest value v_j denotes that zone z_j hosts most qualified VNFs so data transmission fee can be decreased.

$$V = AND(M) = \begin{bmatrix} \sum_{k \in [1,|F_{\mathbf{s}}|-1]} m_{f_k,z_1} \,\&\, m_{f_{k+1},z_1} \\ \sum_{k \in [1,|F_{\mathbf{s}}|-1]} m_{f_k,z_2} \,\&\, m_{f_{k+1},z_2} \\ \vdots \\ \sum_{k \in [1,|F_{\mathbf{s}}|-1]} m_{f_k,z_{|Z|}} \,\&\, m_{f_{k+1},z_{|Z|}} \end{bmatrix}$$

$$= \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{|Z|} \end{bmatrix}, \quad M = \begin{bmatrix} m_{f_1,z_1} & \cdots & m_{f_{|F_{\mathbf{s}}|},z_1} \\ \vdots & \ddots & \vdots \\ m_{f_1,z_{|Z|}} & \cdots & m_{f_{|F_{\mathbf{s}}|},z_{|Z|}} \end{bmatrix}$$

The pseudo-code of OZD (**Algorithm 1**) is stated as follows. In **Algorithm 1**, we first determine the VMs that host the instances of required VNFs. In line 2, relevant parameters are initialized, where time indicator T is used to estimate the time at which each VNF in $F_{\mathbf{s}}$ should start to process the data. Candidate sets $I_{f_1}, I_{f_2}, \ldots, I_{f_k}$ are used to store the qualified instances for each VNF. Note that, for the first VNF in SFC $\mathbf{s}$, processing should start after the data is transmitted from user node $\overline{s}$ to the datacenter with a latency of d/C_{in}, where d is data size and C_{in} is ingress network capacity from the user to the public cloud. From line 4 to 6, if the demand must finish before deadline DDL, each VNF instance is checked whether it is available at a certain moment, and time indicator T is updated with the estimated processing time according to the basic throughput of the VNF. Otherwise, all instances are selected as candidates since the demand is insensitive to latency as stated in line 8. In line 9, to find the optimal zone, the matrix-based method presented above is employed, and then the results are returned.

Algorithm 1: OZD Algorithm

Input: Service demand r and its deadline DDL
Output: Optimal zone z, and set of qualified VNFs Q in z

```
1 Find set of VMs hosting VNFs required by demand r,
    i.e., F_s = (f_1, f_2, ..., f_|F_s|);
2 Initialize time indicator T = a + d/C_in and candidate
    instance sets, i.e., I_f1, I_f2, ... I_fk = ∅;
3 if DDL ≠ +∞ then
4     for each f ∈ F_s do
5         Add the instance of VNF f to I_f, if it is available
            at time T;
6         Update T = T + d/P_f;
7 else
8     Add all existing instances for each VNF to sets
        I_f1, I_f2, ..., I_fk correspondingly;
9 Employ the matrix-based method to find optimal zone z
    and VNF set Q, and return;
```

Complexity: In line 1, complexity of obtaining VMs is O(V_{max}), where V_{max} denotes maximum number of VMs. From line 4 to 6, it requires O($V_{max}F_{max}$) to check the availability of each instance of each required VNF, where F_{max} is maximum number of the VNFs in a SFC. Matrix operation to find the optimal zone in line 9 requires O($V_{max}|Z|$). Taking all steps into consideration, time complexity of **Algorithm 1** is O($V_{max}(F_{max} + |Z|)$).

B. LRV Algorithm

LRV is responsible to check whether user data can be processed by candidate instance set Λ within deadline DDL.

Pseudo-code of LRV (**Algorithm 2**) can be summarized as follows. Time indicator T is initialized in line 1. Next, in line 3, propagation and transmission latency is calculated for the path from source of the demand to the datacenter hosting the first VNF instance. Specifically, Yen's algorithm [36] is employed to calculate K-shortest paths, and the one with

Algorithm 2: LRV Algorithm

Input: Arrival time a of service demand r, candidate instance set Λ, deadline DDL, and size of data to be processed d
Output: **true**, if DDL can be met; **false**, otherwise
1 Initialize time indicator $T = 0$;
2 Calculate K-shortest path from source $\overline{s}$ of r to location of first VNF instance i_{f_1} in Λ and select the one with least latency $lat(\overline{s}, i_{f_1})$;
3 Set $T = a + d/C_{in} + lat(\overline{s}, i_{f_1})$;
4 **for** *each* $k \in |\Lambda|$ **do**
5   Get its throughput $P^*_{f_k}$, available time TS_{f_k}, and egress network capacity C_{f_k};
6     $T = max(TS_{f_k}, T) + d/P^*_{f_k}$;
7   **if** $k \leq |\Lambda| - 1$ **then**
8         $T = T + d/C_{f_k} + lat(i_{f_k}, i_{f_{k+1}})$;
9   **else**
10         $T = T + d/C_{eg} + lat(i_{f_k}, \overline{d})$;
11 **return** $T \leq DDL$? **true**: **false**;

least latency is selected. From lines 4-10, T is updated after each VNF instance processes the data. Specifically, in line 5, relevant parameters are obtained, where $P^*_{f_k}$ is throughput of VNF instance f_k, TS_{f_k} is the time that f_k can actually start to process the data, and C_{f_k} is egress network capacity of the VM hosting f_k. Egress network capacity is flexible and can be customized. In line 6, T is updated according to the actual start processing time. Then, in lines 7-10, transmission latency and propagation latency are considered. Finally, the result is returned.

