IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 21, 2019, accepted February 27, 2019, date of publication March 7, 2019, date of current version March 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2903476

Virtual Network Embedding Algorithm for Location-Based Identifier Allocation

TIANJIAO CHEN[1], JIANG LIU[1,2], QINQIN TANG[1], TAO HUANG[1,2], AND RU HUO[2]
[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]Beijing Advanced Innovation Center for Future Internet Technology, Beijing University of Technology, Beijing 100124, China

Corresponding author: Jiang Liu (liujiang@bupt.edu.cn)

The work was supported by the National Natural Science Foundation of China under Project 61671086.

ABSTRACT Network virtualization allows multiple isolated virtual networks (VNs) to share the same substrate network (SN). VN embedding (VNE) algorithms can efficiently allocate the limited SN resources to VNs and assign a unique identifier to each VN. However, the fixed bit width of VN identifier in the packet header limits the number of VNs, and extending the bit width leads to the increase of the network traffic. In this paper, we consider the label-combination method to generate VN identifiers by combining the link-grained labels with location information. This method requires the efficient allocation of labels, but the existing VN embedding works only consider the CPU and bandwidth resources. To address this issue, we propose a novel embedding model that considers the label, CPU and bandwidth resource constraints. Furthermore, two window-based heuristic algorithms called VNE-LIA and VNE-iLIA using the greedy algorithm and the proximity principle are presented to solve the VNE problem. The simulation experiments show that our proposed algorithms increase the number of VN identifiers and the revenue to cost ratio under the different resource conditions of SN.

INDEX TERMS Virtual network embedding, virtual network identifier, resource allocation, online algorithm, network virtualization.

I. INTRODUCTION

The rigidity of the current network architecture has resulted in a predicament in deploying new protocols and services [1], and the alterations to the architecture of the Internet will lead to conflicts of interest among multiple competing stakeholders [2]. Therefore, network virtualization is proposed to reconcile this conflict, so that multiple heterogeneous virtual networks (VNs) can share the same substrate network (SN) [3]. In the architecture of the future network, the role of current Internet Service Provider (ISP) is divided into Infrastructure Providers (InPs) and Service Providers (SPs). The multiple SPs create VN requests based on tenant requirements, and then the VN request is implemented on the substrate resources managed by one or more InPs [4], [5]. Eventually, customized end-to-end services can be provided to tenants and different protocols are able to run simultaneously on the same substrate network without interfering with each other. As a result, network virtualization enables the deployment of the VN to be more flexible and fast, and it can significantly reduce the cost of underlying resources [6].

In network virtualization, how to properly allocate SN's resources to VN is realized by the virtual network embedding (VNE) [7]. In the implementation process, the virtual nodes have to be mapped on one or more substrate nodes, and the virtual links need to connect the virtual nodes according to the topology of the VN. A virtual node has multiple choices in the mapping location and a virtual link may consist of multiple substrate links. The resources of the SN are allocated to the VNs when the mapping is successful, so different mapping schemes will bring different resource benefits [8]. Hence the reasonable allocation of resources is essential to improve the number of VNs.

To solve the VNE problem which is NP-hard [2], a number of heuristic-based algorithms have been proposed. The VNE algorithms can be classified into three categories according to the relationship between node mapping and link mapping. *Uncoordinated* VNE algorithms first map the nodes according to different optimization goals, then solve the link mapping problem in a second stage [9], [10].

The associate editor coordinating the review of this manuscript and approving it for publication was Seyedali Mirjalili.

Coordinated VNE algorithms can be achieved in two stages or one stage to improve the overall performance of the embedding. Two stages coordinated VNE algorithms consider the constraints of the link in the node mapping stage [2], [11], [12]. One stage coordinated VNE algorithms simultaneously map virtual nodes and links instead of mapping all the nodes first [13], [14]. *InterInP Coordination* VNE algorithms split the VN request into subgraphs to map to different InPs [15]–[17]. A large number of VNE studies focus on the resource allocation of node CPU and link bandwidth, but recently, many VNE algorithms considering other factors of the network have been proposed. Energy-aware VNE researches [18]–[20] investigate the efficient use of energy to reduce energy consumption across the network. The VNE model in [21] considers the constraints of the storage resource and two heuristic algorithms are proposed to improve the utilization of network resources. NeuroViNE proposed in [22] can preprocess the VN requests by extracting relevant subgraphs to achieve faster and more resource-efficient embeddings.

