Network Load Minimization-Based Virtual Network Embedding Algorithm for Software-Defined Networking

Desheng Xie(✉), Rong Chai, Mengqi Mao, Qianbin Chen, and Chun Jin

Key Lab of Mobile Communication Technology,
Chongqing University of Posts and Telecommunications, Chongqing 400065, China
{1050386890,1747495352}@qq.com, {chairong,chenqb,jinchun}@cqupt.edu.cn

Abstract. In a network virtualization-enabled software-defined networking (SDN), the problem of virtual network embedding (VNE) is a major concern. Although a number of VNE algorithms have been proposed, they fail to consider the efficient utilization of substrate resources or the network load extensively, thus resulting in less efficient utilization of substrate resources or higher blocking ratio of the virtual networks. In this paper, we study the problem of mapping a number of virtual networks in SDN and formulate the VNE problem as a network load minimization problem. Since the formulated optimization problem is NP-hard and it cannot be solved conveniently, we propose a two-stage VNE algorithm consisting of node mapping stage and link mapping stage. Numerical results demonstrate that the effectiveness of our proposed algorithm.

Keywords: Software-defined networking · Network virtualization · Virtual network embedding · Network load

1 Introduction

The emerging demands for mobile Internet applications and multimedia contents bring challenges and difficulties to traditional Internet architecture and technologies. To tackle this problem, software-defined networking (SDN) is proposed as a new networking paradigm which decouples network control from data forwarding functions to enable the flexible network service offering [1]. As another key enabler of the future Internet, network virtualization (NV) is capable of shielding the complexity of the underlying network deployment, and facilitating the flexible management of resources by creating and operating multiple logically-isolated virtual networks on top of substrate network [2]. To achieve NV, the problem of virtual network embedding (VNE) should be considered, which is mapping virtual networks onto a shared substrate network [3].

X. Liu et al. (Eds.): ChinaCom 2018, LNICST 262, pp. 259–270, 2019.
https://doi.org/10.1007/978-3-030-06161-6_26

The problem of VNE has been studied in some recent works. References [4–6] considering the energy consumption issue in designing the VNE algorithms. The authors in [4] proposed an energy-cost model for conducting virtual network embedding in SDN and formulated the energy-aware virtual network embedding problem as an integer linear programming. To stress the problem of non-optimality in energy consumption and the utilization of the substrate network, the authors in [5] proposed an integer linear programming based green mapping algorithm which facilitated the migration of virtual routers and links to achieve the minimum energy consumption. In [6], the authors formulated an energy efficient virtual node embedding model and proposed a minimal element method-based heuristic algorithm to solve the formulated optimization problem and obtain the near-optimal virtual network embedding strategy.

To solve the location-constrained virtual network embedding (LC-VNE) problem efficiently, the authors in [7] reduced LC-VNE to the minimum-cost maximum clique problem and proposed a greedy LC-VNE algorithm and load-balance enhanced LC-VNE algorithm to achieve the integrated node and link mapping strategy with significantly reduced time complexity. To achieve the tradeoff between the profits of physical infrastructure providers and the waiting time for virtual network requests, the authors in [8] proposed an auction-based joint node and link embedding algorithm by using column generation approach.

To support multicast services over the virtual networks, reliability should be a major concern. The authors in [9] formulated a mixed integer linear programming model to determine the upper bound on the reliability of the network with the max-min fairness and proposed a reliability-aware genetic algorithm to achieve reliable multicast VN mapping under low computational complexity. Leveraging the consensus-based resource allocation schemes, the authors in [10] proposed a general distributed auction mechanism for the virtual network embedding problem and demonstrated that the obtained solutions guarantee a worst-case efficiency relative to the optimal virtual network embedding. To solve the cost-effective VNE problem in SDN, the authors in [11] formulated an integer linear programming model and designed a time-efficient heuristic algorithm to achieve the minimum resource consumption.

Though the aforementioned research works consider various VNE algorithms and aim to design the optimal embedding strategies which minimizes the energy consumption or enhances the embedding reliability, they fail to consider the network load or the utilization of the substrate resource extensively, which may result in less efficient utilization of the substrate resources or the high blocking rate of the virtual networks.

In this paper, we study the problem of mapping a number of virtual networks in SDN. Stressing the importance of efficient utilization of substrate resources, we define the network load of the substrate network, and formulate the VNE problem as a network load minimization problem. Since the formulated optimization problem is NP-hard and it cannot be solved conveniently, we propose a two-stage VNE algorithm consisting of node mapping stage and link mapping stage. During the node mapping stage, we examine and rank the performance

metrics of substrate nodes and virtual nodes by using recursive method, and then perform greedy strategy-based node mapping. And finally perform link mapping procedure by applying the K-shortest-path algorithm.

