Multi-objective Embedding of Software-Defined Virtual Networks

Mohammad Khaksar Haghani , Bahador Bakhshi ,
Antonio Capone

PII: S0140-3664(17)31119-2
DOI: 10.1016/j.comcom.2018.07.017
Reference: COMCOM 5736

To appear in: *Computer Communications*

Received date: 19 October 2017
Revised date: 11 May 2018
Accepted date: 10 July 2018

Multi-objective Embedding of Software-Defined Virtual Networks

Mohammad Khaksar Haghani[a], Bahador Bakhshi[a,*], Antonio Capone[b]

[a] Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran
[b] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano
* Corresponding Author
Email Addresses: mkhd_1369@aut.ac.ir (Mohammad Khaksar Haghani), bbakhshi@aut.ac.ir (Bahador Bakhshi), capone@elet.polimi.it (Antonio Capone)

Abstract- Softwarization is the current trend of networking based on the success of technologies like Software Defined Networking (SDN) and Network Virtualization. Network as a Service (NaaS) is a new paradigm based on virtualization that enables customers to instantiate their *virtual* networks over a physical substrate network, mapping necessary resources by a Virtual Network Embedding (VNE) algorithm. Each VNE algorithm defines a resource allocation strategy of the NaaS provider, and determines its expenditures and revenues. Even though the problem of VNE has been widely investigated in recent years, virtualization in SDN introduces new challenges due to the new role of the controller and additional architectural constraints. In this paper, we investigate the VNE problem where both virtual and substrate networks are software defined. We propose a mathematical programming formulation that considers both the objectives of the NaaS provider (profit maximization) and the customers (switch-controller delay minimization). Proposing new design metrics (i.e., k-hop delay, correlation, and distance), we develop a heuristic algorithm, and prove its effectiveness through extensive simulations in the well-known VNE evaluation tool, ALEVIN, and comparisons with other algorithms and mathematical bounds.

Keywords: Software Defined Networking (SDN); Virtual Network Embedding (VNE); Network Virtualization; Multi-Objective Optimization; Network as Services (NaaS)

I. Introduction

In recent years, two technologies had major impact on computer networks, namely Software Define Networking (SDN) [1] and Network Virtualization [2]. SDN has introduced a new networking paradigm where the control plane is fully programmable and located in a logically centralized entity called the *controller*, while switches are simple packet forwarding devices (the data plane) where forwarding rules are programmed via an open interface e.g., ForCES [3], SoftRouter [4], and OpenFlow [5] in their flow tables. In software-defined networks, the switch-controller delay is a new important issue due to its impact on the network performance [6].

Network Virtualization or "Network as a Service" (NaaS) allows to flexibly organize network functions and to deploy multiple *virtual* networks on a shared physical *substrate* network, where functions and resources are logically separated.

Allocation of resources to virtual network (VN) requests, the key issue in NaaS, is commonly referred to as the *Virtual Network Embedding* (VNE) problem. In this problem, a set of VNs with given resource and topology requests should be mapped on a resource-limited substrate network optimizing specific efficiency objectives. A solution of the problem by a VNE algorithm is a resource allocation mechanism of the NaaS provider that directly impacts expenditures (since it determines substrate network resource consumption) and revenue (since it determines the VN requests that can be accommodated). Hence, from business point of view, efficient VNE algorithms are vital tools to manage the business of the NaaS providers.

The VNE problem can be divided into two sub-problems namely the *virtual nodes mapping* (VNoM) and *virtual links mapping* (VLiM) problems. These sub-problems can be dealt with separately, in a coordinated way, or jointly. More coordination leads to higher efficiency, however at the cost of significant computational complexity. Embedding virtual networks with nodes and links constraints has been proven to be a NP-Hard problem [7]. This has motivated researchers to focus on the development of heuristic methods able to provide good quality solutions for large instances in a reasonable time.

Isolating the virtual tenant networks is a major implementation issue in network virtualization. Recently, both the substrate and virtual networks are moving toward the SDN paradigm and use its flexibility and programmability to enhance network virtualization including the isolation issue. SDN hypervisor, e.g., FlowVisor [8], facilitates the isolation. It is a critical component for virtualizing software-defined networks that abstracts the underlying physical SDN network into multiple logically isolated virtual SDN networks [9]. It is located between the substrate network switches and the controllers of the *virtual* SDN networks, and divides the substrate network resources into virtual slices where each is under the control of a virtual network controller.

However, virtualization in SDN raises new issues in the VNE problem due to fundamental architectural differences of SDN from the traditional networks including: a) the centrality and importance of the controller which makes its placement an important problem [6], b) the constraints of switches on the number of entries in the forwarding table, c) the critical role of switch-controller delay, and d) virtualization by hypervisors whose location is vital in network performance. Therefore, whereas the VNE problem in the traditional networks has been widely studied and various approaches has been proposed [10], they are not directly applicable for the VNE problem in SDN.

Recently, a few attempts to tackle the VNE problem in SDN have been done, although with some limitations. In [11], only the VLiM problem is investigated and the stages of nodes mapping and controller placement are not taken into consideration. In [12] and [13], the placement of controller is not considered and only the problems of embedding nodes and links are addressed. In [14], the limitations of the resources of the substrate network are not considered and no coordination between the mapping of nodes and links is made.

In this paper, we investigate the problem of embedding a set of virtual SDN-based networks in a SDN-based substrate network where each virtual node/link is mapped on a physical node/path in the substrate network. And, by assuming that a distributed hypervisor is used in the substrate network, the controller of each VN is placed on a hypervisor instance in the substrate network. We consider both the objective of the NaaS provider to maximize its profit and the objective of customers to minimize the switch-controller delay in the VNs. We decompose the problem into two sub-problems and propose Mixed Integer Linear Programming (MILP) models for each of them.

Even if the MILP formulations provide interesting results on the performance bounds, they cannot be used to solve the problem in short time and/or on large instances. Therefore, we propose a novel heuristic algorithm that considers the unique characteristics of the software-define networks, such as the constraints on the number of entries in the flow table and also the location of the controller and hypervisors. The proposed solution efficiently coordinates controller placement, nodes mapping, and also links mapping. More precisely, the main contributions of this paper include:

- formulating and decomposing the problem of the virtual network embedding in SDN that aims to maximize the business profit of NaaS provider and minimizing the delay between switches and controller of the embedded virtual networks;
- development of appropriate design metrics for VNE algorithms in SDN, including the concept of *correlation* between virtual nodes for node ranking, the *k-hop delay* concept for controller placement, and the *distance* metric to minimize virtual link resource consumption;
- development a heuristic algorithm for VNE in SDN, SVE[1], that is based on the design metrics, and for the first time, coordinates the stages of controller placement, nodes mapping, and links mapping;

[1] SVE stands for **S**DN **V**irtual network **E**mbedding.

- extension of the well-known VNE evaluation tool, ALEVIN [15], to support SDN-based virtual and substrate networks, which is used to evaluate the proposed algorithm in comparison to previous work in different scenarios.

The rest of the paper is organized as follows. Section 2 provides an overview of the related researches. In Section 3, the system model and formulation of the VNE in SDN problem as optimization models are presented. In Section 4, we develop the SVE algorithm for solving the embedding problem. The results obtained from the simulations are presented in Section 5; and in the last section, the future directions are discussed.

II. Previous Work

Virtual network embedding is the key issue of resource allocation in NaaS, which is composed of the VNoM and VLiM sub-problems. The existing VNE approaches can be categorized into coordinated, uncoordinated, and joint groups in term of the interaction between the sub-problems. In the uncoordinated methods [16, 17], VNoM and VLiM are solved in two separated and uncoordinated stages. In contrast, in the coordinated approaches, there is a coordination between the sub-problems [18, 19]. In the joint approaches, both node and link mappings are performed jointly that increases the efficiency of the embedding at the significant cost of computational complexity.

In [17], a two-stage node mapping scheme was proposed by means of resources migration to improve resource usage and acceptance ratio of VN requests. In [16], to maximize the acceptance ratio, it is aimed to reduce bottlenecks in the substrate network by a greedy algorithm for VNoM that maps virtual nodes on the substrate nodes with the maximum available resources.