Another major aspect of network virtualization is to ensure the high isolation between different VNs to guarantee the quality of different service, so the virtual network identifiers (VNIs) are needed to distinguish different VNs. The network isolation technologies with VNI like VLAN [23] have been widely used. In IEEE 802.1Q protocol, the VNI is implemented by setting a unique VLAN ID in the packet header. The VLAN ID field in an Ethernet frame is 12 bits wide. That allows a theoretical maximum of 4094 VLANs in an Ethernet network (numbers 0 and 4095 are reserved) [24]. With the development of new networks such as the Internet of Things, the number of devices and the type of the services in the network has increased significantly [25]. Accordingly, the number of network links and the number of VN requests has also increased. Yet the fixed bit width of the VNI in the packet header limit the total number of VNs that an SN can accept (called tenant capacity). In order to solve this problem, some extension methods of VLAN such as VxLAN and QinQ have been proposed. These methods increase the bit width of the VLAN ID field in the packet header to extend the tenant capacity. However, as the interaction frequency between network devices increases, the packet header also occupies a large part of the traffic. Especially in the Internet of Things, a considerable part of the traffic is the interaction of device status. This part of the data packet carries less data, so the proportion of bits of VNI in the total traffic size becomes larger. Thus, there is a contradiction between the need to reduce the bit width in the header and the need to expand the tenant capacity.

Programming Protocol-Independent Packet Processors (P4) [26] has recently enabled the modification of packet headers, so the bit width of VLAN ID can be flexibly set according to the number of VN requests. Since we expect to reduce the bit width, a better solution to expand tenant capacity is needed. In [27] the VLAN-reusing method has been proposed to distribute the VLAN ID to each substrate link, similar to MPLS and segment routing [28]. And this method makes a VNI consist of a set of VLAN ID with location information. The number of VNIs will be proportional to the number of VLAN IDs and the size of the network. Consequently, this isolation technology with fine-grained VLAN ID provides the possibility of expanding the tenant capacity. The combination of these two solutions can ensure a larger tenant capacity while reducing the VNI bit width in the packet header.

However, the excessive use of a single link who has reduced VLAN ID (called label) still makes the link a bottleneck in some performance [27]. The limited number of labels brings about the problem of label allocation management. Yet the existing heuristic-based algorithms merely consider the allocation of the CPU and the bandwidth (BW) resources. A method is absent that can reasonably distribute the label resources with a link granularity from a global perspective. This absence can result in poor performance of the tenant capacity.

In this paper, we consider a Label-combination method that is the combination of the VLAN-reusing method and the packet header modification. On the condition that a VNI consists of a set of limited labels of different links, we can consider the label a kind of substrate link's resource. Then we introduce the VNE problem for location-based identifier allocation. The goal is to assign the label resources to each VN reasonably, together with the CPU and the BW, to maximize the number of VNI in a specific SN. Since solving a VNE problem is known to be *NP*-hard even in the offline case, we propose two heuristic algorithms, the VNE algorithm for location-based identifier allocation (VNE-LIA) and its improved algorithm (VNE-iLIA), to allocate the label resource. Based on the greedy algorithm, VNE-LIA considers the constraints of three resources simultaneously to make node selection. When the node mapping is complete, we use the k-shortest paths to map the link. Hence the VNE problem can be solved in polynomial time. Since the label is a type of link resource, we can save label resources by reducing the number of times the substrate link is mapped. We use the proximity principle to improve VNE-LIA then propose the VNE-iLIA to further increase tenant capacity. The VNE-LIA and the VNE-iLIA are window-based algorithms which can handle virtual network requests in batches. Experiment results show that these two algorithms are more suitable for the network with more links.

Major contributions of this paper are as follows:

- We propose a Label-combination method in SDN and P4 environment to ensure a larger tenant capacity while reducing the VNI bit width in the packet header. We take the link-grained label as a kind of SN resource and formulate a VN embedding model with the CPU resources of nodes and the BW/label resources of links.
- Two online VN embedding algorithms based on the greedy algorithm and proximity principle are proposed to solve the location-based identifier allocation problem.

• We perform extensive simulation studies to verify the performance of VNE-LIA and VNE-iLIA. Our proposed algorithms increase the tenant capacity and R/C ratio under the different resource conditions of SN.

The rest of this paper is organized as follows. In Section II, we present the Label-combination method to illustrate the VNI combination of labels based on link granularity and describe the network model and the objectives. The problem statement is described and two heuristic algorithms for location-based identifier allocation are introduced in Section III. In Section IV, the simulation results are analyzed. Finally, we conclude our work in Section V.

II. NETWORK MODEL

In this section, we first describe the Label-combination method, including the structure of location-based identifier and its implementation in SDN. Then we describe the network models based on the CPU, BW and label resources. Finally the objectives are introduced to evaluate the performance of the VNE.

A. LABEL-COMBINATION METHOD

Label-combination method is a flexible way to set the virtual network identifier. It includes two aspects: one is the flexible bit width setting of the identifier in the packet header, and another is the flexible combination of the location-based identifiers which are named label.

In Label-combination method, the VNI consists of a set of labels which contain the label identifier (LID) and the location information (the device port). The representation of VNI in SDN controller is as follows.

$$VNI = \{label(LID\ n,\ port\ i)\}$$

Each label occupies only one LID, and n is an integer in $[0, LID\_MAX]$. LID_MAX is set by bit width of LID.