2 Network Model

2.1 Substrate Network

In this work, we model the substrate network as a weighted undirected graph which is denoted by $G^{\mathrm{s}} = (N^{\mathrm{s}}, E^{\mathrm{s}}, A_N^{\mathrm{s}}, A_E^{\mathrm{s}})$, where $N^{\mathrm{s}} = \{\mathrm{n}_i^{\mathrm{s}}, 1 \leq i \leq M\}$ denotes the set of substrate nodes, $\mathrm{n}_i^{\mathrm{s}}$ denotes the ith substrate node, M is the total number of the substrate nodes, $E^{\mathrm{s}} = \left\{\mathrm{e}_{i,j}^{\mathrm{s}}, 1 \leq i, j \leq M, i \neq j\right\}$ denotes the set of substrate links, and $\mathrm{e}_{i,j}^{\mathrm{s}}$ denotes the substrate link.

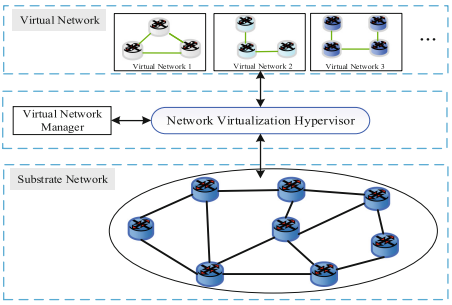

Fig. 1. System model of SDN

$A_N^{\mathrm{s}} = \{\{C(\mathrm{n}_i^{\mathrm{s}}), S(\mathrm{n}_i^{\mathrm{s}}), T(\mathrm{n}_i^{\mathrm{s}})\}, \mathrm{n}_i^{\mathrm{s}} \in N^{\mathrm{s}}\}$ denotes the attribute set of the substrate nodes, where $C(\mathrm{n}_i^{\mathrm{s}})$, $S(\mathrm{n}_i^{\mathrm{s}})$ and $T(\mathrm{n}_i^{\mathrm{s}})$ denote the CPU capacity, the storage capacity and the ternary content addressable memory (TCAM) capacity of $\mathrm{n}_i^{\mathrm{s}}$, respectively. $A_E^{\mathrm{s}} = \{\{B(\mathrm{e}_{i,j}^{\mathrm{s}}), D(\mathrm{e}_{i,j}^{\mathrm{s}})\}, \mathrm{e}_{i,j}^{\mathrm{s}} \in E^{\mathrm{s}}\}$ denotes the attribute set of the substrate links, where $B(\mathrm{e}_{i,j}^{\mathrm{s}})$ and $D(\mathrm{e}_{i,j}^{\mathrm{s}})$ denote the bandwidth capacity and the propagation delay of $\mathrm{e}_{i,j}^{\mathrm{s}}$, respectively.

2.2 Virtual Network

Denoting the number of virtual networks as K_0, we also characterize each virtual network as a weighted undirected graph. For the kth virtual network, we define the weighted undirected graph $G_k^{\mathrm{v}} = (N_k^{\mathrm{v}}, E_k^{\mathrm{v}}, A_{k,N}^{\mathrm{v}}, A_{k,E}^{\mathrm{v}})$, where $N_k^{\mathrm{v}} = \{\mathrm{n}_{k,u}^{\mathrm{v}}, 1 \leq u \leq M_k\}$ and $E_k^{\mathrm{v}} = \{\mathrm{e}_{k,u,r}^{\mathrm{v}}, 1 \leq u, r \leq M_k, u \neq r\}$ denote the set of virtual nodes and virtual links in the kth virtual network, respectively, $\mathrm{n}_{k,u}^{\mathrm{v}}$ denotes the uth virtual node of the kth virtual network, M_k denotes the total number of the virtual nodes of the kth virtual network, $\mathrm{e}_{k,u,r}^{\mathrm{v}}$ denotes the virtual link.

$A^{\mathrm{v}}_{k,N} = \{\{C(\mathrm{n}^{\mathrm{v}}_{k,u}), S(\mathrm{n}^{\mathrm{v}}_{k,u}), T(\mathrm{n}^{\mathrm{v}}_{k,u})\}, \mathrm{n}^{\mathrm{v}}_{k,u} \in N^{\mathrm{v}}_k\}$ denotes the attribute set of the virtual nodes of the kth virtual network, $C(\mathrm{n}^{\mathrm{v}}_{k,u})$, $S(\mathrm{n}^{\mathrm{v}}_{k,u})$ and $T(\mathrm{n}^{\mathrm{v}}_{k,u})$ denote the required CPU capacity, storage capacity and TCAM capacity of $\mathrm{n}^{\mathrm{v}}_{k,u}$, $A^{\mathrm{v}}_{k,E} = \{\{B(\mathrm{e}^{\mathrm{v}}_{k,u,r}), D(\mathrm{e}^{\mathrm{v}}_{k,u,r})\}, \mathrm{e}^{\mathrm{v}}_{k,u,r} \in E^{\mathrm{v}}_k\}$ denotes the attribute set of the virtual links of the kth virtual network, $B(\mathrm{e}^{\mathrm{v}}_{k,u,r})$ and $D(\mathrm{e}^{\mathrm{v}}_{k,u,r})$ denote the required bandwidth capacity and the tolerable propagation delay of $\mathrm{e}^{\mathrm{v}}_{k,u,r}$, respectively.