A recursive algorithm named VT-Planner was proposed in [18], which coordinates between VNoM and VLiM by minimizing the link pressure index in the node mapping stage. In [19], coordination between these two sub-problems was formulated as a MIP model; and to tackle its complexity, two rounding techniques, namely deterministic and random, were used.

Even though the VNE problem in traditional networks has been widely investigated [2, 10], the proposed approaches cannot be directly applied to VNE in SDN because of the fundamental different characteristics of SDN. First, due to the crucial role of the controller in SDN, its placement is an important issue. In [6], the controller placement problem in a given (not necessarily virtual) SDN network, and its effect on the switch-controller delay, as an important factor in the network performance, was studied. However, it is not considered in the traditional VNE algorithms. It must be noted that network (nodes and links) embedding via the traditional VNE algorithms and then placing the controller in the embedded network, which are conducted in two separated stages, is not a feasible/efficient solution for the VNE in SDN problem. Because node mapping should be coordinated with controller placement in order to satisfy the required switch-controller delay in the embedded virtual network. Neither the traditional VNE algorithms nor the controller placement methods consider this coordination.

Second, the size of flow table in SDN switches, as a new kind of substrate network resources, should also be managed. Finally, virtualization technologies in SDN are different from the traditional networks. In SDN, hypervisor has the responsibility of virtualizing the substrate network. It sits between the tenant SDN controllers and their respective virtual SDN networks and processes control traffic exchanged between them. Since switch-controller communication takes place through the hypervisor, its placement also impacts the performance of virtual networks. For scalability, performance and fault tolerance reasons, a distributed hypervisor can be used in the substrate network wherein multiple instances of the controller are installed in different locations. The problem of obtaining the required number and locations of the hypervisor instances, kwon as hypervisor placement problem (HPP), was studied in [20] and [21]. These papers assumed that the mapping of virtual nodes and virtual links and also the placement of the virtual SDN controllers are given and aimed to minimize the switch-controller latency through the solution of HPP. In contrary, in this paper, we propose a solution for embedding virtual SDN networks, composed of node and link mapping and controller placement stages in a coordinated manner, which not only minimizes the latency, but also maximizes the profit of the NaaS provider. In this approach, the location of hypervisors is determined by the placement of the virtual controllers, i.e., an instance of the hypervisor is installed on a substrate node if at least one virtual controller is mapped on it.

Recently, few studies have studied the VNE problem in SDN including [11], [12], [13], and [14]. In [12], the embedding problem was modeled as an ILP problem and a heuristic algorithm was proposed to solve the problem in large instances. The algorithm minimizes the link bandwidth consumption and the number of used substrate nodes. Whereas the solution was proposed for SDN, the controller placement problem is not considered, and also there is no effort to minimize the switch-controller delay.

In [11], the VLiM problem was formulated as a MILP model and a new algorithm named VLM was developed. VLM maps virtual links according to available bandwidth of physical links. It attempts to maximize the number of accepted links while minimizing substrate resources consumption. In [11], only VLiM is solved and the VNoM problem is neglected. Moreover, the controller placement, switch-controller delay, and flow table limitation are not considered. In [13], a MIP formulation, for a coordinated node and link mapping was proposed. The aim in this work is to maximize the revenue while minimizing resource consumption. The constraints taken into account are CPU capacity of nodes and the bandwidth on links. Moreover the controller placement is not considered.

The closest work to this paper, i.e., [14], takes controller placement into account. In that paper, two objectives are considered, namely balancing stress on substrate resources and minimizing the switch-controller delay. To determine the stress on a substrate node, in addition to the number of mapped virtual nodes on it, the computational power and the flow table space needed to handle the virtual links go through the node are also considered. In that paper, two algorithms named SBE[1] and DME[2] were developed. SBE focuses on balancing the stress on substrate resources while keeping the switch-controller delay within a given bound. On the other hand, DME minimizes the switch-controller delay while limiting the stress on substrate nodes and links. In comparison to other works, the SBE and DME algorithms are more applicable to VNE in SDN; however, they have considerable drawbacks. In these algorithms, the constraints of the capacity of the substrate resources, especially the size of switches' flow table, are not considered. Besides, node mapping, link mapping, and controller placement are performed in three separated and *uncoordinated* steps. Finally, the SBE and DME algorithms do not directly consider the business profit of the NaaS provider.

In summary, there is a considerable research gap in the problem of embedding software-defined virtual networks. In this paper, we formulate and solve the multi-objective VNE in SDN to maximize the profit of NaaS provider and minimize the switch-controller delay. We consider the special constraints of SDN in addition to traditional networks. More importantly, we coordinate the node mapping, link mapping, and the controller placement stages in our solution. Moreover, assuming using of a distributed hypervisor, we also determine the location of the instances of the hypervisor.

III. System Model and Problem Statement

In this section, first, the assumptions made in this paper are clarified. Then the abstract model of the substrate network and VN requests are explained. Finally, the problem of multi-objective embedding of software-defined virtual network is decomposed and formulated as two MILP optimization models. The notations used in this paper are summarized in Table I.

[1] **S**tress-**B**alancing **E**mbedding
[2] **D**elay-**M**inimizing **E**mbedding

A. Assumptions

In this paper, we study the off-line version of the VNE problem wherein all VN requests are known and given at the beginning. Each VN request can only be mapped on a subset of substrate nodes[1], if there are enough resources, i.e., CPU, flow table and bandwidth, in the subset, this request is accepted, otherwise it is rejected.

Table I. Notations

Notation	Description
V^s	The set of substrate network nodes
E^s	The set of substrate network links
C^s	The set of substrate network nodes capable to host VN's controller and substrate hypervisor instances
c_s	The CPU capacity of substrate node s
f_s	The flow table capacity of substrate node s
$b_{(s,t)}$	The bandwidth capacity of substrate link (s,t)
$d_{(s,t)}$	The minimum delay between substrate nodes s and t
V_i^v	The set of nodes of i^{th} VN request
E_i^v	The set of links of i^{th} VN request
$\mathrm{V_{i,s}}$	The subset of substrate nodes that i^{th} VN can be mapped on them
$\mathrm{F_t}$	The subset of substrate nodes that delay between them and node t is greater than r.
$\mathrm{c_v^i}$	The required CPU capacity for $v \in V_i^v$
$\mathrm{f_v^i}$	The required flow table capacity for $v \in V_i^v$
$\mathrm{b_{(u,v)}^i}$	The required bandwidth capacity for $(u,v) \in E_i^v$
D	The set of VN requests
$\mathrm{n_N(s)}$	The number of virtual nodes mapped on $s \in V^s$
$\mathrm{n_L(s)}$	The number of virtual links go through $s \in V^s$
$\mathrm{T(s,t)}$	The set of virtual links mapped on $(s,t) \in E^s$
r	The upper limit of the switch-controller delay
α_1	Revenue of allocating a unit of CPU for a customer
α_2	Revenue of allocating a unit of bandwidth for customer
β	Cost of using a unit of bandwidth of substrate links