Complexity: In line 2, complexity of Yen's algorithm is O($K|V|(|E| + |V| \log |V|)$). Complexity of the for loop from line 4 to 10 is O($F_{max}K|V|(|E| + |V| \log |V|)$). In conclusion, complexity of **Algorithm 2** is O($F_{max}K|V|(|E| + |V| \log |V|)$).

C. SDP Algorithm

SDP is responsible to serve a single demand that arrives dynamically, and pseudo-code of SDP is reported in **Algorithm 3**. In lines 1-3, deadline DDL is determined based on type of latency requirement of r. **Algorithm 1** is called to find optimal zone z and corresponding VNFs in line 4. Lines 5-18 employ existing or newly-installed VNF instances to serve r. Specifically, in lines 6-7, an existeing instance of the required VNF in zone z is selected. If there is no available instances, the VNF prior to it is checked to see whether they are both ST or MT, in lines 9-10. If they have the same attribute, a new VNF instance is installed in the VM hosting the prior VNF instance. In lines 12-14, if previous procedures fail, other zones are checked to determine whether there are qualified instances. In lines 15-17, a new VM will be booted, for which number of required CPU cores is calculated in Eq. (24).

Algorithm 3: SDP Algorithm

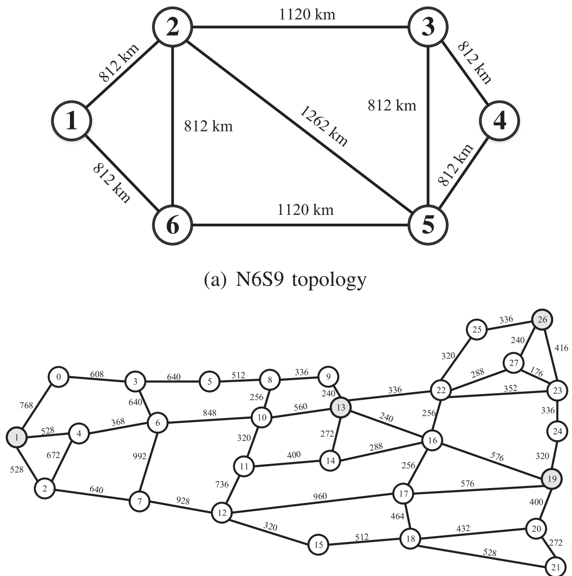
Input: Service demand r
1 Initialize set of VNF instances to serve r, i.e., $\Lambda = \varnothing$, deadline for r, i.e., $DDL = l_r$;
2 **if** *r has a variable latency requirement* **then**
3   Set $DDL = l_r^{req}$;
4 Call **Algorithm 1** with $< r, DDL >$ to find optimal zone z and qualified VNF set Q;
5 **for** *each* $f_k \in F_{\mathbf{s}}, k \leq |F_{\mathbf{s}}|$ **do**
6   **if** $f_k \in Q$ **then**
7   Find qualified instance i_f in zone z, $\Lambda = \Lambda \cup i_f$;
8   **else**
9   **if** *$f_{k-1} \in Q$, and meantime, f_{k-1} and f_k have the same attribute* **then**
10   Install an instance i_{f_k} in f_{k-1}' VM;
11   **else**
12   Check all instances of f_k in other zones;
13   **if** *there exist qualified instances* **then**
14   Select the one i_{f_k} in the most inexpensive zone;
15   **else**
16   Set up a new VM in zone z with N CPU cores, calculated by Eq. (24);
17   Install an instance i_{f_k} in the VM;
18         $\Lambda = \Lambda \cup i_{f_k}, Q = Q \cup f_k$;
19 Call **Algorithm 2** with args $< a, \Lambda, DDL, d >$ and get the returned result $flag$;
20 **if** $flag ==$ ***true*** **then**
21   Serve the demand with Λ;
22 **else if** $DDL == l_r^{req}$ **then**
23   Set $DDL = l_r^{max}$, **go to Step** 4;

In Eq. (24), if VNF f is ST, number of required CPU core is one. Otherwise, it is calculated according to deadline DDL and processing latency of other VNFs, i.e., $\sum_{f' \in F_{\mathbf{s}}/f} \frac{d}{P_{f'}}$. Note that processing latencies of other VNFs are estimated in terms of their basic throughput; hence, actual processing latency can be smaller than the estimated value.

$$N = \begin{cases} 1, & f \text{ is ST} \\ \left\lceil d/\left(\left(DDL - \sum_{f' \in F_{\mathbf{s}}/f} d/P_{f'}\right) \times P_f\right)\right\rceil, & f \text{ is MT} \end{cases} \tag{24}$$

In line 19, **Algorithm 2** is called to verify whether the latency requirement is met. From line 22 to 23, if LRV fails, we check whether the latency requirement can be relaxed.

Complexity: In line 4, **Algorithm 1** is called and its complexity has been analyzed. Complexity of the for loop in lines 5-18 is O($F_{max}V_{max}$). Complexity of **Algorithm 2** has



(a) N6S9 topology

(b) US Backbone topology with link length in km

Fig. 4. Network topologies used in simulation.

also been analyzed. Taking all steps into consideration, complexity of **Algorithm 3** is $O(F_{max}(K|V|(|E|+|V|\log|V|)+V_{max})+V_{max}|Z|)$, which runs in polynomial time.