In this paper, we introduce a functional model called network virtualization hypervisor (NVH) to conduct virtual network embedding. We assume that the NVH is capable of collecting network state information. Upon receiving the virtual networks from the tenants, the NVH conducts the proposed virtual network embedding, and sends the strategy to the switches in the network. Figure 1 shows the system model considered in this paper.

3 Network Load Optimization Problem Formulation

In this section, we formulate the virtual network embedding problem of the SDN as a network load minimization problem. The detail problem formulation is described in this section.

3.1 Network Load Formulation

Considering the load of the substrate network for embedding all the virtual networks, we formulate the network load Ψ as

$$\Psi = \sum_{k=1}^{K_0} \Psi_k \tag{1}$$

where Ψ_k denotes the load of the substrate network caused by mapping the kth virtual network, and can be expressed as

$$\Psi_k = \Psi^{\mathrm{N}}_k + \Psi^{\mathrm{E}}_k \tag{2}$$

where Ψ^{N}_k and Ψ^{E}_k denote the load of the substrate nodes and substrate links, respectively, caused by mapping the kth virtual network.

Ψ^{N}_k in (2) can be expressed as

$$\Psi^{\mathrm{N}}_k = \sum_{\mathrm{n}^{\mathrm{v}}_{k,u} \in N^{\mathrm{v}}_k} \sum_{\mathrm{n}^{\mathrm{s}}_i \in N^{\mathrm{s}}} \alpha_{k,u,i} \frac{C(\mathrm{n}^{\mathrm{v}}_{k,u}) + S(\mathrm{n}^{\mathrm{v}}_{k,u}) + T(\mathrm{n}^{\mathrm{v}}_{k,u})}{C_k(\mathrm{n}^{\mathrm{s}}_i) + S_k(\mathrm{n}^{\mathrm{s}}_i) + T_k(\mathrm{n}^{\mathrm{s}}_i)} \tag{3}$$

where $\alpha_{k,u,i}$ denotes the binary node embedding variable that equals to 1 when $\mathrm{n}^{\mathrm{v}}_{k,u}$ is mapped to $\mathrm{n}^{\mathrm{s}}_i$, and 0 otherwise, $C_k(\mathrm{n}^{\mathrm{s}}_i)$, $S_k(\mathrm{n}^{\mathrm{s}}_i)$ and $T_k(\mathrm{n}^{\mathrm{s}}_i)$ denote respectively the residual CPU capacity, storage capacity and TCAM capacity of $\mathrm{n}^{\mathrm{s}}_i$ after conducting the virtual node mapping for the first $k-1$ virtual networks. In this paper, we assume that the K_0 virtual networks will be mapped successively from the first one to the K_0th one, hence, we obtain

$$C_k(\mathrm{n}^{\mathrm{s}}_i) = C(\mathrm{n}^{\mathrm{s}}_i) - \sum_{l=1}^{k-1} \sum_{\mathrm{n}^{\mathrm{v}}_{l,u} \in N^{\mathrm{v}}_l} \alpha_{l,u,i} C(\mathrm{n}^{\mathrm{v}}_{l,u}), \tag{4}$$

$$S_k(\mathrm{n}_i^{\mathrm{s}}) = S(\mathrm{n}_i^{\mathrm{s}}) - \sum_{l=1}^{k-1} \sum_{\mathrm{n}_{l,u}^{\mathrm{v}} \in N_l^{\mathrm{v}}} \alpha_{l,u,i} S(\mathrm{n}_{l,u}^{\mathrm{v}}). \tag{5}$$

$$T_k(\mathrm{n}_i^{\mathrm{s}}) = T(\mathrm{n}_i^{\mathrm{s}}) - \sum_{l=1}^{k-1} \sum_{\mathrm{n}_{l,u}^{\mathrm{v}} \in N_l^{\mathrm{v}}} \alpha_{l,u,i} T(\mathrm{n}_{l,u}^{\mathrm{v}}) - \sum_{l=1}^{k-1} \sum_{\mathrm{e}_{l,u,r}^{\mathrm{v}} \in E_l^{\mathrm{v}}} \beta_{l,u,r,i} T(\mathrm{e}_{l,u,r}^{\mathrm{v}}) \tag{6}$$

where $\beta_{l,u,r,i}$ denotes the binary embedding variable of intermediate node that equals to 1 when $\mathrm{n}_i^{\mathrm{s}}$ is an intermediate node of the substrate path that $\mathrm{e}_{l,u,r}^{\mathrm{v}}$ is embedded onto, and 0 otherwise.