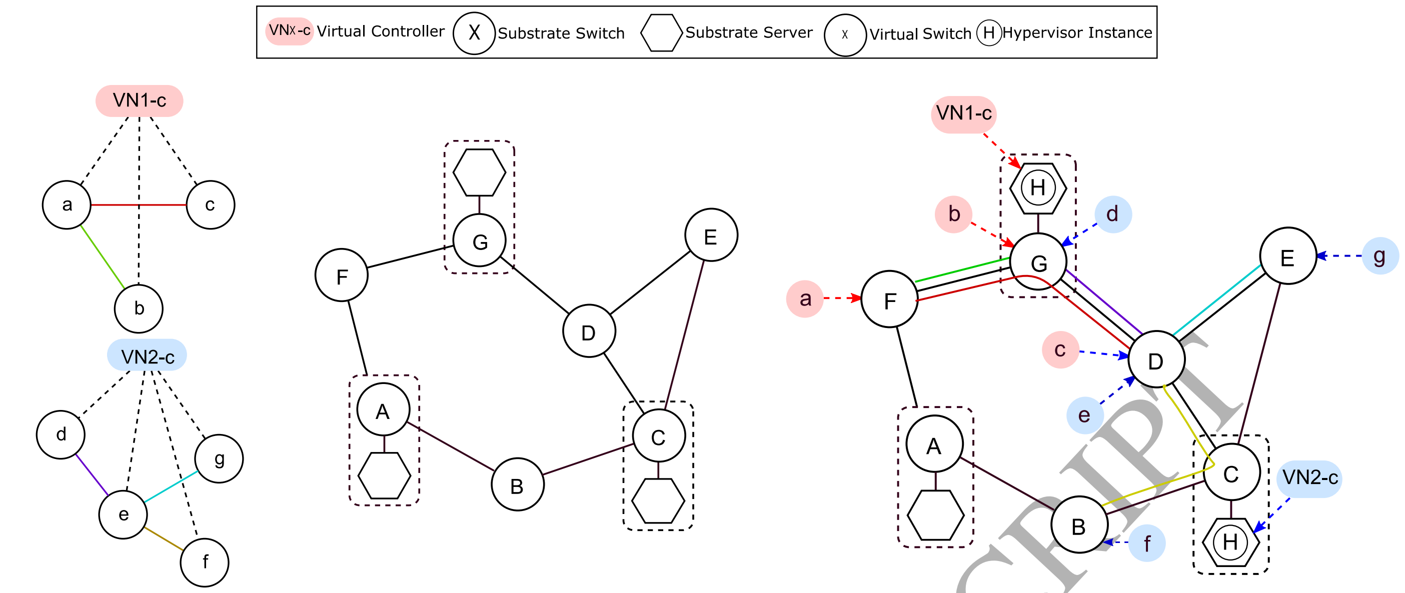
Each virtual node of a given VN is mapped on only one substrate node, and two virtual nodes of a VN are not mapped on the same substrate node.

To provide virtualization functionality in SDN, different hypervisor architectures were introduced in [21]. In this article, we assume that the NaaS provider uses the distributed hypervisor to isolate tenants' virtual networks.

In this architecture, multiple hypervisor instances are distributed over several locations in the network, and the substrate network switches support the multiple-controller feature2, and consequently can be controlled by multiple hypervisor instances3. We assume that in the physical location of a subset of substrate nodes, there is a server in addition to the switch. In the case of mapping of at least a virtual SDN controller to these nodes, an instance of the distributed hypervisor is installed on the server to host the mapped virtual controller(s). It is assumed that there is only one controller per virtual SDN.

The architectures of the virtual networks and the substrate network, based on these assumptions, are illustrated in Fig. 1(a) and Fig. 1(b), respectively.

[1] The set considers various constraints, e.g., geographical constraint.
[2] This feature was introduced in version 1.5 of the OpenFlow protocol.
[3] The architecture is identical to the "Distributed Network Hypervisor Architecture for Multi-Controller SDN Switches," in proposed in [21]. For more details about the architecture, please see. Fig. 1(c) and Section III.C.



(a) Two virtual networks (b) Substrate network (c) Embedding of the VNs in the substrate network

Fig. 1. An illustration of the architecture of virtual networks, the architecture of the substrate network, and mapping of the VNs in the substrate network. The controllers of both VN requests are mapped on node G; hence, an instance of the distributed hypervisor is installed on the server to host the virtual controllers.

As it shown, in the virtual networks, there is a controller that is connected to each switch via a dedicated link. In the illustrated substrate network, in nodes A, C, and G, there is also a server, depicted by hexagonal, besides the switch. Fig. 1(c) shows a mapping of the VNRs in the substrate network. The mappings of the nodes and links are depicted by dashed arrows. In this example, both the virtual controllers are mapped on node G. Therefore, an instance of the hypervisor is installed on the server in this node.

A. Substrate Network Model

The substrate network is represented by a directed graph $G^s = (V^s, E^s, C^s)$ where V^s is the set of substrate nodes, E^s is the set of substrate network links, and $C^s \subseteq V^s$ is a subset of the substrate nodes which are capable to host a hypervisor instance and consequently virtual SDNs' controllers, i.e., in addition to switch, there is a server in these nodes, e.g., nodes A, C, and G in Fig. 1(b). For each substrate node $s \in V^s$, the CPU and flow table capacities[1] are respectively denoted by c_s and f_s. The substrate link between nodes s and t has a bandwidth capacity which is indicated by $b_{(s,t)}$. The delay from node s to node t via the shortest (minimum delay) path between them is denoted by $d_{(s,t)}$. For each $t \in C^s$, we define its forbidden set as $F_t = \{\forall s \in V^s \mid d(t,s) > r\}$ where r is the maximum acceptable switch-controller delay in the VNs. If the controller of a given VN is mapped on t, its respective switches cannot be mapped on $s \in F_t$.

The business model of the NaaS provider who is the owner of the substrate network is as follows. The revenue is generated by accepting VN requests and allocating the required resources. It is proportional to the amount of the requested resources; more specifically, the revenue by allocating a unit of CPU and bandwidth for customers is α_1 and α_2 respectively. Regarding the expenditure, it is assumed that the substrate switches and servers are the provider's assets and installed in physical locations owned by the provider. Hence, allocating resources on the substrate nodes does not impose any cost for the NaaS provider. In contrast, bandwidth allocation on the substrate links is costly since

[1] In this paper, we focus on substrate switches resources and don't consider the capacity constraint of the servers.

it is assumed that the NaaS provider leases the links between substrate nodes from another carrier [22][1]. In summary, bandwidth allocation on the substrate links is the only cost that the NaaS provider should pay which is β per unit of bandwidth.

B. Virtual Network Request Model

The i^{th} VN request is represented by a graph $G_i^v = (V_i^v, E_i^v, V_{i,s})$; where V_i^v is the set of the nodes of the request, E_i^v is the set of the links, and $V_{i,s} \subseteq V^s$ is a subset of substrate nodes that can be used for mapping $v \in V_i^v$. The required CPU and flow table capacities of node $v \in V_i^v$ are denoted by f_v^i and c_v^i, respectively. The requested bandwidth on link $(u, v) \in E_i^v$ is denoted by $b_{(u,v)}^i$. The set D contains all VN requests.

As depicted in Fig. 1(a), in each virtual network, there is a direct link between each switch and the controller for the southbound communications (i.e., OpenFlow protocol). Its delay must be less than a given parameter r. More precisely, assuming that the controller of G_i^v is mapped on substrate node t, virtual node $v \in V_i^v$ must not be mapped on the substrate nodes $s \in F_t$ since the minimum delay between s and t is more than r.

C. Formulation of VNE in SDN

In this section, we formulate the problem of multi-objective embedding of software-defined virtual networks. For a given substrate network G^s, a set D of VN requests, and the maximum acceptable switch-controller delay r, the objective is to accept a set of requests that maximizes the profit of the NaaS provider (according to the aforementioned business model) while minimizing the switch-controller delay in the accepted VNs. Even though the maximum of the delay is bounded by parameter r, its minimization is crucial for the timely response of the controller to switch that impacts the performance of the virtual SDNs. This two-fold objective simultaneously takes the goals of the NaaS provider and customers into account.

This embedding problem can be formulated as a single *multi-objective* MILP optimization model. However, that leads to a complicated model that even cannot be solved for small instances because of the coupling between decision variables in the objective function. In the following, we decompose the multi-objective MILP into two sub-problems.

The VNE algorithm is the NaaS provider's tool to maximize its profit. In short-term, the goal is achieved by satisfying the QoS requirements of the accepted requests. In long-term, it is influenced by customers' QoE that depends on the performance of the embedded virtual network; and it is determined by the switch-controller delay. Therefore, minimizing the delay not only is the objective of the customers but also boosts the NaaS provider's profit. Accordingly, the NaaS provider should consider minimizing the delay in the VNE algorithm; however, since its effect is long-term and indirect, the direct short-term profit maximization is prioritized over the delay minimization. Based on this fact, in the following, the problem is formulated in two stages. At the first step, we formulate maximizing the profit while maintaining QoS requirements of accommodated requests. Then, in the second stage, to improve the performance of accepted requests, we formulate minimizing the maximum switch-controller delay while maintaining the maximum achievable profit determined by the first sub-problem.

The following decision variables are used to develop the optimization models:

- x_i is a binary variable that is equal to 1 if G_i^v is accepted; otherwise it is 0.
- $x_{v,s}^i$ is a binary variable that is equal to 1 if virtual node $v \in V_i^v$ is mapped on substrate node $s \in V^s$; otherwise, it is 0.

