Distributed Profitable Deployment of Network Services to Geo-distributed Edge Systems

Yi-Chia Chen and Li-Hsing Yen
Department of Computer Science, College of Computer Science, National Chiao Tung University, Hsinchu, Taiwan.
Email: {yijia1127.cs06g, lhyen}@nctu.edu.tw

***Abstract*—Recent advance of Network Function Virtualization (NFV) enables deployment of NFV-based network services on a cloud-base infrastructure. Meanwhile, edge computing provides cloud infrastructure on the the edge of access network to serve end user. This paper proposes a decentralized market-based framework that matches independent network service providers (NSPs) with independent edge service providers (ESPs) for possible deployments of network services with associated payments. The framework benefits all participants by guaranteeing that the results are profitable for all participants: no NSP or ESP can be better off by dropping out of the match result. Simulation results indicate that the proposed approach had the highest average number of deployed network services compared with counterparts.**

I. Introduction

Network functions such as firewall, deep packet inspection (DPI), intrusion detection system (IDS), and network address translator (NAT) in telecoms network are traditionally provided by dedicated hardware-based middleboxes. These middleboxes support specific network function and can hardly be reused for other purposes. This means deploying new network services often requires purchasing new hardware. For its significant capital and operational expenses, hardware-based implementation becomes an obstacle when telecom operators want to offer low-cost service to end users. As a remedy, European Telecommunications Standards Institute (ETSI) has proposed Network Function Virtualization (NFV) [1] which by utilizing virtualization technology consolidates network functions onto industry-standard servers, switches, and storage hardware. In NFV architecture, network functions are realized as software instances named virtual network functions (VNFs), which run on virtual machines (VMs) on top of commodity high volume servers. One promising use case of NFV is Service Function Chaining (SFC), which dynamically chains VNFs in a particular order to provide specific NFV-based network services to users. For example, a service chain may consist of an edge-router at the customer premises, followed by a series of network functions before reaching the service provider's core network.

SFC placement is to select a set of physical servers to place VNFs for a particular SFC. The placement should meet resource constraints of servers yet different placements may incur different operation costs. In the literature, many algorithms have been proposed for VNF or SFC placement in cloud data centers. These placements aim at resource utilization [2], [3], load balancing [3], and operation cost [4], [2]. As the placements are in centralized data centers, service latency of SFC is not a primary concern. However, if an SFC is to serve end users on the edge of access network, placing it in a data center that is distant from end users may introduce high service latency and is not suitable for latency-sensitive services. Examples of these services include those aiming to support autonomous driving, automated factory, or smart buildings. For these applications, edge computing [5] provides storage and computational resources close to end users, which enables processing end-user traffic at the edge of networks. With edge computing, telecoms operators or independent service providers could deploy and manage network services at the edge, providing latency-sensitive or context-aware NFV-based network service to end users. In fact, multi-access edge computing (MEC) is considered complementary to NFV [6], since both technologies are based on virtualized platforms.

In this paper, we consider a business model with two groups of participants in NFV and MEC environment. One group consists of *edge server providers* (ESPs) and the other group consists of *network service providers* (NSPs). Each ESP owns a set of geo-distributed edge servers with heterogeneous capabilities. ESP leases resources on edge servers to NSPs for VNF placement at the edge of the network. ESP collects revenues from and also pays expense incurred in resource lease. NSP leases resources from ESPs for SFC deployments with specific needs and constraints. NSP pays resource lease fee to ESP yet receives revenues from network service users outside the model. NSPs and ESPs all want to maximize their payoffs. However, NSPs maximize their payoffs by minimizing their payments while ESPs do so by maximizing the resource lease prices. Therefore, their objectives are conflicting. The goal of our study is to design a trading mechanism that benefits all participants. We propose a decentralized framework for ESPs and NSPs to check possible SFC deployment with resource lease between every pair of ESP and NSP. The check involves a placement of each VNF in the SFC to an edge servers in the ESP that meets all the constraints while minimizing overall operational expense. The minimal operational expense serves as the ask price of the ESP, based on which a bargaining process known as *matching auction* exercises between ESPs and NSPs for payment determination. We shall analyze whether an ESP or NSP can be better off by deviating from the result of the bargaining. We also conducted simulations to study the profits of NSPs and ESPs in various demand and supply situations.

The rest of this paper is organized as follows. Sec. II describes the system model and problem formulation. The following section presents the proposed mechanism with analysis. Sec. IV shows our simulation results. Sec. V concludes this paper.