LID is a field in the packet header used to distinguish different virtual networks. Unlike VLAN, the LID's bit width is not fixed by a certain protocol but is set by P4 according to requirements. P4 can set the bit width of the LID when defining the packet header and parse it without the constraint of network protocols. This method can reduce the bit width of identifier to less than 12 bits and save the packet header traffic in the network. This paper focuses on how to expand the tenant capacity under a small fixed number of labels, hence the details of packet header modification will not be elaborated.

Different from the traditional usage of VLAN to isolate broadcast domain, the LID carried in different packet headers in the same VN can be different, and the LID in different VN can be the same. Therefore, distributed networks are difficult to handle this situation. We use the SDN controller to centrally manage the labels to handle the network slice, instead of being configured distributed, which means the label can be used quite similar to the tag switch mode. When a packet arrives from *port i*, the controller can determine which VN the packet belongs to based on its *LID n*. Then the controller selects another port of the same VN according to the destination address and rewrites the LID for forwarding. Rewriting the LID will be the standard action of the data plane in SDN, which means that the Label-combination method will be implemented quickly.

Label-combination method sets up the VNI by combining labels and it has the ability to extend the tenant capacity with minor label bit width in the packet header. When setting a LID for *port n*, an ILD is also set for the other *port m* on the link(m, n). So we just need to ensure that there are no virtual networks using the same label on the same substrate link. The network model will be introduced in detail in the remaining subsections.

B. SUBSTRATE NETWORK

We describe the substrate network as an undirected graph $G^S = (N^S, L^S)$, where N^S represents the set of substrate nodes and the N^S the set of substrate links. A substrate node $n^S \in N^S$ has a CPU capacity cpu(n^S). Each substrate link $l^S(x, y) \in L^S$ between n_x^S and n_y^S is associated with the bandwidth capacity bw(l^S) and the LID set $D(l^S) = \{n \in \mathbb{N} \mid n \leq LID\_MAX\}$. P^S denotes the set of loop-free paths in the substrate network.

The lower half of Fig. 1 shows a substrate network, where the black numbers in rectangles represent available CPU resources and the black numbers over the links represent available bandwidth and available LID resources.

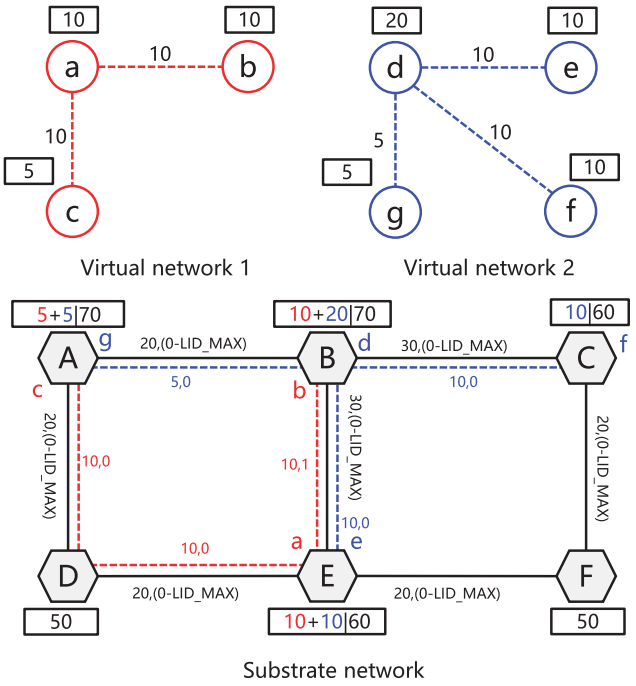


FIGURE 1. Example of VN and SN.

C. VIRTUAL NETWORK REQUEST

The virtual network is modeled as an undirected graph $G^V = (N^V, L^V)$, where N^V and L^V respectively denote the sets

of virtual nodes and links. N^V denotes the lifetime. Each VN request requires a certain amount of resources under the QoS constraints. We denote cpu(n^V) as the capacity constraint of virtual node $n^V \in N^V$, and bw(l^V) as the bandwidth capacity constraint of virtual link $l^V(x, y) \in L^V$ between n_x^V and n_y^V. Besides, each VN needs a unique VN identifier for network isolation.

The upper half of Fig. 1 shows two virtual networks, where the numbers in rectangles represent required CPU resources and the numbers over the links required available bandwidths.

D. VIRTUAL NETWORK EMBEDDING

Virtual network embedding problem mainly focuses on efficiently allocating SN resources to VNs along with node and link constraints. However, when the location-based identifier is considered, the elements are extended to three. Therefore, we propose a new virtual network embedding model with nodes, links and labels.