[1] In practical cases, establishment of links between physical locations needs authorities which are given to a limited number of carrier companies.

- $y^i_{(u,v),(s,t)}$ is a binary variable that is equal to 1 if link $(s,t) \in E^s$ belongs to the path that link $(u,v) \in E^v_i$ is mapped on; otherwise, it is 0.
- p^i_t is a binary variable that is equal to 1 if the controller of G^v_i is mapped on $t \in V^s$; otherwise, it is 0.
- m^i_s is a binary variable that is equal to 1 if a $v \in V^v_i$ is mapped on node $s \in V^s$; otherwise, it is 0.

In the following, the objective function and constraints are formulated and then the important notes about them are clarified.

In the first model, the goal is to maximize the profit; thus, the objective function is

$$P^{\max} = rev - cost, \tag{1}$$

where rev and $cost$ are the revenue and cost of embedding, respectively. According to the aforementioned business model, they are as follows:

$$rev = \alpha_1 \sum_{G^v_i \in D} \sum_{v \in V^v_i} c^i_v \cdot x_i + \alpha_2 \sum_{G^v_i \in D} \sum_{(u,v) \in E^v_i} b^i_{(u,v)} \cdot x_i \tag{2}$$

$$cost = \beta \sum_{(s,t) \in E^s} \sum_{G^v_i \in D} \sum_{(u,v) \in E^v_i} b^i_{(u,v)} \cdot y^i_{(u,v),(s,t)} \tag{3}$$

The constraints of the problem are the node and link capacity constraints, virtual node and link mapping, and the switch-controller delay constraints, which are formulated as follows.

The constraints on the capacity of substrate node's CPU and flow table are (4) and (5).

$$\sum_{G^v_i \in D} \sum_{v \in V^v_i} c^i_v \cdot x^i_{v,s} \le c_s \qquad \forall s \in V^s \tag{4}$$

$$\sum_{G^v_i \in D} \sum_{v \in V^v_i} f^i_v \cdot x^i_{v,s} \le f_s \qquad \forall s \in V^s \tag{5}$$

The substrate link's bandwidth constraint is

$$\sum_{G^v_i \in D} \sum_{(u,v) \in E^v_i} b^i_{(u,v)} \cdot y^i_{(u,v),(s,t)} \le b_{(s,t)} \qquad \forall (s,t) \in E^s \tag{6}$$

The constraint to map a virtual link on a path in substrate network is (7).

$$\sum_{(s,t) \in E^s} y^i_{(u,v),(s,t)} - \sum_{(t,s) \in E^s} y^i_{(u,v),(t,s)} = x^i_{u,s} - x^i_{v,s} \tag{7}$$

$$\forall G^v_i \in D, \forall (u,v) \in E^v_i, \forall s \in V_{i,s}$$

The constraints for the relations between the variables are

$$\sum_{s \in V_{i,s}} x^i_{v,s} = x_i \qquad \forall G^v_i \in D, \forall v \in V^v_i \tag{8}$$

$$\sum_{v \in V^v_i} x^i_{v,s} \le x_i \qquad \forall G^v_i \in D, \forall s \in V_{i,s} \tag{9}$$

$$m^i_s = \sum_{v \in V^v_i} x^i_{v,s} \qquad \forall G^v_i \in D, \forall s \in V_{i,s} \tag{10}$$

$$\sum_{t \in C^s} p_t^i = x_i \qquad \forall G_i^v \in D \qquad (11)$$

The following equation formulates the maximum tolerable delay between switches and controller,

$$p_t^i + m_s^i \leq 1 \qquad (12)$$

$$\forall G_i^v \in D, \forall t \in C^s, \forall s \in F_t$$

Finally the domain constraints are as follows.

$$x_i \in \{0,1\} \qquad \forall G_i^v \in D \qquad (13)$$

$$x_{v,s}^i \in \{0,1\} \qquad \forall G_i^v \in D, \forall v \in V_i^v, \forall s \in V_{i,s} \qquad (14)$$

$$y_{(u,v),(s,t)}^i \in \{0,1\} \qquad \forall G_i^v \in D, \forall (u,v) \in E_i^v, \forall (s,t) \in E^s \qquad (15)$$

$$p_s^i, m_s^i \in \{0,1\} \qquad \forall G_i^v \in D, \forall s \in V^s \qquad (16)$$

The following notes about these equations need to be clarified:

- Equations (1), (2), and (3) define the objective function. It is formulated based on the explained business model that maximizes the profit obtained from embedding VN requests in cost of using the carrier links.
- Substrate network resource constraints are formulated by (4), (5), and (6). As mentioned, in this paper, we don't consider the resources needed for the controllers, and moreover, since the volume of the southbound communication traffic is negligible in comparison to data traffic, it is not considered in the formulation.
- Constraint (7) guarantees that the virtual link $(u, v) \in E_i^v$ is mapped on a path between the substrate nodes that u and v are mapped on them.
- The constraint (8) guarantees that in the case of accepting G_i^v, each virtual node $v \in V_i^v$ is mapped on a single substrate node.
- The constraint (9) guarantees that for each accepted request, two virtual nodes are not mapped on the same substrate node.
- The constraint (10) indicates that if node $s \in V_{i,s}$ is used for mapping a node of G_i^v.
- The constraint (11) guarantees that if G_i^v is accepted, its controller must be mapped on a substrate network node.
- The constraint (12) guarantees that if the controller of G_i^v is mapped on $t \in V^s$, i.e., $p_t^i = 1$, and delay between s and t exceeds r, i.e., $s \in F_t$, then none of the switches of the request can be mapped on s, i.e., m_s^i must be zero. In other words, this constraint implies that the switches of G_i^v can only be mapped on a nodes that satisfy the switch-controller delay constraint.

In summary, the optimization model of the first stage is as follows.

maximize (1)

s. t. (2) – (16)

As explained, in the second stage, the objective is to minimize the maximum switch-controller delay while maintaining the maximum obtainable profit. Therefore, in the second stage, the maximum profit obtained at the first stage ($P^{\max}$) is added as the following constraint.

$$rev - cost \geq P^{\max} \qquad (17)$$

All the constraints of the first stage model should also be satisfied in this model. Moreover, the following constraint is added to limit the maximum switch-controller delay by ε which is minimized in the objective function.

$$(p_t^i + m_s^i - 1)d(s,t) \leq \varepsilon \qquad \forall G_i^v \in D, \forall t \in C^s, \forall s \in V_{i,s} \backslash F_t \tag{18}$$

This inequality implies that if both p_s^i and m_s^i are equal one, ε has to be at least $d(s,t)$, otherwise it does not impose any constraint.

In summary the optimization model of the second stage is as follows:

Minimize ε

s. t. (2) – (18)

The solution of the first stage problem determines the maximum achievable profit and the corresponding acceptable VN requests. The solution of second model rearranges the mappings of the accepted requests in order to minimize the switch-controller delay while maintaining the profit.

Note that the solution of the second stage problem specifies the mapping of virtual nodes and links and also the placement of the virtual controllers. When the controller of G_i^v is mapped on substrate node t, $p_t^i = 1$, a hypervisor instance must also be installed on the server in node t to host the controller. Thus, in our approach, by solving the VNE problem that specifies the placement of the controllers, the location and the required number of hypervisor instances, the HPP problem, are also determined; in other words, the hypervisor placement problem is also implicitly solved.

Whereas decomposing the multi-objective problem into the sub-models decreases its complexity, unfortunately these formulations also cannot be used to solve the problem in a reasonable time and/or in practical instances due to the NP-Completeness of the problem [7]. This encourages us to develop a heuristic algorithm for embedding software-defined VNs. It should be noted that in the following sections, these optimization models are relaxed to obtain the performance bounds to evaluate the heuristic algorithms.

IV. The SVE Algorithm

In this section, we propose a heuristic algorithm named SVE to solve the multi-objective software-defined VNE problem. At the beginning, the features and underlying ideas of SVE are explained, and then, in the following subsections, the details are discussed.

The main features of SVE differentiating it from the existing methods are as follows:

- In addition to coordination between nodes and links mapping stages, for the first time, SVE coordinates the controller placement and nodes mapping stages too.
- SVE tries to maximize the profit of NaaS provider meanwhile it also guarantees that the worst switch-controller delay does not exceed *r*.
- In SVE, in addition to the CPU capacity, the number of entries in flow table is also considered as a node constraint.