II. Background

A. System Model

We assume n network services $\mathcal{N} = \{ns_1, ns_2, \ldots, ns_n\}$ to be deployed in the edge system. We denote the NSP of ns_i by nsp_i. Each network service ns_i consists of a chain of VNFs $\{vnf_{i,1}, vnf_{i,2}, \ldots, vnf_{i,n_i}\}$, where n_i is the number of VNFs in the chain. The NFV Orchestrator (NFVO) [7] of nsp_i generates and manages a network service descriptor for ns_i. In the descriptor, each VNF in the chain is described by a VNF descriptor which links to a VNF image and comes with a Virtual Deployment Unit (VDU) descriptor. The VDU specifies among other settings $vm_{i,j}$, the VM type for running $vnf_{i,j}$, and $L_{i,j}$, a set of locations where $vnf_{i,j}$ can be placed. For the sake of our problem, we characterize each $vnf_{i,j} \in ns_i$ by a pair $(vm_{i,j}, L_{i,j})$. The whole network service ns_i is also associated with latency constraint t_i, which is the maximum latency that is acceptable when running ns_i.

We assume m ESPs $\mathcal{P} = \{esp_1, esp_2, \ldots, esp_m\}$ in the edge system. Each ESP $esp_k \in \mathcal{P}$ accommodates NFV Management and Orchestration Architecture (NFV-MANO) [8] and owns a set of m_k geo-distributed edge servers $\mathcal{S}_k = \{s_{k,1}, s_{k,2}, \ldots, s_{k,m_k}\}$. The location of each server $s_{k,l} \in \mathcal{S}_k$ is denoted by $sl_{k,l}$. Let R be the set of all different types of physical resources (CPU, memory, storage, etc.) For each type of physical resource $r \in R$, edge server $s_{k,l} \in \mathcal{S}_k$ has a limited supply $C^r_{k,l}$. We characterize each edge server $s_{k,l}$ by a pair $(\{C^r_{k,l}\}_{r\in R}, sl_{k,l})$. The physical resources on edge server are virtualized into pool of VMs and managed by Virtual Infrastructure Manager (VIM), such as OpenStack, for dynamically VM provisioning [9].

A network service can only be served by one ESP (i.e., cannot be split and deployed to different ESPs) while an ESP can serve multiple network services subject to its capacity and the latency constraints associated with network services. To figure out which ESPs can serve ns_i, nsp_i sends a query to each $esp_k \in \mathcal{P}$ with the associated latency constraint t_i and $(vm_{i,j}, L_{i,j})$ for each VNF $vnf_{i,j} \in ns_i$. When an ESP esp_k receives the query, it checks for each $vnf_{i,j} \in ns_i$ whether there is some edge server $s_{k,l} \in \mathcal{S}_k$ in $L_{i,j}$ that has enough resources to host $vnf_{i,j}$. If the check is positive, esp_k estimates the expected latency $\theta_k(ns_i)$ if it serves ns_i to see if $\theta_k(ns_i) \le t_i$. If esp_k meets all the requirements, esp_k becomes an *eligible supplier* of ns_i and replies a quotation to nsp_i. The quotation includes information like $\theta_k(ns_i)$ and esp_k's ask price (minimal selling price) $\beta_{i,k}$.

Not all eligible suppliers of ns_i will be considered by nsp_i. Each NSP nsp_i has a maximal price v_i (i.e., its budget) that it is willing to pay for the deployment of ns_i. Only eligible suppliers with ask prices not exceeding v_i are *candidate suppliers* of ns_i. Among all candidate suppliers, the NSP selects one, say, esp_k, to submit its offer with a bid price p^k_i. The bid price should not exceed the NSP's budget but greater than or equal to the candidate supplier's ask price.

An ESP may receive offers from more than one NSPs. Normally the ESP will accept all the received offers. However, since ESP performs eligibility check for network services individually, it is possible that an ESP can accept the offers individually but not collectively due to its resource capacity. When this happens, the ESP selectively accepts some offers while rejecting all the others. If an offer is rejected, the NSP may in turn resubmit its offer with a raised bid price to the same ESP, or submit another offer (possibly with a different bid price) to another candidate supplier. The process repeats until the offer gets accepted or all candidate suppliers reject the NSP's offer yet the NSP cannot further raise its bid price (i.e., any raise would exceed its budget).

B. Problem Formulation

We model the deployment of network services to edge systems as a matching between the two parties. We use matrix $\mathbf{X} = [x^k_i]$ to denote the matching result, where $x^k_i \in \{0, 1\}$ indicates whether network service ns_i is deployed to ESP esp_k. The result also involves the setting of p^k_i, nsp_i's payment to ESP esp_k if $x^k_i = 1$.