Similarly, Ψ_k^{E} in (2) can be expressed as

$$\Psi_k^{\mathrm{E}} = \sum_{\mathrm{e}_{k,u,r}^{\mathrm{v}} \in E_k^{\mathrm{v}}} \sum_{\mathrm{e}_{i,j}^{\mathrm{s}} \in E^{\mathrm{s}}} \gamma_{k,u,r,i,j} \frac{B(\mathrm{e}_{k,u,r}^{\mathrm{v}})}{B_k(\mathrm{e}_{i,j}^{\mathrm{s}})} \tag{7}$$

where $\gamma_{k,u,r,i,j}$ denotes the binary link embedding variable that equals to 1 when $\mathrm{e}_{k,u,r}^{\mathrm{v}}$ is mapped to $\mathrm{e}_{i,j}^{\mathrm{s}}$, and 0 otherwise, $B_k(\mathrm{e}_{i,j}^{\mathrm{s}})$ represents the residual bandwidth capacity of $\mathrm{e}_{i,j}^{\mathrm{s}}$ after conducting the virtual link mapping for the first $k-1$ virtual networks, which can be expressed as

$$B_k(\mathrm{e}_{i,j}^{\mathrm{s}}) = B(\mathrm{e}_{i,j}^{\mathrm{s}}) - \sum_{l=1}^{k-1} \sum_{\mathrm{e}_{l,u,r}^{\mathrm{v}} \in E_l^{\mathrm{v}}} \gamma_{l,u,r,i,j} B(\mathrm{e}_{l,u,r}^{\mathrm{v}}). \tag{8}$$

3.2 Optimization Constraints

The optimization problem of virtual network embedding should subject to a number of constraints, as discussed in this subsection in detail.

Node Mapping Constraints In this paper, we assume that one virtual node of a particular virtual network can only be mapped onto one substrate node, i.e.,

$$\mathrm{C1}: \sum_{\mathrm{n}_i^{\mathrm{s}} \in N^{\mathrm{s}}} \alpha_{k,u,i} = 1. \tag{9}$$

Similarly, we assume that a substrate node can only map to at most one virtual node from a particular virtual network, hence, we obtain

$$\mathrm{C2}: \sum_{\mathrm{n}_{k,u}^{\mathrm{v}} \in N_k^{\mathrm{v}}} \alpha_{k,u,i} \leq 1. \tag{10}$$

Resource Constraints To map virtual nodes onto substrate nodes, the resource constraints on the substrate nodes, including the total CPU usage, the storage usage and the TCAM usage, should be satisfied, i.e.,

$$\mathrm{C3}: \sum_{k=1}^{K_0} \sum_{\mathrm{n}_{k,u}^{\mathrm{v}} \in N_k^{\mathrm{v}}} \alpha_{k,u,i} C(\mathrm{n}_{k,u}^{\mathrm{v}}) \leq \mathrm{C}(\mathrm{n}_i^{\mathrm{s}}), \tag{11}$$

$$\mathrm{C4}: \sum_{k=1}^{K_0} \sum_{\mathrm{n}_{k,u}^{\mathrm{v}} \in N_k^{\mathrm{v}}} \alpha_{k,u,i} S(\mathrm{n}_{k,u}^{\mathrm{v}}) \leq \mathrm{S}(\mathrm{n}_i^{\mathrm{s}}), \tag{12}$$

$$\mathrm{C5}: \sum_{k=1}^{K_0} \left(\sum_{\mathrm{n}_{k,u}^{\mathrm{v}} \in N_k^{\mathrm{v}}} \alpha_{k,u,i} T(\mathrm{n}_{k,u}^{\mathrm{v}}) + \sum_{\mathrm{e}_{k,u,r}^{\mathrm{v}} \in E_k^{\mathrm{v}}} \beta_{k,u,r,i} T(\mathrm{e}_{k,u,r}^{\mathrm{v}}) \right) \leq \mathrm{T}(\mathrm{n}_i^{\mathrm{s}}). \tag{13}$$

And the bandwidth constraint on the substrate links should also be satisfied, i.e.,

$$\mathrm{C6}: \sum_{k=1}^{K_0} \sum_{\mathrm{e}_{k,u,r}^{\mathrm{v}} \in E_k^{\mathrm{v}}} \gamma_{k,u,r,i,j} B(\mathrm{e}_{k,u,r}^{\mathrm{v}}) \leq \mathrm{B}(\mathrm{e}_{i,j}^{\mathrm{s}}). \tag{14}$$

Flow Conservation Constraint While mapping a virtual link to a substrate link, the flow conservation constraints should be met, which can be expressed as

$$\mathrm{C7}: \sum_{\mathrm{n}_j^{\mathrm{s}} \in N(\mathrm{n}_i^{\mathrm{s}})} \gamma_{k,u,r,i,j} - \sum_{\mathrm{n}_j^{\mathrm{s}} \in N(\mathrm{n}_i^{\mathrm{s}})} \gamma_{k,u,r,j,i} = \alpha_{k,u,i} - \alpha_{k,r,i}. \tag{15}$$