The SEV algorithm is designed based on a few key ideas. First, traditional VNE algorithms, e.g., [14], aim to distribute VN requests in the substrate network in order to balance the load and avoid bottlenecks in the substrate network. However, in SVE, an opposite idea is used to reflect the architectural differences between SDN and traditional networks. SVE takes the importance of switch-controller delay into account by mapping virtual nodes around the controller.

Second, distributing virtual nodes in the substrate network leads to significant resource consumption by virtual links since they are mapped on long paths. To reduce the cost, SVE coordinates the node and link mapping stages by considering the bandwidth of virtual links in the node mapping stage.

Third, in the off-line version of the VNE problem, the information about all VN requests is available at the beginning. This information is used by SVE to increase the profit by maximizing the number of accepted requests while minimizing the cost of mapping. Both the acceptance probability and the mapping cost are proportional to the amount of substrate resources needed for mapping a request; and it is mainly influenced by virtual links because each virtual node is always mapped on a single substrate node but a virtual link can be mapped on a *path* in substrate network where its length determined by the embedding algorithm. To take this fact in consideration, SVE sorts VN requests according to the total number of virtual links in *descending* order. In this way, large requests that make more revenue are processed before other requests; therefore, their acceptance probabilities increase that enhances the revenue, and since substrate links have not been consumed by the other VNs, their large number of virtual links are mapped on short paths, that reduces the mapping cost.

Based on these ideas, the SVE algorithm, after sorting the requests, maps each VN in three stages. At first step, the location of network controller is specified wherein mapping the nodes around the controller is taken in consideration. In the second step, it maps the nodes while reflects the cost of link mapping. Finally, at the third step, the links are mapped. The details and design considerations of these steps are explained in the following subsections.

A. Controller Placement Stage

Controller placement has significant impact on both the NaaS provider's and customer's objectives. More precisely, resource availability around the controller's location in the substrate network determines both the acceptance probability and the switch-controller delay. If controller is mapped on a node where there is not enough resource in the neighbor, the VN request is likely rejected or the virtual nodes of the request have to be mapped away from the controller that increases the delay and also the mapping cost.

To locate a suitable place for controller and switches, in this paper, we use the *stress index* as the load measure. The node and link stress indexes are defined in [23]; and redefined for SDN paradigm in [14]. In this paper, we use the same definition of the node and link stress indexes.

For a substrate node $s \in V^s$, its stress is a weighted sum of the number of virtual nodes mapped on it ($n_N(s)$), and the number of virtual links traversing that node ($n_L(s)$). More formally, the stress of substrate node s, denoted by $S_N(s)$, is

$$S_N(s) = \gamma \cdot n_N(s) + \theta \cdot n_L(s), \tag{19}$$

where γ and θ are design parameters that determine the importance of each factor of the node stress.

The stress of the substrate link $(s,t) \in E^s$ is the sum of data and control traffic loads. Since we assumed that control plane traffic is negligible in comparison to the data plane, it is not taken into account. Let $T(s,t)$ denotes the virtual links mapped on substrate link (s,t); the stress of the link is

$$S_L(s,t) = \sum_{G_i^v \in D} \sum_{(u,v)\in E_i^v \cap T(s,t)} \frac{b_{(u,v)}^i}{b_{(s,t)}}. \tag{20}$$

To measure the availability of resources around a substrate node, we used the concept of neighborhood resource availability (NR) which is defined in [23] as follows:

$$NR(s) = \left(S_N^{max} - S_N(s)\right) \left(\sum_{(s,t)\epsilon L_s}\left(S_L^{max} - S_L(s,t)\right)\right), \tag{21}$$

where $S_N^{max} = \max_{s\in V^s} S_N(s)$ and $S_L^{max} = \max_{(s,t)\in E^s} S_L(s,t)$ are respectively the maximum node and link stress of the substrate network. L_s is the set of substrate links adjacent to s. A high $NR(s)$ value indicates that node s and its adjacent links are lightly loaded.

$NR(s)$ reflects the NaaS provider's objective in controller placement stage. However, to take the customer's objective into account, the delay between node s and other substrate nodes should also be considered. For this purpose, another metric named *k-hop delay*, denoted by $HD(s,k)$, is defined. Let $K(s,k)$ is the k-hop bounded neighbors of s, i.e., the set of substrate nodes that are connected to node $s \in V^s$ through at most k hops, and then we have

$$HD(s,k) = \frac{\sum_{t \in K(s,k)} d(s,t)}{|K(s,k)|}. \qquad (22)$$

The value of k depends on node s and VN request G_i^v. It should be large enough to be able to map all $v \in V_i^v$ on the neighbors. More formally,

$$k = \text{argmin}\{|V_i^v| \leq |K(s,k)|\}. \qquad (23)$$

$HD(s,k)$ is different from the estimation used in [14], where the average delay to *all* substrate nodes is obtained. However, the idea behind $HD(s,k)$ is that the substrate nodes which are unlikely be used for mapping $v \in G_i^v$ in the case of selecting s for hosting the controller of G_i^v, should not affect the placement of the controller.

For placing the controller, SVE calculates $NR(s)$ for all nodes $s \in C^s$; the node with the largest NR is selected to place the controller. If there are several nodes with the same value of NR, the node with the lowest $HD(s,k)$ is selected.

B. *Nodes Mapping Stage*

The main question of the node mapping stage is to find a substrate node to map a given virtual node. However, in SVE, at first, we try to *find the proper order of the virtual nodes for mapping*. To clarify the importance of this issue, consider mapping of two virtual nodes $u, v \in V_i^v$, where the required bandwidth between them, $b_{(u,v)}^i$, is very high. If after mapping u on $s \in V^s$, nodes other than v are mapped, likely the substrate nodes around s are used. Hence, there is not any room to map v near s; so, v must be mapped on a node away from s that increases the length of the path for mapping virtual link (u,v); and consequently increases the mapping cost.

We introduce the *correlation coefficient* metric to determine the appropriate order of virtual nodes mapping that minimizes the cost. For a given request G_i^v, the correlation coefficient for every *non-mapped* virtual node $v \in V_i^v$ with respect to the mapped nodes of the request is defined as

$$Cor(v) = \max_{u \in M_i^v} b_{(u,v)}^i \qquad \forall G^i \in D, \forall v \in V_i^v, \qquad (24)$$

where, $M_i^v \subseteq V_i^v$ is a set of the virtual nodes that mapped before v. This metric measures the correlation between a non-mapped node v and the mapped nodes M_i^v in terms of required bandwidth between them. $Cor(v) > Cor(v')$ indicates that required bandwidth between v and the mapped nodes is greater than the corresponding bandwidth for v'; so, as explained before, to minimize the cost of the virtual link mapping in the substrate network, node v should be mapped before v'. Based on this idea, in the node mapping stage, SVE *dynamically* ranks virtual nodes based on $Cor(v)$ in descending order[1].

This node ranking mechanism is different from the procedures proposed in [24-26] that determine a *static* order of virtual node mapping at the beginning. In SVE, the order of unmapped nodes depends on the mapped nodes. In this way, it considers the resources that will be used by virtual links and makes coordination between the node mapping and the link mapping stages. Moreover, since this metric reflects the required virtual links bandwidth, SVE maps bandwidth intensive links on shorter paths to reduce the cost.

Ranking substrate nodes for mapping a virtual node is the second issue in the node mapping stage. SVE ranks the substrate nodes according to the *weight* metric. The weight for a substrate node $s \in V_{i,s}$ with respect to virtual node $v \in V_i^v$ is defined as follows:

[1] For mapping the first node, where M_i^v is empty, the node with the largest degree is selected.

$$w(s,v) = \sum_{u \in M_i^v} b_{(u,v)}^i \cdot h(s, s_u), \tag{25}$$

where, and s_u is the substrate node that virtual node u is mapped on, and $h(s, s_u)$ is the number of hops between the substrate nodes s and s_u. By this definition, the weight of s for mapping v is the lower bound on the total bandwidth which will be used in the substrate network for mapping the virtual links (u, v) if v is mapped on s. By selecting a substrate node s with the minimum weight, SVE coordinates the node and link mapping stages even more to minimize the cost.