The payments affects NSP's and ESP's objectives in two different ways. On one hand, nsp_i wants to maximize its payoff defined by $v_i - p^k_i$ from the deployment while minimizing the latency of the network service. Therefore, we define NSP's utility as the payoff per unit of latency. The objective of every $nsp_i \in \mathcal{N}$ is

$$\max_{x^k_i, p^k_i} \sum_{k=1}^{m} \left(x^k_i \cdot \frac{v_i - p^k_i}{\theta_k(ns_i)} \right). \tag{1}$$

On the other hand, an ESP's profit is the total revenue from the deployment of network services minus the associated operational cost. For a network service ns_i, the associated operational cost comes from the hosting of all VNFs $vnf_{i,j} \in ns_i$. Because different types of resources may incur different costs at different edge servers, the costs of hosting $vnf_{i,j}$ vary at different edge severs. Let $c^r_{k,l}$ be the unit cost of physical resource type r at edge server $s_{k,l}$ and variable $y^{k,l}_{i,j} \in \{0, 1\}$ indicate whether $s_{k,l}$ hosts $vnf_{i,j}$. The total operational cost incurred by ns_i on esp_k can be calculated as

$$\omega_k(ns_i) = \sum_{j=1}^{n_i} \sum_{l=1}^{m_k} \sum_{r\in R} \left(y^{k,l}_{i,j} \cdot d^r_{i,j} \cdot c^r_{k,l} \right), \tag{2}$$

where $d^r_{i,j}$ is the amount of type-r physical resource demanded by $vnf_{i,j}$. To make the deployment profitable, esp_k's ask price $\beta_{i,j}$ should not be less than $\omega_k(ns_i)$. Each esp_k's objective is to maximize its own profit, which can be expressed as

$$\max_{\{x_i, p^k_i\}} \sum_{i=1}^{n} \left(x^k_i \cdot \left(p^k_i - \omega_k(ns_i) \right) \right). \tag{3}$$

These objectives are subject to several constraints. First, the *satisfaction constraint* ensures that if esp_k serves ns_i, each

$\mathit{vnf}_{i,j} \in \mathit{ns}_i$ should be hosted by exactly one edge server $s_{k,l} \in \mathit{esp}_k$. Formally,

$$x_i^k = 1 \rightarrow \sum_{l=1}^{m_k} y_{i,j}^{k,l} = 1,\ \forall \mathit{vnf}_{i,j} \in \mathit{ns}_i. \tag{4}$$

Second, the *location constraint* ensures that edge server $s_{k,l}$ hosts $\mathit{vnf}_{i,j}$ only if the location of $s_{k,l}$ is in the set of locations $L_{i,j}$ associated with $\mathit{vnf}_{i,j}$. It can be expressed as

$$y_{i,j}^{k,l} \le \pi_{k,l}(\mathit{vnf}_{i,j}),\ \forall \mathit{vnf}_{i,j} \in \mathit{ns}_i,\ s_{k,l} \in \mathit{esp}_k, \tag{5}$$

where $\pi_{k,l}(\mathit{vnf}_{i,j}) \in \{0,1\}$ indicates whether the location of $s_{k,l}$ meets the demand of $\mathit{vnf}_{i,j}$. That is, $\pi_{k,l}(\mathit{vnf}_{i,j}) = 1$ if $sl_{k,l} \in L_{i,j}$ and $\pi_{k,l}(\mathit{vnf}_{i,j}) = 0$ otherwise.

The *capacity constraint* asserts that the total amount of physical resource of any type allocated to VNFs at any edge server $s_{k,l}$ cannot exceed $s_{k,l}$'s capacity. Formally,

$$\sum_{i=1}^{n}\sum_{j=1}^{n_i}(x_i^k \cdot y_{i,j}^{k,l} \cdot d_{i,j}^r) \le C_{k,l}^r,\ \forall r \in R,\ s_{k,l} \in \mathit{esp}_k,\ \mathit{esp}_k \in \mathcal{M}. \tag{6}$$

The constraint also suggests that an edge server can host multiple VNFs as long as it has enough capacity.

The third constraint is *latency constraint*:

$$\theta_k(\mathit{ns}_i) \le t_i, \quad \text{if } x_i^k = 1, \quad \forall \mathit{ns}_i \in \mathcal{N},\ \mathit{esp}_k \in \mathcal{M}. \tag{7}$$

The *budget constraint* limits the maximum bid price of each NSP nsp_i:

$$0 \le p_i^k \le v_i, \quad \forall \mathit{ns}_i \in \mathcal{N}. \tag{8}$$

Finally,

$$x_i^k \in \{0,1\},\ \forall \mathit{ns}_i \in \mathcal{N}, \tag{9}$$

and

$$\sum_{k=1}^{m} x_i^k \le 1,\ \forall \mathit{ns}_i \in \mathcal{N}. \tag{10}$$

Constraints (9) and (10) demand that a network service cannot be split and deployed to more than one ESPs.