To be more specific, we will discuss the differences between the VLAN ID and the label. In the case of VLAN, the node and link are relevant to each other in the mapping procedure, since the virtual link is constraint by the position of the two end virtual nodes. But the VLAN ID is irrelevant to the other two factors in most scenarios. In such a condition, the VLAN ID will be embedded alone and valid in the whole substrate network. In contrast, the idea of Label-combination method exploits the centralized SDN controller to recognize a virtual network with a set of labels, which will increase the tenant number since the LID can be reused in different locations. The virtual network embedding algorithm is also affected, which means the labels are embedded relevant to the links. To be specific, if a virtual link is mapped to the substrate path, the embedding algorithm should make sure that the LID of the substrate link is also available.

Virtual network embedding $M(G^V) : G^V \rightarrow G^S$ can be divided into node and link mapping as follows.

(1)Node mapping: Each virtual node in the same VN is mapped to a different substrate node and the virtual nodes from different VNs can be mapped to the same substrate node. The node mapping function is denoted by $M(N^V) : N^V \rightarrow N^S$. Meanwhile, the substrate node allocates CPU resources to the virtual node. The residual resource capacity of substrate node n^S is denoted as

$$R_{cpu}(n^S) = cpu(n^S) - \sum_{n^V \rightarrow n^S} cpu(n^V) \tag{1}$$

As shown in Fig. 1, two tenants are embedded in the same substrate network with nodes, links and the labels. The substrate network has six nodes: A-F. The VN of tenant 1 has the node mapping $\{a \rightarrow E, b \rightarrow B, c \rightarrow A\}$. The VN of tenant 2 has the node mapping $\{d \rightarrow B, e \rightarrow E, f \rightarrow C, g \rightarrow A\}$.

When a virtual node n^V from next VN attempts to be mapped to substrate node n^S, the resource of n^V is subject to

$$cpu(n^V) \leq R_{cpu}(M(n^V)) \tag{2}$$

(2)Link mapping: Each virtual link is mapped to a loop-free substrate path(unsplittable flow) according the result of node mapping. The node mapping function is denoted by $M(L^V) : L^V \rightarrow P^S$. Meanwhile, the substrate link allocates BW and LID resources to the virtual link. The residual BW and LID resource capacity of substrate link l^S are respectively denoted as

$$R_{bw}(l^S) = bw(l^S) - \sum_{l^V \rightarrow l^S} bw(l^V) \tag{3}$$

$$R_D(l^S) = D(l^S) \setminus \bigcup_{l^V \rightarrow l^S} D(l^V) \tag{4}$$

where $D(l^V)$ denotes the set of LID used by l^S and has only one element because each virtual link uses only one LID of the substrate link. In the virtual network of tenant 1 in Fig. 1, virtual link *a-b* is mapped to substrate link *E-B* assigned with *LID 1*, virtual link *a-c* is mapped to substrate link *E-D-A* assigned with *LID 0*. In the virtual network of tenant 2, virtual links *d-e*, *d-f*, *d-g* are mapped to substrate links *B-E*, *B-C*, *B-A* assigned with *LID 0* respectively. As for substrate link *B-E*, *LID 0* and *1* are occupied now, so *LID 2-LID_MAX* are available for next tenants.

When a virtual link from next VN attempts to be mapped to substrate link l^S, the resource of l^V is subject to

$$bw(l^V) \leq R_{bw}(M(l^V)) \tag{5}$$

$$card(R_D(M(l^V))) \geq 1 \tag{6}$$

where $card(D)$ denotes the number of the elements in set D. In addition to meeting the bandwidth constraints, link mapping needs to ensure that at least one LID on the substrate link can be allocated.

E. OBJECTIVES

In order to evaluate the performance of the virtual network embedding algorithms, the revenue, cost and acceptance ratio are proposed to maximize the profit of InPs.

The *revenue* of the embedding action is defined according to the virtual request as follows:

$$Rev(M(G^V)) = \alpha_R \sum_{l^V \in L^V} bw(l^V) + \beta_R \sum_{n^V \in N^V} cpu(n^V) \tag{7}$$

where α_R and β_R are the weights for bandwidth and CPU requirements, respectively.

The *cost* is defined by resources used for the virtual request in the substrate network:

$$Cost(M(G^V)) = \alpha_C \sum_{l^V \in L^V} hop(l^V) bw(l^V) + \beta_C \sum_{n^V \in N^V} cpu(n^V) \tag{8}$$

where $hop(l^V)$ denotes the number of substrate links in P^S chosen by $M(l^V)$. α_C and β_C are the weights for bandwidth and CPU costs, respectively.

The R/C *ratio* indicates the utilization of substrate network resource and is defined by revenue and cost as follows:

$$R/C = \frac{Rev(M(G^V))}{Cost(M(G^V))} \tag{9}$$

The *acceptance ratio* is defined by the following formulation, which means how many requests are successfully embedded from all the requests.

$$AR = \frac{Sum_{AVN}}{Sum_{VN}} \tag{10}$$

where Sum_{AVN} denotes the number of already accepted VN requests, Sum_{VN} denotes the number of arrived VN requests.