Similar to the ranking of the virtual nodes, substrate nodes' ranking should also consider the NaaS customer's objective, i.e., minimizing the switch-controller delay. For this purpose, we define the *Distance* of substrate node $s \in V^s$ with respect to virtual node $v \in V_i^v$ as follows:

$$Distance(s,v) = (1-\delta)\left(\frac{w(s,v)}{\max\limits_{t \in N_i^v} w(t,v)}\right) + \delta\left(\frac{d_s^c}{\max\limits_{t \in N_i^v} d_t^c}\right), \tag{26}$$

where, $N_i^v \subseteq V_{i,s}$ is the set of substrate nodes that are **not** used for mapping virtual nodes $v \in V_i^v$ up to now, d_s^c is the delay between s and the substrate node that the controller of G_i^v is mapped on it; and δ is a coefficient between 0 to 1. By changing the value of δ, the importance of each term is controlled. Small value of δ reduces the importance of the switch-controller delay while its large value reduces the importance of substrate resource consumption. To select the proper substrate node for mapping v, $Distance(s, v)$ is calculated for all substrate nodes $s \in N_i^v$ that have enough CPU and flow table resources, then the node with the smallest Distance value is selected to map this virtual node.

C. *Link Mapping Stage*

In this paper, each virtual link is mapped on a single path in the substrate network; therefore mapping all virtual links of a given VN is an instance of the *integer* multi-commodity flow problem, which is NP-hard [27]. To tackle its complexity, in SVE algorithm, the *K-Shortest Path* algorithm is used to map the virtual links. To map $(u, v) \in E_i^v$, the κ minimum-delay paths are found from s_u to s_v in the substrate network and (u, v) is mapped on the shortest feasible path, i.e., has sufficient residual bandwidth. The flowchart of the SVE algorithm is illustrated in Fig. 2.

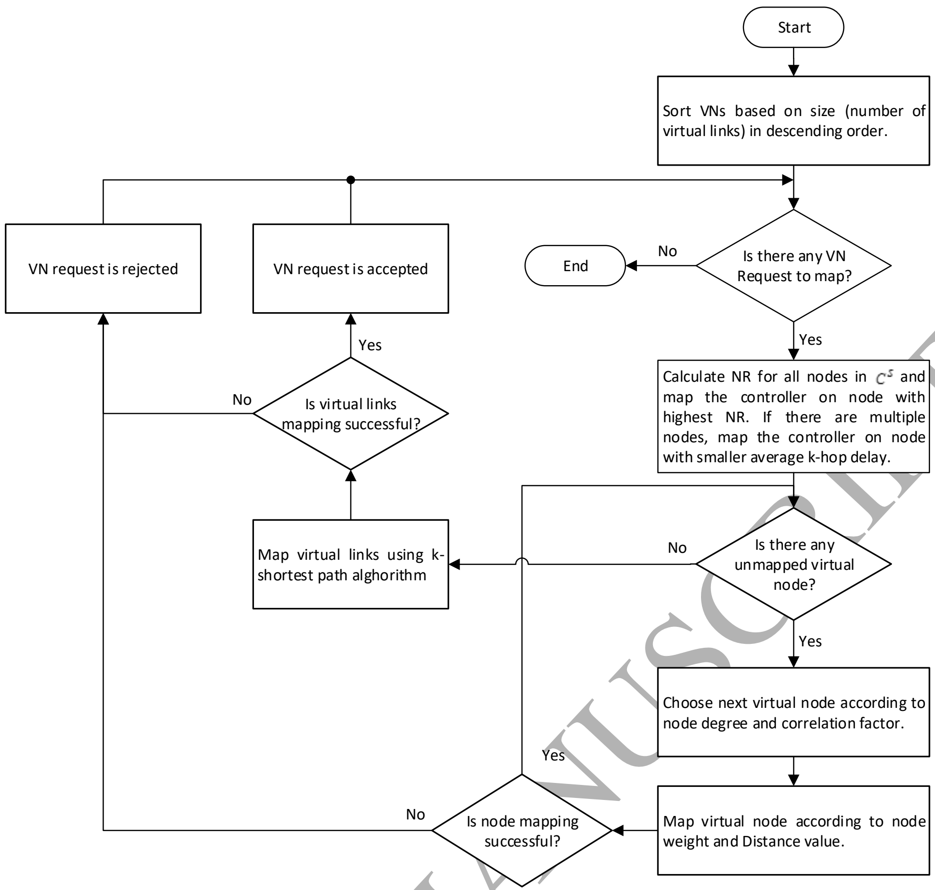

Start
Sort VNs based on size (number of virtual links) in descending order.
Is there any VN Request to map?
No
End
Yes
Calculate NR for all nodes in C^S and map the controller on node with highest NR. If there are multiple nodes, map the controller on node with smaller average k-hop delay.
Is there any unmapped virtual node?
No
Yes
Choose next virtual node according to node degree and correlation factor.
Map virtual node according to node weight and Distance value.
Is node mapping successful?
Yes
No
Map virtual links using k-shortest path alghorithm
Is virtual links mapping successful?
Yes
No
VN request is accepted
VN request is rejected

Fig. 2. The Overall flowchart of the SVE algorithm

V. Evaluation and Numerical Results

In this section, the performance of SVE is compared with the SBE and DME algorithms [14] and the bounds obtained from the optimization models. SBE aims to balance stress on substrate nodes and links while guaranteeing the worst switch-controller delay. On the other hand, DME tries to minimize the average switch-controller delays while limiting the stress.

A. Simulation Settings

For evaluations, the ALEVIN simulator[10] was used [15]. The simulations were performed in 10 substrate networks with different topologies and sizes, depicted in Table II. The Waxman Generator was used to create the virtual network requests, where the number of nodes is between 5 to 18, parameter α is randomly selected from interval [0.3, 0.7], and $\beta = 0.5$.

In the following figures, the horizontal axis is the number of VN requests, which is the measure of network load. In the simulations, for each number of requests, we created five different equal-size sets of VNs. The sets of VNs are mapped on 10 different substrate networks. Therefore, the results in the following figures are the average of 50 different embedding experiments, which is sufficient for about 94% confidence interval in the results.

[10] ALEVIN does not support SDN by default. It is extended for this purpose which is available at **ceit.aut.ac.ir/~bakhshis/papers/alevin-fork.zip**

In these simulations, we assumed that all substrate nodes are capable to host the controller, i.e., $C^s = V^s$; moreover, all substrate nodes can be used for mapping each request, i.e., $V_{i,s} = V^s \ \forall G_i^v$. The simulation settings are summarized in Table III.

Since the SBE and DME algorithms do not consider the node and link capacity constraints and consequently always accept VN requests, we made modifications to enforce the resource constraints in the algorithms.

Table II. Substrate Networks in Simulation

Substrate network	# of nodes	# of links
1	39	172
2	37	114
3	40	178
4	54	162
5	65	216
6	50	176
7	37	164
8	34	166
9	33	132
10	32	116

Table III. Simulation Settings

Parameter	Value
Virtual Network Generator	Waxman Generator
Waxman α parameter	Randomly in [0.3, 0.7]
Waxman β parameter	0.5
Number of nodes per VNR	[5, 18]
C^s	V^s
$V_{i,s}$	$V^s \ \forall G_i^v$
SVE δ parameter	0.25
r	50 ms
SVE γ parameter	1
SVE θ parameter	1
SVE κ parameter	50
α_1	100
α_2	100
β	1

In addition to the heuristic algorithms, the results of the optimization models are also presented as the benchmark to evaluate the efficiency of the algorithms. Since the problem is NP-Hard, the models cannot be solved even for small instances; so, in the following results, the upper-bounds for the revenue and lower-bounds for the switch-controller delay were obtained by relaxing the binary variables $x_{v,s}^i$ and $y_{(u,v),(s,t)}^i$.

B. *Numerical Results*

In this section, we use the acceptance rate, revenue, and revenue-to-cost ratio metrics to evaluate the algorithms; these are the commonly used evaluation criteria of VNE algorithms. Moreover, the average and maximum of the switch-controller delay, the particular metric of software-defined virtual networks, are also evaluated.