Any setting of $\mathbf{X}$ corresponds to a definition of a one-to-many matching function $\mu\colon \mathcal{N} \cup \mathcal{P} \rightarrow \mathcal{N} \cup \mathcal{P}$ such that $\mu(\mathit{ns}_i) = \mathit{esp}_k$ if $x_i^k = 1$ ($\mu(\mathit{ns}_i) = \mathit{ns}_i$ if $x_i^k = 0$ for all k) and $\mu(\mathit{esp}_k) = \{\mathit{ns}_i \mid \mu(\mathit{ns}_i) = \mathit{esp}_k\}$ ($\mu(\mathit{esp}_k) = \{\mathit{esp}_k\}$ if $x_i^k = 0$ for all i). The result may not be the most preferred one for every ESP and network service. For every $\mathit{nsp}_i \in \mathcal{N}$, we define nsp_i's preference on $\mathcal{P} \cup \{\mathit{ns}_i\}$ such that $\mathit{esp}_j \prec_i \mathit{esp}_k$ if nsp_i prefers esp_j to esp_k. Moreover, $\mathit{ns}_i \prec_i \mathit{esp}_k$ if nsp_i prefers not matching with esp_k under all circumstances. Similarly, we define preference relation $\prec_k$ on $2^{\mathcal{N}} \cup \{\mathit{esp}_k\}$ for every ESP esp_k. Let $S, T \subseteq \mathcal{N}$ be two different sets of network services, we have $S \prec_k T$ if esp_k prefers S to T. Also, $\mathit{esp}_k \prec_k T$ for any $T \subseteq \mathcal{N}$ if and only if esp_k prefers no matching at all to being matched with T.

To guarantee stability, a matching should exclude *blocking individual* and *blocking pair* in the result. An ns_i (resp. esp_k) is a blocking individual if $\mathit{ns}_i \prec_i \mu(\mathit{ns}_i)$ (resp. $\mathit{esp}_k \prec_k \mu(\mathit{esp}_k)$). Intuitively, an NS or ESP is a blocking individual if it will be better off by deviating from the matching result (i.e., quitting). A pair $(\mathit{ns}_i, \mathit{esp}_k)$ is a blocking pair if $\mathit{ns}_i \notin \mu(\mathit{esp}_k)$, $\mathit{esp}_k \prec_i \mu(\mathit{ns}_i)$ and there exists a (possibly empty) set of services $S \subseteq \mu(\mathit{esp}_k)$ such that $S \cup \{\mathit{ns}_i\} \prec_k \mu(\mathit{esp}_k)$.

Definition 1: A matching result is stable if there is no blocking pairs or blocking individuals.

C. Related Work

The marriage problem [10] is a one-to-one matching in which a man/woman is matched to at most one woman/man. It has been used to model the resource allocation of wireless communications [11]. However, our problem is not a one-to-one matching since an ESP can serve more than one network services.

A one-to-many matching allows an agent from one side to be matched with several agents from the other side. A well-known example is the college admission problem [10], where a college can admit more than one students. In this problem, the maximum number of students that a college can admit is the college's quota, which is fixed. In contrast, the quota of an edge server in our problem is not fixed to a constant because the maximal number of VNFs that the server can host depends on the aggregated resource demand of VNFs and the server's resource capacity.

The deferred acceptance (DA) algorithm [10] guarantees stability for one-to-one matching problem. For one-to-many matching, DA may fail finding stable matching if the acceptors (e.g., the colleges) do not have fixed quotas. Xu and Li [12] proposed a revised one-to-many matching model for the assignment of jobs to machines. In this problem, each job has a size and each machine has a capacity. A machine can be assigned to several jobs as long as the total size of the jobs assigned to it does not exceed the capacity of the machine. The authors also proposed an algorithm for the job-machine problem which is extension to the DA algorithm. Compared to other matching models, our problem is most closely related to the job-machine model since network services are to jobs as ESPs are to machines.

Unlike traditional matching, *matching with transfers* involves monetary exchange. Shapley and Shubik [13] first described it as an assignment game, where buyers pay sellers in exchange for goods. A seller has a valuation on each item it sells, and each buyer also values the item it wants to buy. If a buyer values an item higher than the item seller, they may trade at a certain price. The matching with transfers model can be progressed in the form of a matching auction [14].