Intermediate Node Constraint To ensure that the intermediate nodes of the substrate path that carries $\mathrm{e}_{k,u,r}^{\mathrm{v}}$ are identified correctly, we obtain

$$\mathrm{C8}: \sum_{\mathrm{n}_j^{\mathrm{s}} \in N(\mathrm{n}_i^{\mathrm{s}})} \gamma_{k,u,r,i,j} - \beta_{k,u,r,i} = \alpha_{k,u,i}. \tag{16}$$

Tolerable Propagation Delay Constraint The total propagation delay of the substrate path that carries $\mathrm{e}_{k,u,r}^{\mathrm{v}}$ should not exceed the tolerable propagation delay of $\mathrm{e}_{k,u,r}^{\mathrm{v}}$, i.e.,

$$\mathrm{C9}: \sum_{\mathrm{e}_{i,j}^{\mathrm{s}} \in E^{\mathrm{s}}} \gamma_{k,u,r,i,j} \mathrm{D}(e_{i,j}^{\mathrm{s}}) \leq D(\mathrm{e}_{k,u,r}^{\mathrm{v}}). \tag{17}$$

3.3 Optimization Problem Modeling

Aiming to minimize the total network load subject to the constraints, we formulate the optimization problem as

$$\begin{aligned} \min_{\alpha_{k,u,i}, \beta_{k,u,r,i}, \gamma_{k,u,r,i,j}} \quad & \Psi \\ \text{s.t.} \quad & \mathrm{C1} - \mathrm{C9}. \end{aligned} \tag{18}$$

4 Solution to the Optimization Problem

The optimization problem formulated in (18) is NP-hard, the optimal solution of which is very difficult to obtain in general. In this paper, to tackle the resource sharing problem among the virtual networks, we propose a sequential VNE method by ordering the virtual networks according to their resource requirement. Then for individual virtual network, we propose a two-stage VNE algorithm consisting of node mapping stage and link mapping stage.

4.1 Ordering the Virtual Networks

In this subsection, we examine the resource requirement of the virtual networks by summing up the required resources of all the virtual nodes and links contained in the virtual network. For the kth virtual network, the amount of the required resources can be calculated as

$$R(G_k^{\mathrm{v}}) = \sum_{\mathrm{n}_{k,u}^{\mathrm{v}} \in N_k^{\mathrm{v}}} (C(\mathrm{n}_{k,u}^{\mathrm{v}}) + S(\mathrm{n}_{k,u}^{\mathrm{v}}) + T(\mathrm{n}_{k,u}^{\mathrm{v}})) + \sum_{\mathrm{e}_{k,u,r}^{\mathrm{v}} \in E_k^{\mathrm{v}}} B(\mathrm{e}_{k,u,r}^{\mathrm{v}}). \quad (19)$$

Based on $R(G_k^{\mathrm{v}})$, $1 \le k \le K_0$, we can then rank the virtual networks according to the amount of the required resources. Aim to utilize the substrate resources in an efficient manner, we first conduct the VNE for the virtual network with the largest resource requirement, and then conduct the VNE for the virtual network with the second largest resource requirement. This process repeats until all the virtual networks have been embedded or there is no enough substrate resources being available.

4.2 Node Mapping Stage

For certain virtual network, we first conduct node mapping. In this paper, we design a recursive method-based performance metrics evaluation algorithm for both the substrate nodes and virtual nodes, then rank the substrate nodes and virtual nodes according to the obtained performance metrics and conduct node mapping according to the rank list based on the greedy strategy.

Recursive Method-Based Performance Metrics Evaluation Algorithm To examine the performance metric of one substrate node, we consider the amount of the resources that the node can offer and define the performance metric of $\mathrm{n}_i^{\mathrm{s}}$, denoted by $V(\mathrm{n}_i^{\mathrm{s}})$ as

$$V(\mathrm{n}_i^{\mathrm{s}}) = (1-\delta)\bar{R}(\mathrm{n}_i^{\mathrm{s}}) + \delta \sum_{\mathrm{n}_j^{\mathrm{s}} \in N(\mathrm{n}_i^{\mathrm{s}})} \frac{\bar{R}(\mathrm{n}_j^{\mathrm{s}})}{\sum_{\mathrm{n}_h^{\mathrm{s}} \in N(\mathrm{n}_i^{\mathrm{s}})} \bar{R}(\mathrm{n}_h^{\mathrm{s}})} V(\mathrm{n}_j^{\mathrm{s}}) \quad (20)$$

where $\delta \in [0, 1)$ is the relative weight of the neighbor performance metric, $N(\mathrm{n}_i^{\mathrm{s}})$ denotes the set of neighbors of $\mathrm{n}_i^{\mathrm{s}}$, and $\bar{R}(\mathrm{n}_i^{\mathrm{s}})$ is the amount of the normalized resources on $\mathrm{n}_i^{\mathrm{s}}$, which can be defined as

$$\bar{R}(\mathrm{n}_i^{\mathrm{s}}) = \frac{R(\mathrm{n}_i^{\mathrm{s}})}{\sum\limits_{\mathrm{n}_j^{\mathrm{s}} \in N^{\mathrm{s}}} R(\mathrm{n}_j^{\mathrm{s}})} \tag{21}$$