B.1. Acceptance, Revenue and Cost

Fig. 3 shows the acceptance rate of the SVE, SBE, and DME algorithms. The results show that SVE achieves better acceptance rate than the others since it considers the substrate resource consumption by means of the correlation coefficient and distance metrics. The efficiency of SVE increases as more load offered to the network, i.e., increasing the number of VN requests, because of the proper ordering of VN requests according to their size by this algorithm. The performance of SVE is comparable to the upper bound especially in the lightly loaded networks.

The revenue generated by each VNE algorithm is important for the NaaS provider. The comparison between the revenue of the algorithms and the mathematical upper bound, i.e., the value of the objective function of the LP relaxation of the first stage model, is depicted in Fig. 4. These results show that, not only SVE accepts more requests, which is depicted in Fig. 3, but also, it generates more revenue since it exploits the available information about the requests to order and map them accordingly. Moreover, it has a comparable performance with respect to the mathematical bound. The gap between SVE and the bound increases by the number of VN request which is in part due to looseness of the bound.

Revenue to cost ratio is another important metric from the NaaS provider's business point of view, since two algorithms can have the same revenue and acceptance rate but in different amount of substrate network resource consumption. This ratio shows how well the substrate network resources are used. The results are shown in Table IV. As it shown, the SVE algorithm achieved considerably higher revenue-to-cost ratio. The reason is the coordination between node and link mapping stages in this algorithm. It tries to map adjacent nodes near each other by using the correlation coefficient and distance metrics that decreases the cost of mapping.

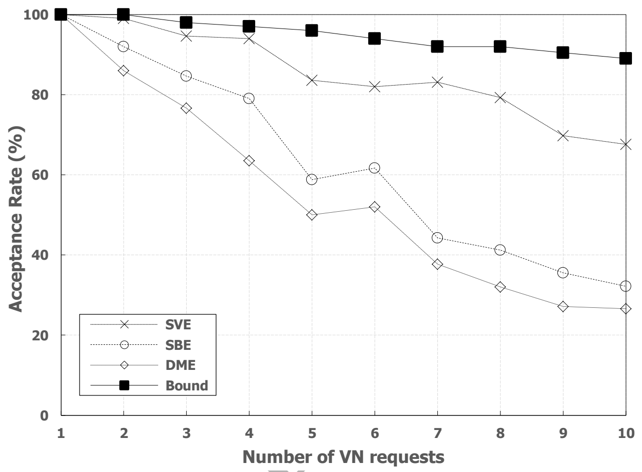


Fig. 3. Acceptance rate of the algorithms and its upper bound with respect to number of VN requests

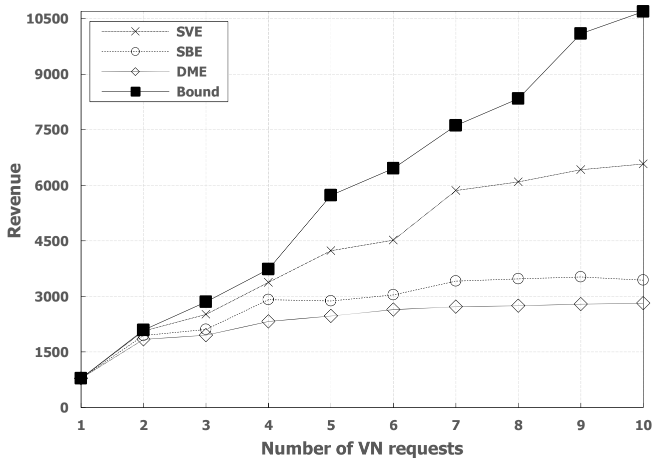


Fig. 4. Revenue of the algorithm and its upper bound with respect to number of VN requests

Table IV. Revenue-to-cost ratio of the algorithms and its upper bound with respect to number of VN requests

The number of requests	Approach			
	SVE	SBE	DME	Bound
2	0.609	0.378	0.486	0.812
4	0.634	0.388	0.485	0.804
6	0.608	0.384	0.489	0.781
8	0.600	0.375	0.460	0.752
10	0.580	0.367	0.455	0.752

B.2. Switch-controller delay

The most important performance metric for the NaaS customer, i.e., the switch-controller delay, is evaluated in this section. The average and maximum of the delay for the heuristic algorithms and the lower bounds are depicted in Fig. 5 and Fig. 6, respectively.

The average switch-controller delay in the networks mapped by SVE is less than the VNs which are mapped by SBE, and it is comparable to DME. Note that results obtained from DME are less than the results by SVE and even by the optimization models, it is not surprising. The algorithm only aims to minimize the delay and does not consider NaaS provider's profit therefore it accepts a fewer requests and minimize the delay in the accepted demands which can be less than the delay in the case of accepting more requests, which is obtained by SVE or the optimization model.

Therefore, it accepts a little number of requests (which is shown in Fig. 3) and tries to map the nodes around the controller to minimize the delay. On the other hand, the average delay for SBE is very high since it does not attempt to minimize it. The lower bound on the delay, i.e., the value of the objective function of the second stage model, is also depicted in the figure that shows the efficiency of SVE. As shown in Fig. 6, similar to the average switch-controller delay, the maximum of the delay for SVE is much less than SBE and more than DME. These results, in conjunction with the results in the previous section, confirm that SVE can efficiently satisfy both the provider and customer objectives.

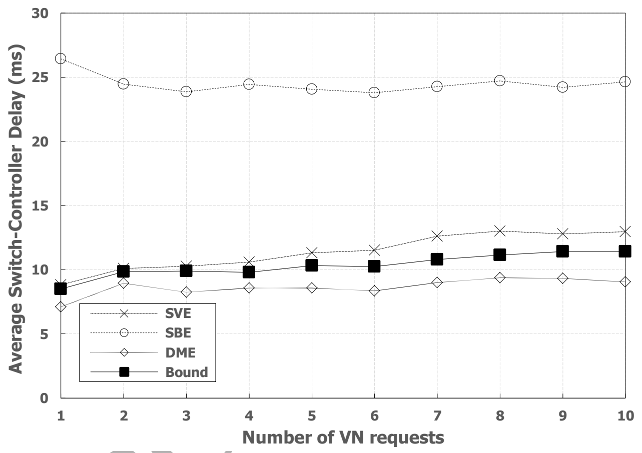


Fig. 5. The average switch-controller delay in mapped VN requests by the algorithms and optimization model

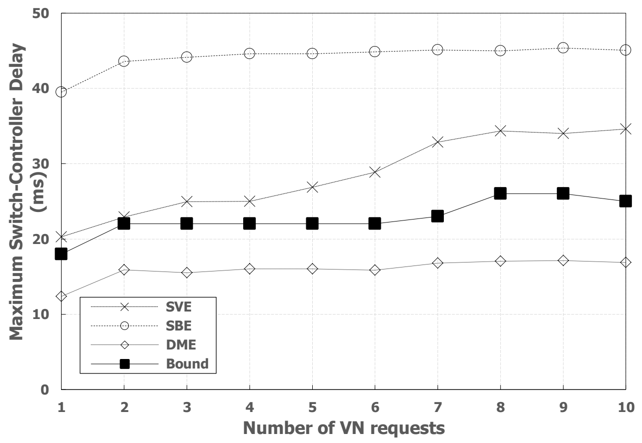


Fig. 6. The maximum switch-controller delay in mapped VN requests by the algorithms and optimization model

B.3. Impact of design parameters

SVE is a parametric algorithm, where the parameters k and δ influence on the performance of the algorithm. The parameter k determines the number of hops which are used to estimate the K-hop delay. In SVE, it is claimed that bounding the number of hops leads to a better estimation of delay, and consequently more suitable location of controller is found that decreases the delay between switches and controllers. This statement is satisfied in Fig. 7 where the average delays of two versions of SVE are depicted. In one version, depicted by dashed lines, the K-hop bounding is used where k is determined by (23), while in the second version, depicted by solid line, the K-hop bounding mechanism is removed from the algorithm. As indicated, using the K-hop delay leads to decrease in the switch-controller communication delay.

The second parameter is δ that balances between the profit and the delay. The effect of this parameter is shown in Fig. 8 that depicts the average switch-controller delay by SVE for different values of δ. By increasing the value of δ, SVE puts more attention to the delay and tries to map virtual nodes as close as possible to the controller. Increasing the value of δ causes that SVE does make effort to map virtual nodes alongside each other. Therefore, virtual links are mapped on longer paths that increase the cost and consequently decrease the revenue-to-cost ratio. This effect of δ on the ratio is shown in Table V.