III. GREEDY HEURISTIC ALGORITHM FOR LOCATION-BASED IDENTIFIER ALLOCATION

In this section, we first describe the problem statement for allocating labels, while analyzing the tenant capacity of the system. Then the VNE algorithm for location-based identifier allocation and its improved algorithm are proposed to solve the label allocation problem.

A. PROBLEM STATEMENT

Label-combination method identifies VNs by combining the location based identifiers. The number of VNIs is proportional to the number of labels and the size of the network, then the tenant capacity is expanded. We first analyze the boundary of the number of VNIs.

We assume there are k links in the substrate network ($l_i^S \in L_i^S, 1 \le i \le k$), and $card(D(l_i^S))$ denotes the amount of VN requests that the link l_i^S can handle. Since the bit width of LID field is limited, $card(D(l_i^S))$ is subject to

$$card(D(l_i^S)) \le LID\_MAX + 1 \tag{11}$$

Fig. 2(a) shows the most ideal situation. We assume there are four nodes *A, B, C, D* and four links *A-B, A-C, B-D, C-D* in the substrate network. When each VN request has only one virtual link and is mapped onto one single substrate link, the substrate links is fully utilized and the tenant capacity reaches the maximum. As for all the k substrate links in the network, we can get the tenant capacity TC as follows:

$$TC \le \sum_{i=1}^{k} card(D(l_i^S)) = k * (LID\_MAX + 1) \tag{12}$$

However, in reality, the topology of a virtual network is usually random, and the number of nodes and links are greater than one, so it is difficult to achieve maximum tenant capacity. Therefore, the virtual network mapping algorithm is needed to perform reasonable link and node placement and resource allocation.

Fig. 2(b) shows a bad situation. When the substrate link *A-B* is overused due to an unreasonable mapping policy, it will affect the acceptance success rate of the following VN request. Thus we expect to reduce the number of bottleneck links like *A-B* through virtual network embedding algorithms.

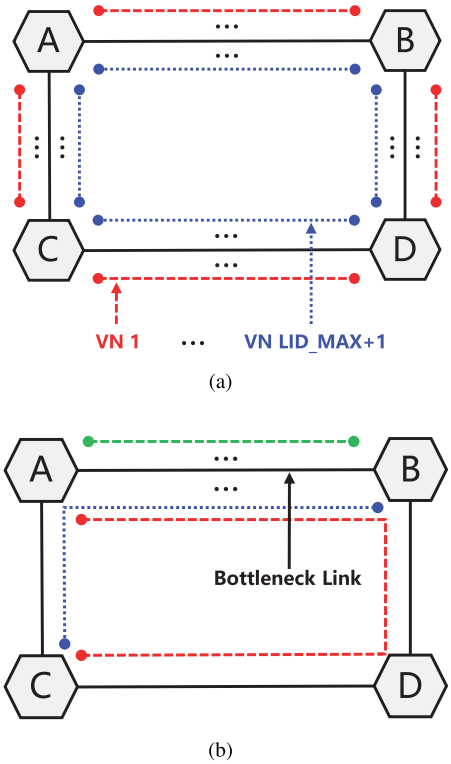


FIGURE 2. The performance upper bound and bottleneck. (a) Upper bound of tenant capacity. (b) Bottleneck of tenant capacity.

B. VNE ALGORITHM FOR LOCATION-BASED IDENTIFIER ALLOCATION

In order to minimize the number of bottleneck links in the SN, an appropriate label resource allocation algorithm needs to be adapted to even increase the number of LIDs used on each substrate link.

Because the VNE problem is NP-hard, a number of heuristic-based algorithms have been proposed. A baseline VNE algorithm proposed in [29] takes into account the limited resources of the CPU and BW and uses a greedy algorithm to select the substrate node. In this case, the virtual node will be preferentially mapped to the substrate node with larger resources, thus successfully minimizing the number of bottleneck nodes and links. In the Label-combination method, LID of label is also a kind of substrate link resource. Based on these two ideas, we propose the virtual network embedding algorithm for location-based identifier allocation (VNE-LIA) as follows:

In the node mapping stage, we use the greedy algorithm to choose the substrate node which has the biggest available resource. We define substrate node available resource (SAR) of a substrate node as follows:

$$SAR(n^S) = R_{cpu}(l^S) \sum_{l^S \in L(n^S)} (R_{bw}(l^S) + card(R_D(l^S))) \tag{13}$$

where $card(R_D(l^S))$ denotes the number of l^S's available LIDs. Each substrate link has LID_MAX+1 available LIDs at first. SAR comprehensively considers the remaining

CPU resource of n^S and BW/LID resource of all adjacent links of n^S.

Similar to SAR, we also define virtual node request resource (VRR) of a substrate node as follows:

$$VRR(n^V) = cpu(n^V) \sum_{l^V \in L(n^V)} (bw(l^V) + 1) \quad (14)$$

In the link mapping stage, we use the k-shortest paths algorithm to choose a path for a virtual link between two nodes. The algorithm checks if all the substrate links in the path have available LIDs, meanwhile check whether the bandwidth of each substrate link meets the constraint of virtual link. If the path meets the constraints, assign one remained LID to the virtual link on each substrate link, and update the SAR of the substrate network. When a virtual network is finished, the occupied LID and other resources will be released.