III. Proposed Mechanisms

We propose a two-layer matching mechanism for the deployment of network service to edge systems. The upper layer is the bargaining process between NSPs and ESPs. The lower layer is to allocate edge resources for VNF deployment within an ESP. In the bargaining process, when an ESP receives a query or an offer, the ESP uses the lower-layer mechanism to check the deployment eligibility and estimate associated operational expense.

A. Bargaining Between NSPs and ESPs

The bargaining process between NSPs and ESPs is for each NSP to determine to which EPS and with what bid price should it submit its offer for the placement of the associated network service. It is also for each ESP to decide whether it should accept a particular offer.

Our algorithm, named Bargaining Process in Deployment of Network Services to Edge System (BP-DNSES), is adapted from deferred acceptance (DA) algorithm [12]. Each NSP nsp_i values a candidate supplier esp_k by the following preference function:

$$\Phi_i(esp_k) = \begin{cases} \frac{v_i - p_i^k}{\theta_k(ns_i)}, & \text{if } \theta_k(ns_i) \leq t_i \\ -1, & \text{otherwise.} \end{cases} \tag{11}$$

This preference function is exactly nsp_i's utility when the latency constraint is met. We explicitly define $\Phi_i(ns_i) = 0$. Therefore, the negative preference value when $\theta_k(ns_i) > t_i$ implies that nsp_i prefers not matching with esp_k under any circumstances. Also, nsp_i will not submit an offer to esp_k with bid price $p_i^k > v_i$ due to negative preference value.

In each round of the bargaining process, each NSP nsp_i selects a candidate supplier esp_k with the highest preference value to submit its offer with a bid price $p_i^k \geq \beta_{i,k}$. When an ESP esp_k receives an offer from nsp_i, it executes the server allocation protocol introduced in the next subsection to see if it can host all VNFs in ns_i. It tentatively accepts the offer if it can. If esp_k cannot accept the current offer due to insufficient resource, it checks whether some tentatively accepted offers should be rejected to make room for this offer. It does so by defining a simple ranking on offers.

The ranking is based on a preference function $\Phi_k(ns_i)$ that ESP esp_k uses to value nsp_i's offer. A straightforward definition for $\Phi_k(\cdot)$ is esp_k's profit from the deployment of ns_i:

$$\Phi_k(ns_i) = p_i^k - \omega_k(ns_i). \tag{12}$$

The value of $\omega_k(ns_i)$ depends on the actual placement of each ns_i, which is quite complicated (subject to resource capacity, location, and latency constraints) and thus deferred to the lower layer. For a quick calculation, we use the following definition.

$$\Phi_k(ns_i) = \frac{p_i^k}{\sum_{j=1}^{n_i} \sum_{r \in R} \left(\alpha_k^r \cdot d_{i,j}^r \right)}, \tag{13}$$

where α_k^r is a weight that indicates the relative importance of physical resource type r among all types in esp_k with the property that $\sum_r \alpha_k^r = 1$. This version considers the revenue per unit of the weighted sum of all physical resource demanded by ns_i.

To make room for ns_i, esp_k checks all tentatively accepted offers in a non-decreasing order of their preference values. For each such offer, esp_k tentatively revokes its deployment and then executes the server allocation protocol to see if the amount of available resource after the revocation becomes high enough to deploy ns_i. If it is, esp_k accepts ns_i. Otherwise, esp_k proceeds to check the next offer. If esp_k still cannot accept ns_i even with the revocation of all tentatively accepted offers with preference values lower than ns_i, esp_k rejects ns_i and recovers all revoked offers. On the other hand, if esp_k accepts ns_i after the revoking of some offer ns_j, ns_j and all other offers that have been tentatively revoked are examined again one by one in the reverse order of the revoking to see if each such offer can actually be accepted after the deployment of ns_i. A tentatively revoked offer will be rejected if the acceptance is impossible.

When its offer is rejected, the NSP may either raise the bid price to $p_{i,k} + \delta$ and resubmit it to the same ESP or submit another offer to another ESP with a possibly different bid price. The decision depends on which offer has the highest preference value.

B. Server Allocation Protocol

The server allocation problem is for an ESP esp_k to arrange one hosting edge server $s_{k,l} \in esp_k$ for each VNF $vnf_{i,j} \in ns_i$. We say that $s_{k,l}$ matches with $vnf_{i,j}$ if $s_{k,l}$ hosts $vnf_{i,j}$. The match is valid if it meets both the capacity and location constraints. Different from matching in combinatorial optimization problem (e.g., bipartite matching), an edge server can match with multiple VNFs. For a valid match $(s_{k,l}, vnf_{i,j})$, the associated cost is

$$c(s_{k,l}, vnf_{i,j}) = \sum_{r \in R} \left(c_{k,l}^r \cdot d_{i,j}^r \right). \tag{14}$$

The minimum-cost server allocation problem is to find a valid match for each VNF in a given service chain such that the associated total cost is minimized.