VI. Performance Evaluation

In this section, we first evaluate the performance of CE-VPS through the MILP in a small-scale network. Then, the heuristic algorithms of CE-VPS and three conventional VPS schemes are compared in large-scale networks.

A. Simulation Setup

The MILP is implemented using ILOG CPLEX v12.5, and heuristic algorithms are coded in Python. All simulations run on a personal computer with Intel i7-7600 2.9 GHz CPU, 16 GB RAM, and Windows 10 operating system.

For MILP, network topology N6S9 shown in Fig. 4(a) is employed, which includes two NFV-capable datacenters belonging to different zones. Prices of CPU core in each zone are \$0.03/hour and \$0.04/hour, respectively. Networking price for data transmission between zones is \$0.01/GB. Data sizes and latency requirements (including fixed and variable) of demands are uniformly distributed in the range [0.1GB, 2GB] and [0.1s, 15s], respectively, according to different types of applications [37]. Further, value of latency requirement is set as infinite for latency-insensitive demands. We assume three VNFs, whose throughputs and attributes are (1Gbps, MT), (2Gbps, MT), and (4Gbps, ST). Also, VBT and VIT are assumed to be 20ms and 10ms, respectively. Basic network capacity Θ is 5 Gbps, and if a VM is allocated with c CPU cores, its egress network capacity is $c \times \Theta$ Gbps [7]. Performance of MILP is compared with CE-VPS heuristic by giving as input the same set of static service demands. Besides, three baseline schemes whose main procedures are as follows.

1) *CPVNF [31]:* For each demand, servers (replaced by VMs in this study for fair comparison) with higher importance rank metric (SIR) are selected for required VNFs. SIR is originally defined according to the remaining computing capacity of a server, bandwidth capacity of links, and whether VNF instances preexist. Since in our study we are considering a public cloud, and computing resource and bandwidth in public cloud can be regarded as unlimited (e.g., the link bandwidth between datacenters can reach over 1 Pbps in Google datacenter network [38]), we modify SIR definition by using the available time and number of CPU cores of a VM instead of remaining computing and bandwidth capacity.
2) *Best-Availability [20]:* For each required VNF of a demand, the scheme attempts to place it into a VM whose current demand queue has the earliest finish time (i.e., best availability).
3) *Low-Latency [22]:* This scheme sets up a new VM to host all VNF instances for each demand. Number of CPU cores allocated to the VM is calculated according to Eq. (24).

The heuristic approach is conducted on US Backbone topology [39], as shown in Fig. 4(b), and there are four datacenters belonging to four different zones. Prices of CPU core in datacenters 1, 13, 19, and 26 are \$0.034/h, \$0.038/h, \$0.04/h, and \$0.035/h, respectively, based on the pricing scheme of GCP [7] as stated before. Traffic arrives dynamically according to a Poisson distribution with λ demands per second. Four different SFCs and six optional VNFs, i.e., NAT, FW, traffic monitor (TM), WAN optimization controller (WOC), intrusion detection and prevention system (IDPS), and video optimization controller (VOC), are considered [25], [40]. Four SFCs are: Web Service (NAT-FW-TM-WOC-IDPS), VoIP (NAT-FW-TM-FW-NAT), Video Streaming (NAT-FW-TM-VOC-IDPS), and Online Gaming (NAT-FW-VOC-WOC-IDPS). We assume throughputs and attributes of optional VNFs are (2Gbps, ST), (1Gbps, MT), (1Gbps, ST), (2Gbps, MT), (4Gbps, ST), and (4Gbps, MT) [40], [41], respectively. The duration of idle state is to 2 seconds according to our previous work. Other parameters are the same as that in the MILP. To obtain good statistical confidence of results, the simulation is run 20 times for each traffic load and we take the average. In each simulation run, 10,000 demands are generated.

B. Performance Comparison: MILP and Heuristics

Fig. 5 shows the cost of different schemes, which is normalized to the largest value achieved by Best-Availability. We observe that MILP can reduce the cost by over 10%, 11%, and 14% on average compared with Low-Latency, CPVNF, and Best-Availability Algorithms, respectively. Moreover, CE-VPS achieves close-to-optimal results, where average gap is 4.7%. Best-Availability has the worst performance, as it prefers to select instances with earliest finish time, even it is in a different zone incurring data transmission cost.

Table II compares the running time of different schemes. We find that, with increasing number of demands, time consumed by MILP increases significantly. It spends over 5 hours on 10 demands, which becomes impractical to be employed. However, all heuristic approaches obtain results with around 100ms.

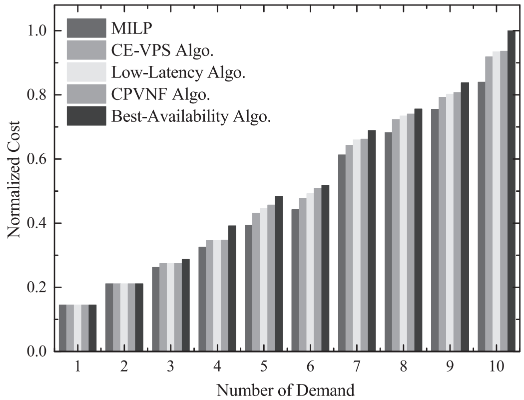


Fig. 5. Normalized cost.