Besides, considering the ease of migration of the algorithm to the actual mapping system, we use a window-based VNE framework [29] which is an admission mechanism to batch process VN requests. At the beginning of the time window, existing VN requests are sorted based on revenue in decreasing order and placed in the request queue. Then we use the VNE algorithm to map the VN to the SN in order. When the node mapping or link mapping fails, this VN request will be placed in the postpone queue and remapped with the next batch of VN requests in the next time window. To prevent requests from being continually remapped, the remapping threshold T_{try} is used to limit the number of remappings. When the threshold is reached, the virtual network will be rejected and will no longer be processed. Under normal circumstances, the tenant's lease time for the virtual network is limited, so each virtual network will have a lifetime T_{life}. T_{try} and T_{life} are in the unit of the time window. At the end of the time window, the VN requests that reach T_{life} will leave and the resources they occupy will be released. The node and link mapping algorithms are detailed in Algorithm 1.

C. IMPROVED VNE ALGORITHM FOR LOCATION-BASED IDENTIFIER ALLOCATION

VNE-LIA considers three resources of CPU, BW and LID, and maps the virtual network to substrate nodes and links with relatively abundant resources so that the number of bottleneck nodes and links is minimized. However, when the residual resource capacity of substrate nodes are unbalanced, the adjacent virtual nodes will be likely to be mapped to two resource-rich but far-reaching substrate nodes. In this situation, the loop-free substrate path mapped by one virtual link contains two or more substrate links, which means a large part of bandwidth and LID resource is wasted.

In [30], the proximity principle of nodes is proposed. In the node mapping process, when a virtual node has been mapped to a substrate node n_0^S, other virtual nodes in this VN will be mapped to the substrate node which is closer to n_0^S, which will greatly save substrate link resources. In Label-combination method, LID is also a kind of link resource. The proximity principle can reduce the amount of LID usage, thus minimizing the bottleneck link in the SN.

As shown in Fig. 3, one virtual network needs to be mapped to a substrate network. When using the greedy algorithm without proximity principle, the nodes of VN are mapped to the nodes E, B and A because these substrate nodes have larger available resources. Finally, two virtual links are mapped on three substrate links. The BW resource and the LID are wasted. When the proximity principle is adopted, the VN is mapped to the E-B and E-D. Only two substrate links are

Algorithm 1 VNE Algorithm for Location-Based Identifier Allocation

Begin

Node Mapping Begin: Sort the VN requests by revenues in non-increasing order in the current time window.
for all the sorted VN requests **do**
  Push all the substrate nodes into a set Ω
  Sort the virtual nodes by $VRR(n^V)$ in non-increasing order
  for all the unmapped virtual nodes in the VN request **do**
    if $\Omega \neq \emptyset$ **then**
      choose n^V with the greatest $VRR(n^V)$
      calculate $SAR(n^S)$ of all the substrate nodes in Ω
      choose n^S with the greatest $SAR(n^S)$
      if $R_{cpu}(n^S) \geq cpu(n^V)$ **then**
        $M(n^V) = n^S$
        $\Omega \leftarrow \Omega \setminus \{n^S\}$
      else
        node mapping fail, defer this VN request, and store it in the request queue. BREAK.
      end if
    end if
  end for
end for
Link Mapping Begin: Sort the node-mapping-succeed VN requests by revenues in non-increasing order.
for all the sorted VN requests **do**
  for all the unmapped virtual links in the VN quest **do**
    search the k-shortest paths for increasing k
    if There is a substrate path that meets the virtual link's bandwidth constraint and each substrate link in this path has available LIDs **then**
      $M(l^V) = P^S$
    else
      link mapping fail, defer this VN request, and store it in the request queue. BREAK.
    end if
  end for
end for

end

occupied, and a part of link resources are saved to support more virtual networks.

In this section, we optimize the VNE-LIA with the proximity principle and propose the improved VNE algorithm for location-based identifier allocation (VNE-iLIA).

In the node mapping stage, we also use the greedy algorithm to choose the substrate node which has the biggest available resource. However, we re-define the SAR as the weighted substrate node available resource (WSAR) as follows

$$WSAR(n^S) = Corr^m R_{cpu}(l^S) \sum_{l^S \in L(n^S)} (R_{bw}(l^S) + card(R_D(l^S))) \tag{15}$$

where $Corr$ in $(1, \infty)$ is the weight of proximity principle, and the coefficient m of n^S is the number of the substrate nodes which is not only already mapped in the same request, but also directly connected to n^S. As shown in Fig. 3, virtual node a is first mapped on the substrate node E. When mapping virtual node b, the m of node B, D and F is 1, and the m of A and C is 0. This procedure can make sure that if some substrate nodes are already selected by this VN, the nodes connected to them will have a better chance to be selected.