Instead of solving the minimum-cost server allocation problem using a centralized algorithm, we use DA as a decentralized server allocation protocol. For each edge server $s_{k,l} \in esp_k$, it prefers $vnf_{i,j}$ to $vnf_{i,j'}$ if $c(s_{k,l}, vnf_{i,j}) < c(s_{k,l}, vnf_{i,j'})$. On the other hand, VNF $vnf_{i,j}$ prefers edge server $s_{k,l}$ to $s_{k,l'}$ if $c(s_{k,l}, vnf_{i,j}) < c(s_{k,l'}, vnf_{i,j})$.

C. Example and Stability Analysis

We consider a bargaining example with two network services ns_1, ns_2 and two ESPs esp_1, esp_2. We assume that all NSPs set set their increments of bid price to 30 dollars ($\delta = 30$). Table I shows the budgets of NSPs and ask prices of ESPs.

We assume that each NSP's initial bid price proposed to an ESP is the ESP's ask price. With that bid price, both nsp_1 and nsp_2 prefer esp_1 to esp_2 and thus propose to esp_1 with bid prices 60 and 100, respectively. Suppose that esp_1 does not have enough resource to accept both offers. Since esp_1's preference value on ns_1 (12.5) is less than that of ns_2 (16.67), esp_1 accepts nsp_2's offer and rejects nsp_1's.

In the second round, nsp_1 still prefers esp_1 to esp_2 even after raising p_1^1 to $60 + 30 = 90$ and thus resubmits its new offer to nsp_1. Although esp_1's bid price is still lower than esp_2's, this time esp_1' preference value on ns_1 (22.5) is higher than that on ns_2 (16.67). Therefore, esp_1 accepts nsp_1's new offer and rejects nsp_2's.

In the next round, nsp_2 attempts raising its bid price to 130 and still prefers esp_1 to esp_2. It resubmits its offer to esp_1 but

TABLE I: Budgets of NSPs and ask prices of ESPs

Network service	Budget	ESP	Ask price
ns_1	110	esp_1	60
		esp_2	105
ns_2	180	esp_1	100
		esp_2	150

still gets rejected because esp_1's preference value on ns_1 is still higher than that on ns_2 (21.67). After nsp_2 further raises p_2^1 to 160, it prefers an alternative offer with bid price 150 to esp_2. This offer is accepted by esp_2.

Consequently, ns_1 and ns_2 are deployed to esp_2 and esp_1 with payments 150 and 90, respectively. The profits of nsp_1 and nsp_2 are 20 and 30, respectively, and the profits of esp_1 and esp_2 are 30 and 0, respectively.

As mentioned, the quota of an edge server in our problem is not fixed to a constant. For this reason, our BP-DESNS algorithm does not guarantee a stable matching result. However, it still precludes the existence of blocking individuals. The reason is not difficult to see. A NSP will never propose to an ESP which gives the NSP a negative payoff. Similarly, an ESP will not accept an offer which gives it a negative profit. Therefore, no NSP or NSP will be better off by giving up its matching result.

IV. Numerical Results

TABLE II: Simulation Parameters

Parameter	Description	Default value
n	Number of network services	200
m	Number of ESPs	5
n_l	Number of served locations	5
$[n_v^l, n_v^u]$	The lower and upper bounds of the number of VNFs in a service chain	$[4, 10]$
$[n_l^l, n_l^u]$	The lower and upper bounds of the number of locations specified by a VNF	$[1, 3]$
$[t^l, t^u]$	The lower and upper bounds of latency constraint specified by a service chain	$[20, 100]$
δ	Bid increment	50
$\mu_c^n, \mu_m^n, \mu_s^n$	Means of CPU, memory, and storage capacities in an edge server	200, 2000, 4000
$\sigma_c^n, \sigma_m^n, \sigma_s^n$	SDs of CPU, memory, and storage capacities in an edge server	20, 200, 400
$\mu_c^c, \mu_m^c, \mu_s^c$	Mean unit costs of CPU, memory, and storage in an edge server	20, 10, 5
$\sigma_c^c, \sigma_m^c, \sigma_s^c$	SDs of CPU, memory, and storage unit costs in an edge server	5, 2, 1

Table II lists all simulation parameters. Each result was an average of 50 trials with a configuration. We used uniform distributions to generated the number of VNFs in a service chain, the number of locations specified by a VNF, and the latency constraint associated with a service chain. Only 70% service chains had latency constraint. The rest did not. We used Gaussian distributions to generate the capacities and unit costs of CUP, memory, and storage in an edge server. We provided the same set of VM types as that offered by Amazon EC2 in US West Region [15] which is shown in Table III. We generated the VM type of each VNF following the Zipf distribution [16] in which the probability of the Medium type is twice of that of the Large type, the probability of the Large type is also twice of that of the XLarge type, and so on.