TABLE II

AVG. RUNNING TIME OF DIFFERENT SCHEMES (s)

Number of demands	2	4	6	8	10	12
MILP	5	65	431	4412	20431	-
CE-VPS	0.099	0.102	0.112	0.119	0.119	0.117
Low-Latency	0.083	0.083	0.083	0.086	0.088	0.091
CPVNF	0.100	0.098	0.109	0.113	0.112	0.111
Best-Availability	0.096	0.109	0.110	0.121	0.978	0.112

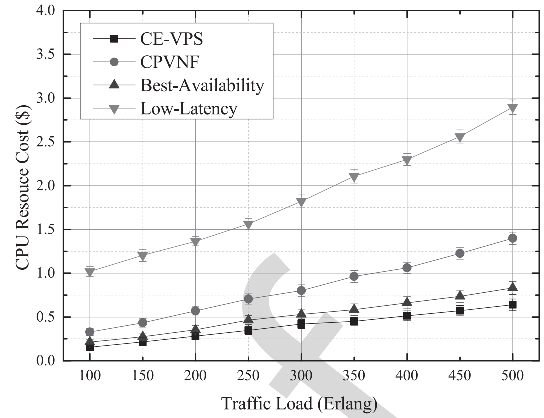
C. Performance Comparison of Different Heuristics

The performance of our proposed CE-VPS heuristic and three baseline algorithms are evaluated according to CPU resource cost, data transmission cost, average latency, and average number of used VMs per service demand. These results are plotted with a confidence level of 95%.

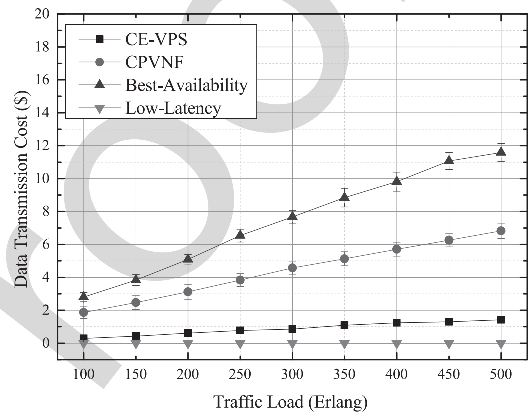
In Fig. 6(a), results show that CPU resource cost increases almost linearly with traffic load for all schemes. Compared to Best-Availability, CPVNF, and Low-Latency schemes, CE-VPS can reduce CPU resource cost by about 23%, 48%, and 78%, respectively, at traffic load of 500 Erlang. The benefits come from the fact that, in CE-VPS, an optimal zone with low price of CPU resource is found to serve the demand. Moreover, in Low-Latency, a VM is established for each demand to host all required VNFs, achieving a highest cost of CPU resources.

Data transmission cost is compared for different schemes in Fig. 6(b). Note that data transmission costs of CE-VPS, CPVNF, and Best-Availability schemes are much higher than costs of CPU resources. However, CE-VPS achieves much lower transmission cost than CPVNF and Best-Availability schemes, and the reduction can reach as high as 76% and 88%, respectively, at traffic load of 500 Erlang. For Low-Latency, there is no transmission cost incurred, but total cost of CE-VPS is still lower than that of Low-Latency.

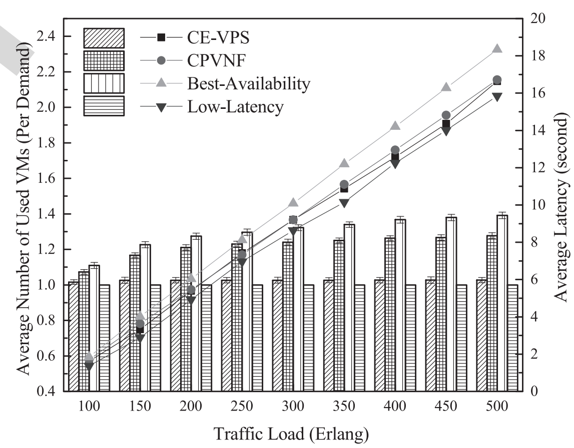
Next, we evaluate the performance in terms of average number of used VMs per service demand for different schemes in Fig. 6(c). In Low-Latency, one VM is set up for each demand, hence the value always equals to one. CE-VPS also achieves a low VM usage, meaning that the frequency at which VMs hosting required VNF instances are reused by multiple demands is much higher than in CPVNF and Best-Availability



(a) CPU resource cost



(b) Data transmission cost



(c) Average number of used VMs and average latency of different schemes

Fig. 6. Simulation results of different schemes.

schemes, which contributes to decreasing the cost of booting new VMs and installing new VNF instances. The frequent reuse benefits from the fact that: 1) scheduling of VNFs can be conducted more efficiently when VNF attributes are considered; and 2) idle state promotes the reuse of VNF instances among multiple demands.