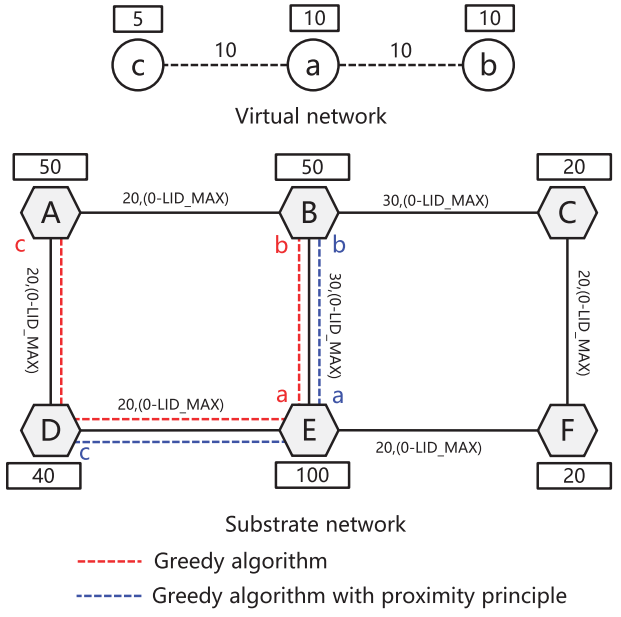


FIGURE 3. The advantage of proximity principle.

The link mapping algorithm of VNE-iLIA is the same as that of VNE-LIA. The improved node mapping algorithms of VNE-iLIA are detailed in Algorithm 2.

D. TIME COMPLEXITY ANALYSIS

For each VN request, the time complexity of the greedy node mapping algorithm is $O(N_S \times N_V)$, where the N_S denotes the number of substrate nodes and N_V denotes the number of virtual nodes. For each virtual link, the time complexity of the k-shortest paths algorithm is $O(E_S + N_S log N_S + k)$ when the SN has E_S links [29]. For a virtual network with E_V links, the time complexity of the link mapping stage is $O(E_V \times (E_S + N_S log N_S + k))$. Link mapping is performed after the node mapping is completed. Consequently, for a time window with N_{vn} sorted VNs, the time complexity is $O(N_{vn} \times E_V \times (E_S + N_S log N_S + k))$.

Algorithm 2 Node Mapping Stage of Improved VNE Algorithm for Location-Based Identifier Allocation

Begin

Sort the VN requests by revenues in non-increasing order in the current time window.
for all the sorted VN requests **do**
  for all the substrate nodes of SN **do**
    $m \leftarrow 0$
  end for
  Push all the substrate nodes into a set Ω
  Sort the virtual nodes by $VRR(n^V)$ in non-increasing order
  for all the unmapped virtual nodes in the VN request **do**
    if $\Omega \neq \emptyset$ **then**
      choose n^V with the greatest $VRR(n^V)$
      calculate $WSAR(n^S)$ of all the substrate nodes in Ω
      choose n^S with the greatest $WSAR(n^S)$
      if $R_{cpu}(n^S) \geq cpu(n^V)$ **then**
        $M(n^V) = n^S$
        $\Omega \leftarrow \Omega \setminus \{n^S\}$
        update m of each substrate node in Ω which connected to n^S
      else
        node mapping fail, defer this VN request, and store it in the request queue. BREAK.
      end if
    end if
  end for
end for
end

IV. SIMULATION AND ANALYSIS

A. SIMULATION ENVIRONMENT SETTINGS

We have implemented a VNE simulator based on the ViNE-Yard platform [2] to evaluate the advantages of VNE-LIA and VNE-iLIA. The SN topology is randomly generated with 50 nodes and 141 links by the GT-ITM tool [31]. The CPU capacity of substrate nodes and the BW capacity of substrate links follow a uniform distribution from 50 to 100 units, respectively. For each VN, the number of virtual nodes follows a uniform distribution from 2 to 10, and each pair of substrate nodes are connected with probability 0.5. The arrivals of VN requests are modeled by a Poisson process with mean of 50 requests per time window. The remapping threshold T_{try} is 1. The lifetime of VN request T_{life}

is set large enough to ensure that no VN request is released during the simulation, so at the end of the simulation, we can take the acceptance rate AR as a measurement of tenant capacity. For the VNE-iLIA, the distance weight *Corr* is 2.

The previous work did not consider the label source of the link, so our comparison algorithm is a two-step algorithm using VLAN. It uses greedy algorithms for node selection considering the CPU and BW constraints and uses k-shortest paths to solve the link mapping problem. The identifier of the VN occupies only one VLAN ID in the entire network.