TABLE III: VM Instance Types Offered by Amazon EC2–US West Region

	Medium	Large	XLarge	XXLarge
CPU	1	2	4	8
Memory (GB)	3.75	7.5	15	30
Storage (GB)	4	32	80	160

For the estimation of expected latency, we first randomly placed five disk-shaped service regions in a 100×100 km^2 area. The service regions each with radius 15 km did not overlap with each other. We then randomly placed one server for each ESP in each service region. Refer to Fig. 1 for the test topology. The latency between two edge servers was set to be proportional to the in-between distance.

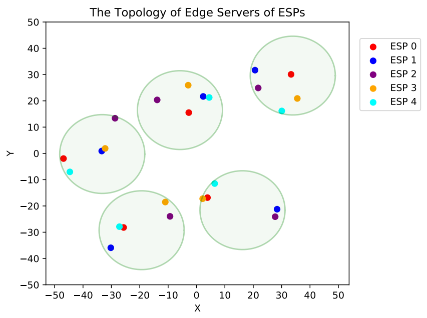
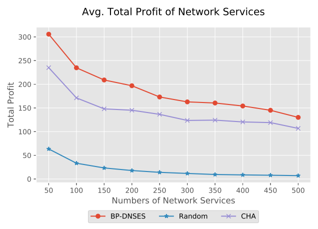


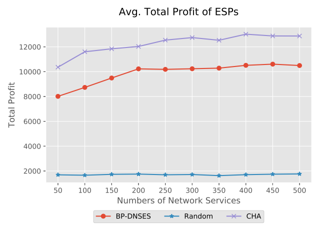
Fig. 1: Service regions and locations of edge servers

We measured the profits of ESPs and NSPs, and the number of served network services. For performance comparisons, we adapted two alternative one-to-many matching mechanisms. The first was capacitated house allocation (CHA) [17], which assumes one-sided (rather than two-sided) preferences. We modified CHA such that ESPs have preferences on network services while NSPs choose ESPs at random. The other alternative was Random, where each network service was randomly assigned to an ESP with bid price set to the mediate value between the ESP's ask price and the NSP's budget.

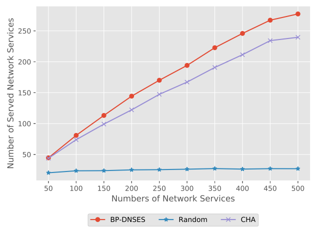
We varied the number of network services from 50 to 500 with an increment of 50. Fig. 2a shows how NSP's average profits changed with increased number of network services. Clearly, NSP's average profits declined in all mechanisms as the number of network services increased. This is because NSP's bid prices became higher when more network services contended for limited resource. Nevertheless, NSP's average profits using BP-DESNS were higher than both CHA and Random. Fig. 2b shows the average profits of ESPs with increasing number of network services. In contrast to Fig. 2a, ESP's average profits slightly increased using BP-DNSES or CHA when the number of network services increased. This can be justified as more competing requests will generally raise the final bid prices. Here CHA brought ESPs higher profits than BP-DNSES due to its setting of bid prices (the mediate values between the ESP's ask prices and the ESP's budgets). Concerning the average number of served network



(a) Avg. profit of NSPs

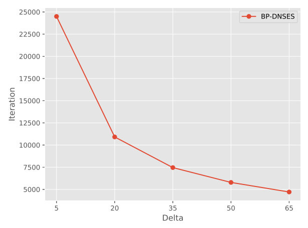


(b) Avg. profit of ESPs

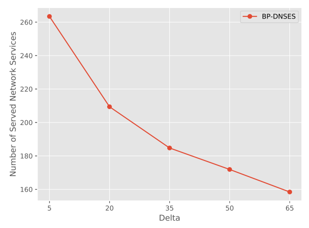


(c) Avg. number of served network services

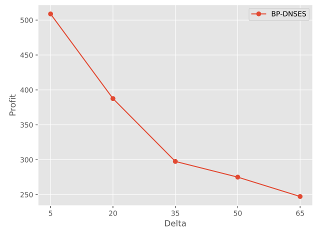
Fig. 2: Performance with increased number of network services