Average latencies of different schemes are also compared, where Low-Latency achieves the best performance as the transmission latency between different VMs is avoided. However, latency reduction between Low-Latency and CE-VPS

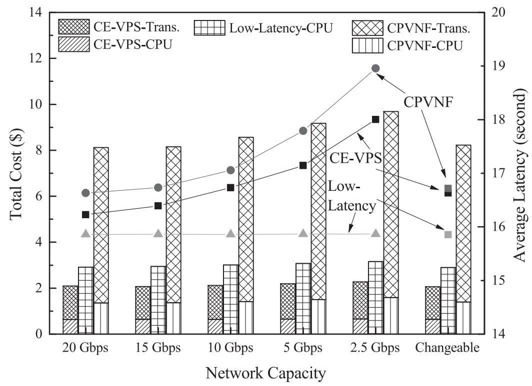


Fig. 7. Total cost (bars) and average latency (curves) for different network capacities.

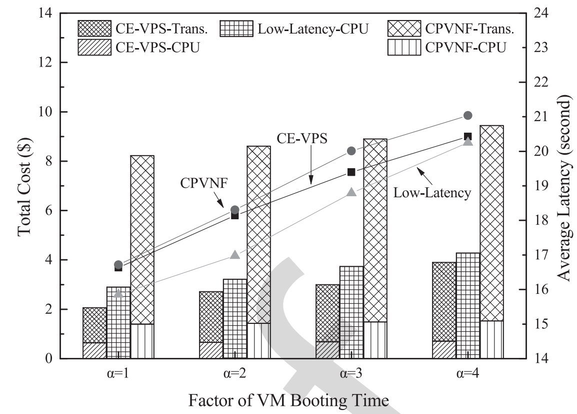
is only about 3% on average, since for CE-VPS, latency requirement will be checked before the demand is finally served. Even compared with CPVNF and Best-Availability, Low-Latency scheme only reduces latency by about 4%, at traffic load of 500 Erlang. With increasing traffic load, average latency of each scheme increases almost linearly. This is because, as average data size of service demands increases, a proportional increment of both processing and transmission latencies is incurred.

D. Performance Comparison for Different Network Capacities

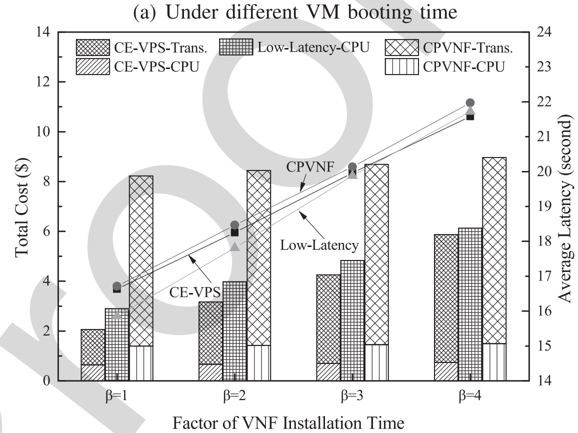
Higher network capacity may reduce the transmission latency and improve reuse of VNFs by multiple demands. Hence, we evaluate performance of different schemes for different network capacities in terms of cost and latency in Fig. 7. In GCP, with more CPU cores, VM can have a higher egress network capacity (see Section VI-A), and such scheme is denoted as “Changeable” in results. Other fixed network capacities, i.e., 20, 15, 10, 5, and 2.5 Gbps, are also considered.

From the results, we find that, with higher network capacity, total cost can be reduced for CE-VPS and CPVNF. Specifically, for CPVNF, CPU resource cost can be reduced by about 13% when network capacity increases from 2.5 Gbps to 5 Gbps, while for CE-VPS, reduction is about 6%. Moreover, cost decreases much slowly when network capacity increases from 10 Gbps to 20 Gbps, which implies that a network capacity of 10 Gbps is enough to guarantee quality of service. Note that CPVNF and CE-VPS with changeable network capacity achieve comparable (or even better) performance compared to that with fixed network capacity of 15 Gbps. This indicates the importance of adjusting network capacity flexibly on reducing transmission cost.

With respect to average latency, we find that Low-Latency remains on the same level with different network capacities. But latency can be reduced by about 11% for CPVNF and by about 6% for CE-VPS when network capacity increases from 2.5 Gbps to 5 Gbps. Performance improvement becomes unremarkable when network capacity is greater than 10 Gbps. The phenomenon indicates that it becomes a bottleneck when network capacity is very small, where service demand can suffer a similar magnitude of transmission latency to VNF processing latency. In this case, performance of both latency and CPU resource cost will deteriorate significantly.



(a) Under different VM booting time



(b) Under different VNF installation time

Fig. 8. Total cost (bars) and average latency (curves) of different schemes.

E. Performance Evaluation Under Different VM Booting Time and VNF Installation Time

To evaluate the effect of VBT and VIT on performance in terms of cost and latency, we run simulation under different parameters. Factors α and β lead to different times of initial VBT and VIT, respectively, e.g., $\alpha = 2$ represents VBT is 40ms (initial time is 20ms). The results are shown in Figs. 8(a) and 8(b).

We find that all schemes consume more CPU resources for increasing VBT, and the increment is more significant for Low-Latency. For CE-VPS, data transmission cost increases more remarkably for a longer booting time. This is because, to provision a demand with a strict latency requirement, an active VNF instance (even in a different zone) is preferred to be selected as booting a new VM will induce a significant latency. But, in CPVNF, available time of VMs and computing resource consumption are both considered, leading to a slight increase of data transmission cost, which is similar to CE-VPS.