where $R(\mathrm{n}_i^{\mathrm{s}})$ denotes the amount of resources on $\mathrm{n}_i^{\mathrm{s}}$. To evaluate the amount of resources on $\mathrm{n}_i^{\mathrm{s}}$, we jointly consider node degree, the characteristics of link resources and node resources, and formulate $R(\mathrm{n}_i^{\mathrm{s}})$ as

$$R(n_i^{\mathrm{s}}) = d(\mathrm{n}_i^{\mathrm{s}}) l(\mathrm{n}_i^{\mathrm{s}}) r(\mathrm{n}_i^{\mathrm{s}}) \tag{22}$$

where $d(\mathrm{n}_i^{\mathrm{s}})$ denotes the degree of $\mathrm{n}_i^{\mathrm{s}}$, which can be expressed as

$$d(\mathrm{n}_i^{\mathrm{s}}) = |N(\mathrm{n}_i^{\mathrm{s}})| \tag{23}$$

where $|x|$ denotes the number of elements in set x.

$l(\mathrm{n}_i^{\mathrm{s}})$ in (22) denotes the characteristics of the link resource of $\mathrm{n}_i^{\mathrm{s}}$, which is defined as

$$l(\mathrm{n}_i^{\mathrm{s}}) = \sum_{\mathrm{n}_j^{\mathrm{s}} \in N(\mathrm{n}_i^{\mathrm{s}})} B(\mathrm{e}_{i,j}^{\mathrm{s}}) \left(\frac{D^{\max} - D(\mathrm{e}_{i,j}^{\mathrm{s}})}{D^{\max} - D^{\min}}\right) \tag{24}$$

where $D^{\max}$ and $D^{\min}$ denote respectively the maximum and the minimum propagation delay of the links in the substrate network, which can be expressed as

$$D^{\max} = \max\left\{D(\mathrm{e}_{i,j}^{\mathrm{s}}), 1 \le i, j \le M, i \ne j\right\}, \tag{25}$$

$$D^{\min} = \min\left\{D(\mathrm{e}_{i,j}^{\mathrm{s}}), 1 \le i, j \le M, i \ne j\right\}. \tag{26}$$

$r(\mathrm{n}_i^{\mathrm{s}})$ in (22) denotes the characteristics of the node resource of $\mathrm{n}_i^{\mathrm{s}}$, which can be calculated as

$$r(\mathrm{n}_i^{\mathrm{s}}) = C(\mathrm{n}_i^{\mathrm{s}}) + S(\mathrm{n}_i^{\mathrm{s}}) + T(\mathrm{n}_i^{\mathrm{s}}). \tag{27}$$

Matrix Form-Based Performance Metrics Calculating Expressing in matrix form, we can rewrite the performance metrics of the substrate nodes as

$$\mathbf{V} = (1 - \delta)\mathbf{R} + \delta \mathbf{Q}\mathbf{V} \tag{28}$$

where $\mathbf{V} = (\bar{V}(\mathrm{n}_1^{\mathrm{s}}), ..., \bar{V}(\mathrm{n}_i^{\mathrm{s}}), ..., \bar{V}(\mathrm{n}_M^{\mathrm{s}}))^T$, $\mathbf{R} = (\bar{R}(\mathrm{n}_1^{\mathrm{s}}), ..., \bar{R}(\mathrm{n}_i^{\mathrm{s}}), ..., \bar{R}(\mathrm{n}_M^{\mathrm{s}}))^T$, $\mathbf{Q} = [Q(\mathrm{n}_i^{\mathrm{s}}, \mathrm{n}_j^{\mathrm{s}})]$ is an $M \times M$ matrix, and $Q(\mathrm{n}_i^{\mathrm{s}}, \mathrm{n}_j^{\mathrm{s}})$ is defined as

$$Q(\mathrm{n}_i^{\mathrm{s}}, \mathrm{n}_j^{\mathrm{s}}) = \begin{cases} \frac{\bar{R}(\mathrm{n}_j^{\mathrm{s}})}{\sum\limits_{\mathrm{n}_h^{\mathrm{s}} \in N(\mathrm{n}_i^{\mathrm{s}})} \bar{R}(\mathrm{n}_h^{\mathrm{s}})} & \mathrm{n}_j^{\mathrm{s}} \in N(\mathrm{n}_i^{\mathrm{s}}) \\ 0 & \mathrm{n}_j^{\mathrm{s}} \notin N(\mathrm{n}_i^{\mathrm{s}}) \end{cases}. \tag{29}$$

Since the calculation of performance metrics vector could be time-consuming as the size of substrate network becomes large, in this subsection, we develop a simple iterative calculation strategy as: $\mathbf{V}^{(t+1)} = (1-\delta)\mathbf{R} + \delta\mathbf{Q}\mathbf{V}^{(t)}$, where $\mathbf{V}^{(t)}$ is the performance metrics vector at the tth iteration. For initialization, we set $\mathbf{V}^{(0)}$ as: $\mathbf{V}^{(0)} = \mathbf{R}$.