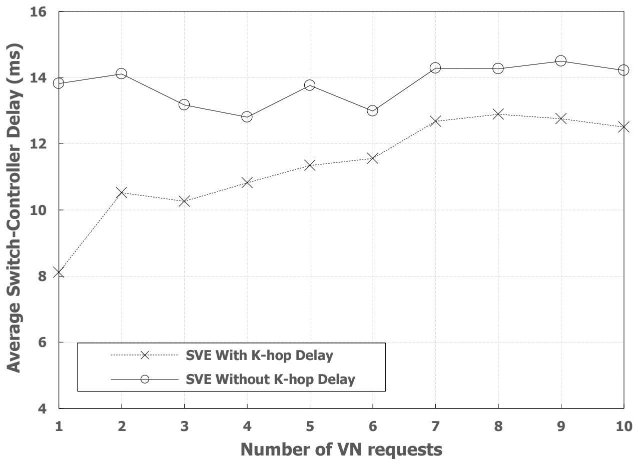


Fig. 7. The impact of “K-hop delay” in the SVE algorithm on the switch-controller delay in the mapped VN requests

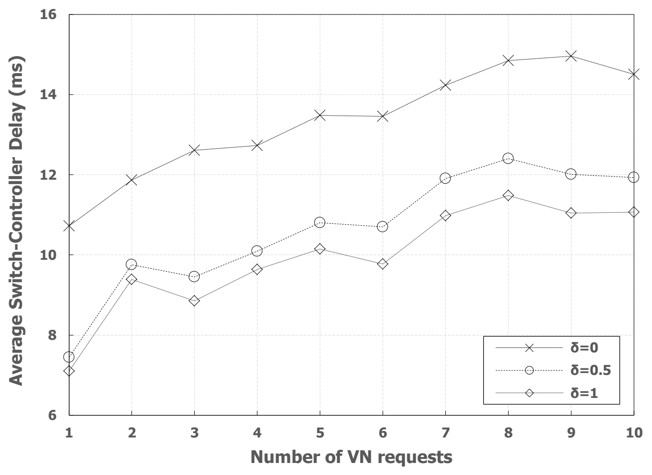


Fig. 8. The average switch-controller delay for different values of parameter δ in the SVE algorithm

Table V. The revenue to cost ratio of SVE for different values of δ

The number of requests	revenue-to-cost ratio		
	$\delta = 0$	$\delta = 0.5$	$\delta = 1$
2	0.631	0.591	0.518
4	0.632	0.614	0.551
6	0.614	0.599	0.550
8	0.612	0.582	0.512
10	0.599	0.563	0.513

VI. Conclusion and Future Work

In this paper, the VNE problem is formulated in SDN ecosystem wherein a set of software-defined VN requests are mapped on a SDN based substrate network in order to maximize the profit of the NaaS provider and minimize the delay between switches and controller in the mapped VNs. The problem is solved by the proposed algorithm which consists of three coordinated stages namely the controller placement, virtual nodes mapping, and virtual links mapping stages.

In the controller placement stage, SVE maps controller to the node with most resources in its neighbor according to the *NR* metric, and also considers the switch-controller delay by the *K-hop delay* metric. In the node mapping stage, it selects virtual nodes for mapping in descending order of the *correlation coefficient* metric that considers the amount of traffic volume between virtual nodes; and maps them on the substrate node with the minimum *weight* and *distance*. Finally, an instance of the distributed hypervisor is installed in the nodes where at least a controller mapped on.

In this paper, we considered several practical aspects of VNE in SDN; the following issues can be investigated in future work:

- Whereas we use the NR, correlation coefficient, and distance metrics to coordinate controller placement, virtual nodes mapping, and virtual links mapping; these stages are carried out in three separate phases. For the next step, to achieve higher efficiency, these problems can be tackled as a joint problem.
- In this paper, the embedding problem was considered in off-line mode, wherein the information of all requests is available at the beginning; at the next step, the problem can be studied in on-line mode where whole VN requests are not known in advance and arrive to the network one-by-one.

References

[1] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE,* vol. 103, pp. 14-76, 2015.

[2] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks,* vol. 54, pp. 862-876, 2010.

[3] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong*, et al.*, "Forwarding and control element separation (ForCES) protocol specification," 2070-1721, 2010.

[4] T. Lakshman ,T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo, "The softrouter architecture," in *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking*, 2004.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford*, et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review,* vol. 38, pp. 69-74, 2008.

[6] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 7-12.

[7] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electronic Notes in Discrete Mathematics,* pp. 213-220, 2016.

[8] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown*, et al.*, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep,* pp. 1-13, 2009.

[9] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Communications Surveys & Tutorials,* vol. 18, pp. 655-685, 2016.

[10] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials,* vol. 15, pp. 1888-1906, 2013.

[11] R. Trivisonno, I. Vaishnavi, R. Guerzoni, Z. Despotovic, A. Hecker, S. Beker*, et al.*, "Virtual Links Mapping in Future SDN-Enabled Networks," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, 2013, pp. 1-5.

[12] A. R. Roy, M. F. Bari, M. F. Zhani, R. Ahmed, and R. Boutaba, "Design and management of dot: A distributed openflow testbed," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1-9.

[13] R. Guerzoni, R. Trivisonno, I. Vaishnavi ,Z. Despotovic, A. Hecker, S. Beker*, et al.*, "A novel approach to virtual networks embedding for SDN management and orchestration," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, 2014, pp. 1-7.

[14] M. Demirci and M. Ammar, "Design and analysis of techniques for mapping virtual networks to software-defined network substrates," *Computer Communications,* vol. 45, pp. 1-10, 2014.

[15] A. Fischer, J. F. Botero Vega, M. Duelli, D. Schlosser, X. Hesselbach Serra, and H. De Meer, "ALEVIN-a framework to develop, compare, and analyze virtual network embedding algorithms," in *Open-Access-Journal Electronic Communications of the EASST*, 2011, pp. 1-12.

[16] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review,* vol. 38, pp. 17-29, 2008.

[17] Y. Zhou, Y. Li, D. Jin, L. Su, and L. Zeng, "A virtual network embedding scheme with two-stage node mapping based on physical resource migration," in *Communication Systems (ICCS), 2010 IEEE International Conference on*, 2010, pp. 761-766.

[18] R. Riggio, F. De Pellegrini, E. Salvadori, M. Gerola, and R. D. Corin, "Progressive virtual topology embedding in OpenFlow networks," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, 2013, pp. 1122-1128.

[19] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking (TON),* vol. 20, pp. 206-219, 2012.

[20] A. Blenk, A. Basta, J. Zerwas, and W. Kellerer, "Pairing SDN with network virtualization: The network hypervisor placement problem," in *Network Function Virtualization and Software Defined Network (NFV-SDN) ,IEEE Conference on*, 2015, pp. 198-204.

[21] A. Blenk, A. Basta, J. Zerwas, M. Reisslein, and W. Kellerer, "Control plane latency with SDN network hypervisors: The cost of virtualization," *IEEE Transactions on Network and Service Management,* vol. 13 ,pp. 366-380, 2016.

[22] N. M. K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Communications magazine,* vol. 47, 2009.

[23] Y. Zhu and M. H. Ammar, "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," in *INFOCOM*, 2006.

[24] J. Ding, T. Huang, J. Liu, and Y.-j. Liu, "Virtual network embedding based on real-time topological attributes," *Frontiers of Information Technology & Electronic Engineering,* vol. 16, pp. 109-118, 201.5

[25] M. Feng, J. Liao, J. Wang, S. Qing, and Q. Qi, "Topology-aware virtual network embedding based on multiple characteristics," in *Communications (ICC), 2014 IEEE International Conference on*, 2014, pp. 2956-2962.

[26] Z. Wang, Y. Han, T. Lin, H. Tang ,and S. Ci, "Virtual network embedding by exploiting topological information," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, 2012, pp. 2603-2608.

[27] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *Foundations of Computer Science, 1975., 16th Annual Symposium on*, 1975, pp. 184-193.