It can also be found from Fig. 8(a) that service demands suffer a longer average latency when VBT increases, and performance of Low-Latency is significantly affected by VBT. Specifically, when $\alpha = 4$, CE-VPS achieves average latency close to Low-Latency.

Effect of VIT on performance for different schemes is shown in Fig. 8(b). We find that VIT has a more notable influence on performance than VBT. For CE-VPS, data transmission cost rises a lot when VIT becomes longer. Specifically, when $\beta = 4$, total cost of CE-VPS becomes very close to Low-Latency. In CPVNF, CPU resource cost almost remains the same while data transmission cost increases slightly with VIT being longer, since it tries to achieve a trade-off between CPU resource consumption and latency performance. Moreover, the effect of VIT on CPU resource cost is more vital for Low-Latency.

Average latency for the three schemes increases when β factor becomes larger, because for demands that are latency-insensitive, required VNFs are more likely to be executed in a VM with fewer CPU resource allocated. It should be noted that the low-latency advantage of Low-Latency over CE-VPS disappears when β equals 4.

Thus, we conclude that both VBT and VIT have significant impact on the performance in terms of CPU resource cost, data transmission cost, and average latency. Specifically, Low-Latency, for each service demand, sets up a new VM and initializes required VNF instances, and this affects negatively its performance, both in terms of cost and latency (VIT has more impact than VBT). CE-VPS, instead, minimizes the costs of CPU resource and data transmission by attempting to re-use existing VNF instances and to avoid data transmission among different zones. As CE-VPS also ensures that latency requirement is satisfied, a superior trade-off between latency performance and cost can be achieved.

As a whole, we show that re-using existing VM/VNF instances allows to more effectively satisfy latency requirements, especially for latency-sensitive applications, e.g., the emerging VR gaming. In turn, this indicates that it is desirable to have technologies for rapid VNF booting and to deploy VNFs in public clouds with short VBT.

VII. Conclusion

Cloud computing allows SPs to deploy VNFs into high-performance VMs in public cloud datacenters operated by CIPs. When deploying VNFs in cloud, the SP aims to minimize the cost paid to lease computing and networking resources, while satisfying diverse latency requirements of different service demands. The optimization problem addressed in this study, namely the "VNF placement and scheduling in public cloud networks (VPS-CD)", is different from other conventional versions of the VPS problem. In VPS-CD, we incorporate the impact of several realistic factors which are typically neglected in existing VPS solutions, e.g., VNF threading attributes, VM booting time, and VNF installation time. A solution to the VPS-CD problem has not been investigated before until now. In this study, a cost-efficient VPS-CD scheme is proposed, and to formulate the VPS-CD problem, a MILP and an efficient heuristic are designed for small-scale and large-scale networks, respectively. Our results confirm the importance of developing VNFs with short installation time and of using algorithms (such as VPS-CD) which promote the reutilization of existing VM/VNF instances. Compared to two baseline schemes, Best-Availability and Cost-Efficient Proactive VNF Placement (CPVNF), both total cost and latency can be reduced by CE-VPS. Also, a better trade-off between resource consumption and latency performance is achieved by CE-VPS when compared to a conventional Low-Latency scheme.

References

[1] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Comput. Commun.*, vol. 102, pp. 1–16, Apr. 2017.

[2] C. Donley, J. Berg, and M. Kloberdans, "Network function virtualization (NFV)," U.S. Patent 14 788 684, Jan. 7, 2016,

[3] G. ETSI, "Network functions virtualisation (NFV); Use cases," *V1*, vol. 1, p. 10, Dec. 2013.

[4] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-Art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[5] R. Yu, G. Xue, V. T. Kilari, and X. Zhang, "Network function virtualization in the multi-tenant cloud," *IEEE Netw.*, vol. 29, no. 3, pp. 42–47, May 2015.

[6] L. Gupta, M. Samaka, R. Jain, A. Erbad, D. Bhamare, and C. Metz, "COLAP: A predictive framework for service function chain placement in a multi-cloud environment," in *Proc. IEEE 7th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2017, pp. 1–9.

[7] *Google Cloud Platform*. [Online]. Available: https://cloud.google.com/44

[8] *Amazon EC2*. [Online]. Available: https://aws.amazon.com/ec2/

[9] (Jul. 2018). *2018 NFV Report Series Part 3: State of the VNF Ecosystem*.https://noteya.com/telco-systems-sdxcentral-2018-nfv-report-series-part%-3/

[10] H. Chen *et al.*, "MOSC: A method to assign the outsourcing of service function chain across multiple clouds," *Comput. Netw.*, vol. 133, pp. 166–182, Mar. 2018.

[11] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2014, pp. 7–13.

[12] T. Lin, Z. Zhou, M. Tornatore, and B. Mukherjee, "Demand-aware network function placement," *J. Lightw. Technol.*, vol. 34, no. 11, pp. 2590–2600, Jun. 1, 2016.

[13] R. Mijumbi, "Placement and scheduling of functions in network function virtualization," 2015, *arXiv:1512.00217*. [Online]. Available: http://arxiv.org/abs/1512.00217

[14] A. Gupta, M. Farhan Habib, U. Mandal, P. Chowdhury, M. Tornatore, and B. Mukherjee, "On service-chaining strategies using virtual network functions in operator networks," *Comput. Netw.*, vol. 133, pp. 1–16, Mar. 2018.