Greedy Strategy-Based Node Mapping Method we can also calculate the performance metrics of virtual nodes by conducting similar procedure. Based on the performance metrics, we rank the substrate nodes and the virtual nodes in non-increasing order and conduct greedy strategy-based node mapping.

Algorithm 1. Recursive method-based performance metrics evaluation algorithm for substrate nodes

Input: Network topology $G^{\mathrm{s}} = (N^{\mathrm{s}}, E^{\mathrm{s}}, A_N^{\mathrm{s}}, A_E^{\mathrm{s}})$, the maximum number of iterations $t_{\max}$, and the maximum tolerance ϵ;
Output: Performance metrics vector $\mathbf{V}$;
1: Calculate $\mathbf{R}$ and $\mathbf{Q}$ according to (21) and (29), respectively;
2: Initialization: $\mathbf{V}^{(0)} = \mathbf{R}$, $t = 0$, $\mu = \infty$;
3: **while** $t < t_{\max}$ and $\mu > \epsilon$ **do**
4: $\quad \mathbf{V}^{(t+1)} = (1-\delta)\mathbf{R} + \delta\mathbf{Q}\mathbf{V}^{(t)}$;
5: $\quad \mu = \left\| \mathbf{V}^{(t+1)} - \mathbf{V}^{(t)} \right\|$;
6: $\quad t = t + 1$;
7: **end while**
8: $\mathbf{V} = \mathbf{V}^{t+1}$.

4.3 Shortest-Path Algorithm-Based Link Mapping Stage

With embedding all virtual nodes of the certain virtual network, the virtual links of the virtual network demand to be embedded. For the link mapping stage of the certain virtual network, we order the virtual links in non-increasing order according to the bandwidth requirement, i.e., $B(\mathrm{e}^{\mathrm{v}}_{k,u,r})$, and then conduct link mapping from the one of the largest bandwidth requirement to that with the least bandwidth requirement.

To map one certain virtual link, we jointly consider substrate link bandwidth consumption, which can be characterized by the hops of the substrate paths, and the substrate link bandwidth capacity.

In this paper, we propose to use the K-shortest-path algorithm to determine the shortest paths between the two corresponding substrate nodes of one virtual link. For convenience, we characterize the substrate network as a weighted graph $G^{\mathrm{s}} = (N^{\mathrm{s}}, E^{\mathrm{s}}, W^{\mathrm{s}})$, where $W^{\mathrm{s}} = \left\{ \mathrm{w}^{\mathrm{s}}_{i,j}, 1 \le i, j \le M, i \ne j \right\}$ denotes the weight set of the substrate links, and $\mathrm{w}^{\mathrm{s}}_{i,j}$ denotes the weight of $\mathrm{e}^{\mathrm{s}}_{i,j}$, which is set to be 1. By applying the K-shortest-path algorithm in G^{s}, the K candidate shortest paths can be obtained. Let $P_{k,u,r,f}$ denote the fth candidate substrate path of $\mathrm{e}^{\mathrm{v}}_{k,u,r}$, $1 \le f \le K$, we then select the substrate path with the largest minimum link bandwidth capacity from $\{P_{k,u,r,f}, 1 \le f \le K\}$, i.e.,

$$f^* = \arg\max \left\{ \min_{f \in \{1,\ldots,K\}} \left\{ B(\mathrm{e}^{\mathrm{s}}_{i,j}), \mathrm{e}^{\mathrm{s}}_{i,j} \in P_{k,u,r,f} \right\} \right\}. \tag{30}$$

5 Performance Evaluation

In this section, we perform numerical simulations to examine the performance of the proposed algorithm. In the simulation, the substrate nodes are uniformly distributed in a square region with the size being Lkm $\times$ Lkm, and any two substrate nodes are connected with the probability of $P = a\exp(\frac{-d^5}{bL})$, where a and b are network characteristic parameters, d is the Euclidean distance between two nodes and L denotes the length of the region, which equals 100. The number of the substrate nodes is chosen as 30 and 40, respectively. The CPU, storage, TCAM capacity of the substrate nodes and the bandwidth capacity of the substrate links are real numbers which are uniformly chosen between 40 and 50. The propagation delay of each substrate link is proportional to the geographical distance between substrate nodes.

For each virtual network, the virtual nodes are uniformly distributed in a square region with the size being Lkm $\times$ Lkm. Similar to the substrate nodes, any two virtual nodes are connected with the probability of $P = a\exp(\frac{-d^5}{bL})$ and the number of virtual nodes in each virtual network is randomly chosen between 4 and 8. The CPU, storage, TCAM demand of the virtual nodes and the bandwidth demand of the virtual links are real numbers uniformly chosen between 5 and 10. Simulation results are averaged over 1000 independent processes involving different simulation parameters.