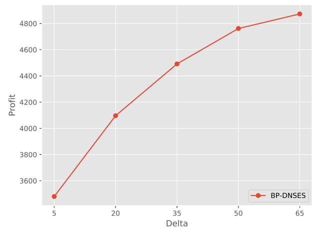
(a) Avg. number of iterations



(b) Avg. number of served network services



(c) Avg. profit of NSPs



(d) Avg. profit of ESPs

Fig. 3: Performance with respect to the setting of δ

services, Fig. 2c shows BP-DNSES outperformed both CHA and Random.

We also examined the effects of bid increment (δ) in BP-DESNS. Fig. 3a shows the average total number of iterations in a match auction versus δ, where less iterations were needed with larger δ. This was expected as a large δ decreases the number of times that NSPs can raise their bid prices. For the same reason, the number of served service chains also decreased as shown in Fig. 3b.

Figs. 3c and 3d show the average profits of NSPs and ESPs, respectively. We can see that, as δ increased, NSP's average profits decreased while ESP's average profits increased. This can be explained as a large bid increment usually means that buyers generally pay prices much higher than needed to get served. This increased seller's profits while decreasing buyer's.

V. Conclusions

We have proposed a decentralized framework for NSPs to negotiate with ESPs possible SFC deployments with associated payments. The negotiation involves a check for possible placement of each VNF in the SFC to an edge server in the ESP. The result is profitable for both sides and precludes the existence of blocking individuals, meaning that no NSP or NSP will be better off by dropping out of its matching result. Simulation results show that, compared with counterparts, the proposed approach slightly favors NSPs, but also makes the most average number of network services deployed to edge systems.

References

[1] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng *et al.*, "Network functions virtualisation: introductory white paper," in *SDN and OpenFlow World Congress*, 2012.

[2] A. Leivadeas, M. Falkner, I. Lambadaris, and G. Kesidis, "Optimal virtualized network function allocation for an SDN enabled cloud," *Computer Standards & Interfaces*, vol. 54, pp. 266–278, 2017.

[3] D. Dietrich, C. Papagianni, P. Papadimitriou, and J. S. Baras, "Network function placement on virtualized cellular cores," in *9th Int'l Conf. on Communication Systems and Networks*, 2017, pp. 259–266.

[4] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted NFV service chain deployment based on affiliation-aware vNF placement," in *Proc. IEEE Globecom Conf.*, 2016, pp. 1–6.

[5] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1657–1681, thirdquarter 2017.

[6] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[7] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider, "NFV and SDN: Key technology enablers for 5G networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2468–2478, 2017.

[8] V. Sciancalepore, F. Giust, K. Samdanis, and Z. Yousaf, "A double-tier MEC-NFV architecture: Design and optimisation," in *IEEE Conf. on Standards for Communications and Networking*, 2016, pp. 1–6.

[9] A. Corradi, M. Fanelli, and L. Foschini, "VM consolidation: A real case based on OpenStack cloud," *Future Generation Computer Systems*, vol. 32, pp. 118–127, 2014.

[10] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[11] S. Bayat, R. H. Louie, B. Vucetic, and Y. Li, "Dynamic decentralised algorithms for cognitive radio relay networks with multiple primary and secondary users utilising matching theory," *Transactions on Emerging Telecommunications Technologies*, vol. 24, no. 5, pp. 486–502, 2013.

[12] H. Xu and B. Li, "Anchor: A versatile and efficient framework for resource management in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1066–1076, 2012.

[13] L. S. Shapley and M. Shubik, "The assignment game I: The core," *International Journal of Game Theory*, vol. 1, no. 1, pp. 111–130, 1971.

[14] D. Fershtman and A. Pavan, "Matching auctions," Northwestern University, Working Paper 0144, 2017.

[15] L. Mashayekhy, M. M. Nejad, and D. Grosu, "A PTAS mechanism for provisioning and allocation of heterogeneous cloud resources," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 9, pp. 2386–2399, Sep. 2015.

[16] L. A. Adamic and B. A. Huberman, "Zipf's law and the Internet," *Glottometrics*, vol. 3, no. 1, pp. 143–150, 2002.

[17] D. F. Manlove and C. T. S. Sng, "Popular matchings in the capacitated house allocation problem," in *Algorithms – ESA 2006*, Y. Azar and T. Erlebach, Eds. Springer Berlin Heidelberg, 2006, pp. 492–503.