[15] J. Martins *et al.*, "ClickOS and the art of network function virtualization," in *Proc. 11th USENIX Conf. Networked Syst. Design Implement.*, 2014, pp. 459–473.

[16] A. Sheoran, X. Bu, L. Cao, P. Sharma, and S. Fahmy, "An empirical case for container-driven fine-grained VNF resource flexing," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2016, pp. 121–127.

[17] G. ETSI, "Network functions virtualisation (NFV): Architectural framework," *ETsI Gs NFV*, vol. 2, no. 2, p. 1, 2013.

[18] J. Gil Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[19] J. F. Riera, E. Escalona, J. Batalle, E. Grasa, and J. A. Garcia-Espin, "Virtual network function scheduling: Concept and challenges," in *Proc. Int. Conf. Smart Commun. Netw. Technol. (SaCoNeT)*, Jun. 2014, pp. 1–5.

[20] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Proc. 1st IEEE Conf. Netw. Softw. (NetSoft)*, Apr. 2015, pp. 1–9.

[21] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3746–3758, Sep. 2016.

[22] K. Yang, H. Zhang, and P. Hong, "Energy-aware service function placement for service function chaining in data centers," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[23] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 98–106.
[24] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.
[25] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2015, pp. 191–197.
[26] J. Anderson, H. Hu, U. Agarwal, C. Lowery, H. Li, and A. Apon, "Performance considerations of network functions virtualization using containers," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2016, pp. 1–7.
[27] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 13–24, Sep. 2012.
[28] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "CloudNaaS: A cloud networking platform for enterprise applications," in *Proc. 2nd ACM Symp. Cloud Comput. (SOCC)*, 2011, pp. 1–8.
[29] M. D. Ananth and R. Sharma, "Cost and performance analysis of network function virtualization based cloud systems," in *Proc. IEEE 7th Int. Advance Comput. Conf. (IACC)*, Jan. 2017, pp. 70–74.
[30] P. Bellavista *et al.*, "Virtual network function embedding in real cloud environments," *Comput. Netw.*, vol. 93, pp. 506–517, Dec. 2015.
[31] M. Dieye *et al.*, "CPVNF: Cost-efficient proactive VNF placement and chaining for value-added services in content delivery networks," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 2, pp. 774–786, Jun. 2018.
[32] T. Gao *et al.*, "Demand-adaptive VNF placement and scheduling with low latency in optical datacenter networks," in *Proc. 17th Int. Conf. Opt. Commun. Netw. (ICOCN)*, Feb. 2019, Art. no. 110480.
[33] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei, "Measuring the latency of cloud gaming systems," in *Proc. 19th ACM Int. Conf. Multimedia*, 2011, pp. 1269–1272.
[34] E. Steinbach, N. Farber, and B. Girod, "Adaptive playout for low latency video streaming," in *Proc. Int. Conf. Image Process.*, 2001, pp. 962–965.
[35] J. D. Touch and D. J. Farber, "An experiment in latency reduction," in *Proc. Conf. Comput. Commun.*, Jun. 1994, pp. 175–181.
[36] J. Y. Yen, "An algorithm for finding shortest routes from all source nodes to a given destination in general networks," *Quart. Appl. Math.*, vol. 27, no. 4, pp. 526–530, Jan. 1970.
[37] M. Kuzlu, M. Pipattanasomporn, and S. Rahman, "Communication network requirements for major smart grid applications in HAN, NAN and WAN," *Comput. Netw.*, vol. 67, pp. 74–88, Jul. 2014.
[38] *Google's Data Centers Grow So Fast it Has to Build its Own Networks*. [Online]. Available: https://www.computerworld.com/article/2937831/googles-data-centers-grow%-so-fast-it-has-to-build-its-own-networks.html
[39] T. Gao, W. Zou, X. Li, B. Guo, S. Huang, and B. Mukherjee, "Distributed sub-light-tree based multicast provisioning with shared protection in elastic optical datacenter networks," *Opt. Switching Netw.*, vol. 31, pp. 39–51, Jan. 2019.
[40] A. Gupta, B. Jaumard, M. Tornatore, and B. Mukherjee, "A scalable approach for service chain mapping with multiple SC instances in a wide-area network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 529–541, Mar. 2018.
[41] K. Argyraki *et al.*, "Can software routers scale?" in *Proc. ACM Workshop Program. Routers Extensible Services Tomorrow*, 2008, pp. 21–26.

Tao Gao received the B.S. degree in communication engineering from the Dalian University of Technology (DLUT) in 2010, and the Ph.D. degree from the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications (BUPT), in 2019. He was a Visiting Ph.D. Student with the Network Laboratory, University of California at Davis, Davis, CA, USA, from 2017 to 2018. His research interests include the survivability of optical networks and the virtualization of network functions.

Xin Li was born in Shandong, China, in 1986. He received the B.S. degree in communication engineering from the Shandong University of Science and Technology in 2008 and the Ph.D. degree in communication and information system from the Beijing University of Posts and Telecommunications (BUPT) in 2014. He is currently a Lecturer at BUPT. His research focuses on software-defined optical networking, elastic optical networks, datacenter networking, and optical network survivability.