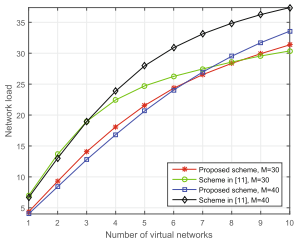

Fig. 2. Network load versus the number of virtual networks

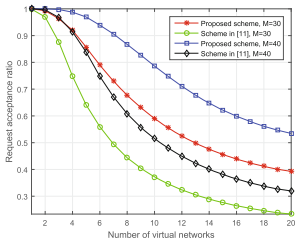

Fig. 3. Request acceptance ratio versus the number of virtual networks

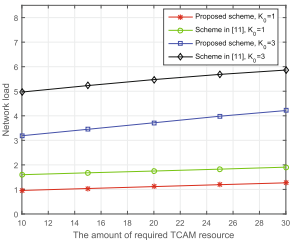

Fig. 4. Network load versus required TCAM resource

Figure 2 shows the network load versus the number of virtual networks. For comparison, we examine the performance of our proposed algorithm and the algorithm proposed in [11]. It can be observed from the figure that the network load increases with the increase of the number of virtual networks for both schemes. This is because the increase in the number of virtual networks results in the increased resource consumption in the substrate network, thus causing the increase of the network load in turn. It can also be seen from the figure that as the number of the substrate nodes increases, the network load decreases. This is

because the available substrate resources increase with the increase of the number of substrate nodes, thus offering better network load performance. Comparing the results obtained from our proposed scheme and the scheme proposed in [11], we can see that our proposed scheme offers better network load performance.

In Fig. 3, we plot the request acceptance ratio versus the number of virtual networks. For comparison, we examine the performance of our proposed algorithm and the algorithm proposed in [11]. It can be seen from the figure that the request acceptance ratio decreases with the increase of virtual networks for both schemes. This is because the consumption of substrate network resources increases with the increase of virtual networks. Comparing the results obtained from our proposed scheme and the scheme proposed in [11], we can see that our proposed scheme can accommodate more virtual networks and obtain higher request acceptance ratio.

In Fig. 4, we consider different number of virtual networks and examine the network load versus the amount of required TCAM resources. It can be seen from the figure that the network load obtained from both schemes increases with the increase of the amount of required TCAM resources and our proposed algorithm can achieve lower network load than the algorithm proposed in [11].

6 Conclusions

In this paper, we study the problem of mapping a number of virtual networks in SDN. To stress the importance of efficient utilization of substrate resources, we formulate the VNE problem as a network load minimization problem. To solve the formulated problem, a two-stage VNE algorithm is then proposed consisting of node mapping stage and link mapping stage. Numerical results demonstrate the effectiveness of our proposed algorithm.

References

1. Chowdhury, N.M.K., Boutaba, R.: Network virtualization: state of the art and research challenges. IEEE Commun. Mag. **47**(7), 20–26 (2009)
2. Mijumbi, R., Serrat, J., Gorricho, J.L., Bouten, N., De Turck, F., Boutaba, R.: Network function virtualization: state-of-the-art and research challenges. IEEE Commun. Surv. Tutor. **18**(1), 236–262 (2016)
3. Fischer, A., de Meer, H.: Generating virtual network embedding problems with guaranteed solutions. IEEE Trans. Netw. Serv. Manag. **13**(3), 504–517 (2016)
4. Su, S., Zhang, Z., Liu, A.X., Cheng, X., Wang, Y., Zhao, X.: Energy-aware virtual network embedding. IEEE Trans. Netw. **22**(5), 1607–1620 (2014)
5. Rodriguez, E., Alkmim, G.P., Fonseca, N., Batista, D.: Energy-aware mapping and live migration of virtual networks. IEEE Syst. J. **11**(2), 637–648 (2017)
6. Chen, X., Li, C., Jiang, Y.: Optimization model and algorithm for energy efficient virtual node embedding. IEEE Commun. Lett. **7**(9), 1327–1330 (2015)
7. Gong, L., Jiang, H., Wang, Y., Zhu, Z.: Novel location-constrained virtual network embedding LC-VNE algorithms towards integrated node and link mapping. IEEE/ACM Trans. Netw. **24**(6), 3648–3661 (2016)

8. Jarray, A., Karmouch, A.: Decomposition approaches for virtual network embedding with one-shot node and link mapping. IEEE/ACM Trans. Netw. **23**(3), 1012–1025 (2015)
9. Gao, X., Ye, Z., Fan, J., Zhong, W., Zhao, Y.: Virtual network mapping for multicast services with max-min fairnessof reliability. IEEE/OSA J. Opt. Commun. Netw. **7**(9), 942–951 (2015)
10. Esposito, F., Paola, D.D., Matta, I.: On distributed virtual network embedding with guarantees. IEEE/ACM Trans. Netw. **24**(1), 569–582 (2016)
11. Huang, H., Li, S., Han, K., Sun, Q., Hu, D., Zhu, Z.: Embedding virtual software-defined networks over distributed hypervisors for vDC formulation. In: IEEE ICC, Paris, pp. 1–6 (2017)