Yu Wu received the B.E. degree from the Department of Information and Communication Engineering, Zhejiang University, China, in 2014, and the Ph.D. degree from the Department of Computer Science, University of California at Davis, Davis, CA, USA, in 2018. He currently works with Google Cloud focusing on software-defined network (SDN) research and development. During his Ph.D. study, he has authored 14 peer-reviewed conference papers and journal articles. His research interests include cloud and optical network resource management/optimization, green energy communication, and 5G emerging technologies. He was a recipient of the Best Paper Award granted by the IEEE Transmission, Access, and Optical Systems (TAOS) Technical Committee in IEEE GLOBECOM 2017.

Weixia Zou received the bachelor's degree from Tongji University in 1994, the master's degree from Shandong University in 2002, and the Ph.D. degree from the Beijing University of Posts Telecommunications (BUPT), China, in 2006. She is currently an Associate Professor with BUPT. Her current research focused on new technologies of short-range wireless communications and the electromagnetic compatibility (EMC).

Shanguo Huang (Member, IEEE) received the Ph.D. degree from the Beijing University of Posts Telecommunications (BUPT), Beijing, China, in 2006. He is currently a Professor with the State Key Laboratory of Information Photonics and Optical Communications (IPOC), and the Dean of the School of Sciences, BUPT. He has been actively undertaking several national projects, published three books and more than 150 journals and refereed conferences, and authorized 14 patents. His current research interests include the networks designing, planning, the traffic control and resource allocations, routing algorithms, and performance analysis. He was awarded the Beijing Higher Education Young Elite Teacher, the Beijing Nova Program, and the Program for New Century Excellent Talents in University from the Ministry of Education, from 2011 to 2013. AQ:5

Massimo Tornatore (Senior Member, IEEE) received the Ph.D. degree in information engineering from the Department of Electronics, Information and Bioengineering, Politecnico di Milano, Italy, in 2006. He is currently an Associate Professor with the Department of Electronics, Information and Bioengineering, Politecnico di Milano. He also holds an appointment as an Adjunct Professor with the Department of Computer Science, University of California at Davis, Davis, CA, USA. He has authored over 300 peer-reviewed conference papers and journal articles. His research interests include performance evaluation, optimization and design of communication networks (with an emphasis on the application of optical networking technologies), cloud computing, and energy-efficient networking. He was a co-recipient of 11 best-paper awards. He is a member of the Editorial Board of *Photonic Network Communications* (Springer), *Optical Switching and Networking* (Elsevier), IEEE COMMUNICATION SURVEYS AND TUTORIALS, and IEEE COMMUNICATION LETTERS.

Biswanath (Bis) Mukherjee (Fellow, IEEE) received the B.Tech. degree (Hons.) in electronics from the Indian Institute of Technology, Kharagpur, in 1980, and the Ph.D. degree in electrical engineering from the University of Washington, Seattle, in 1987.

He presented the first proposal/prototype for a network intrusion detection system in 1990; today it is a 10B/year industry. Also, in 1990, he proposed and prototyped the first filtering firewall, which is 8B/year business today. He led the proposal/prototype for the first dynamic bandwidth allocation algorithm in Ethernet-based fiber-to-the-home networks in 2002; today over 100 million units are deployed worldwide, including AT&T's Gigabit Fiber offering. In 2007, he provided the first proposal to integrate optical and wireless networks, forming the basis for the fronthaul of the upcoming 5G industry. He served a five-year term as a Founding Member of the Board of Directors of IPLocks, a Silicon Valley startup company (acquired by Fortinet). He was also a Founding Director of an optical startup, Optella, Inc. (acquired by Cosemi, Inc.) He has served on the Technical Advisory Board of over a dozen startup companies, including Teknovus (acquired by Broadcom). He is the Founder and the President of Ennetix, Inc., a Davis-based, SBIR-funded startup company specializing in AI-powered, application-centric, and network analytics for optimal user experience. He is also a Distinguished Professor of computer science with the University of California at Davis, Davis, CA, USA, where he has been for the past three decades. He was the author of the graduate-level textbook *Optical WDM Networks* (Springer, January 2006). He was the General Co-Chair of the IEEE/OSA Optical Fiber Communications (OFC) Conference in 2011, the TPC Co-Chair of OFC2009, and the TPC Chair of IEEE INFOCOM'96. He has supervised 77 Ph.D. students to completion. He is the Winner of the 2004 Distinguished Graduate Mentoring Award, the 2009 College of Engineering Outstanding Senior Faculty Award, and the 2016 UC International Community Building Award at UC Davis. He was the Co-Winner of the 2018 Charles Kao Award (named after Nobel Laureate and Fiber-Optic Pioneer Charles Kao) for the Best Paper in IEEE/OSA Journal of Optical Communications and Networks. He is also the Co-Winner of 12 other Best Paper Awards, mainly from IEEE conferences and journals; and also from the National Computer Security Conference. He was also the Winner of the IEEE Communications Society's Inaugural (2015) Outstanding Technical Achievement Award for pioneering work on shaping the optical networking area. He is an Editor of *Optical Networks* (Springer). He has served on eight journal editorial boards, most notably IEEE/ACM Transactions on Networking and IEEE Network; and also as a Guest Editor for IEEE Journal on Selected Areas in Communications, IEEE/OSA Journal of Lightwave Technology, Proceedings of the IEEE, and IEEE Communications.