B. PERFORMANCE SIMULATION RESULTS

1) TENANT CAPACITY COMPARISON

The proposed solution in this paper is to reduce the number of rejections of virtual network requests caused by limited bit width of labels. In order to focus on evaluating the benefits of VNE for location-based identifier allocation and its improved algorithm on tenant capacity expansion, the VN request rejection due to insufficient CPU and BW should be avoided. So we set $cpu(n^V)$, $bw(l^V)$ as 0.1% of $cpu(n^S)$ and $bw(l^S)$. Then we keep track of AR during 1000 time windows to guarantee the stability and repeatability of the results.

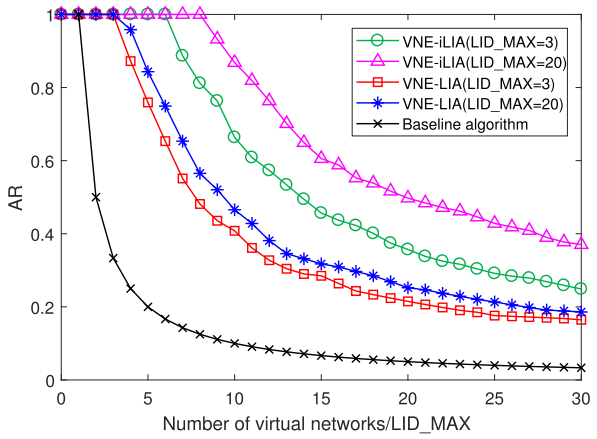


FIGURE 4. Acceptance ratio under different algorithms and different LID_MAX.

The request acceptance ratio curves of different algorithms under different LID_MAX is shown in Fig.4. In this experiment, LID_MAX is not set to a fixed value because it is flexibly set according to the number of VN requirements. To assess the extent of tenant capacity expansion, we use the ratio of VN number to label number (V2L ratio). When the acceptance rate drops from 100%, it indicates that some virtual networks are rejected because of the limitation of the label resource. As VN requests continue to come, label resources are gradually exhausted. Moreover, the AR will finally approach 0 because no virtual network is released.

The AR of baseline algorithm begins to decline when V2L ratio is equal to 1, because Label-combination method is not used and the number of accepted VNs is equal to the bit width of the identifier in the packet header. Then no VN can be accepted when the V2L ratio is greater than 1. The V2L ratio of VNE-LIA and VNE-iLIA when AR begins to drop is greater than that of baseline algorithm. Furthermore, the extent of RA decline of VNE-LIA and VNE-iLIA is more gradual. When VN begins to be rejected, it does not indicate that all the label resources in the substrate network are exhausted. The rejection may be because the remaining links with available labels cannot satisfy the topology of the virtual network so that some VN requests can still be accepted. In general, VNE-LIA can make the substrate network accept more virtual networks, and VNE-iLIA can further expand the tenant capacity.

It can be seen from Fig.4 that V2L ratio when AR begins to decline can reflect the size of the tenant capacity to some extent. So we call the V2L ratio at this time the multiple of tenant capacity expansion (TCE multiple). The relationship between TCE multiple and LID_MAX (or network size) will be discussed in the next experiment.

2) PERFORMANCE ON DIFFERENT SUBSTRATE NETWORKS

In this simulation, We explore the effect of network size and LID_MAX on the ability of algorithms to extend tenant capacity. TCE multiple is used to evaluate the expansion effect of tenant capacity. We conduct three experiments to investigate the impact of the number of nodes, links and labels per link on TCE multiple. The $cpu(n^V)$, $bw(l^V)$ are still 0.1% of $cpu(n^S)$ and $bw(l^S)$ to eliminate the impact of limited CPU and BW resources on the results.

In order to evaluate the effect of the number of substrate links on TCE multiple, the number of substrate nodes is set to 50, and the number of substrate links varies from 50 to 450. For each SN size, we set up different LID_MAX as (10,20,30,40,50,60,70,80,90,100) for testing. As shown in Fig.5, the error bar of each link number presents the range of TCE multiple values of different LID_MAX. The average TCE multiple of VNE-LIA and VNE-iLIA grows as the number of links grows. This is because in the Label-combination method, the identifier of VN is a combination of labels of multiple substrate links. Therefore, the number of identifiers will be positively related to the number of links.

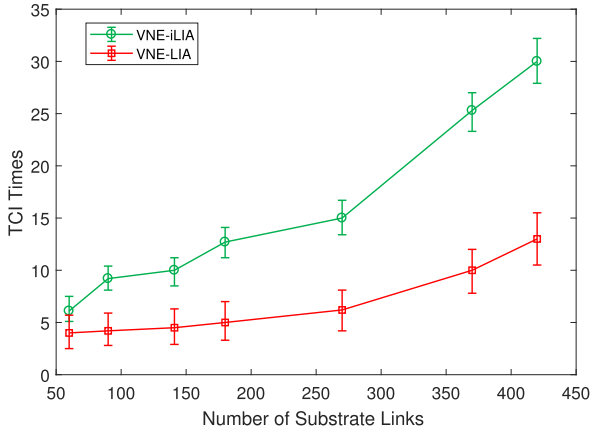


FIGURE 5. Effect of substrate link number on TCE multiple.

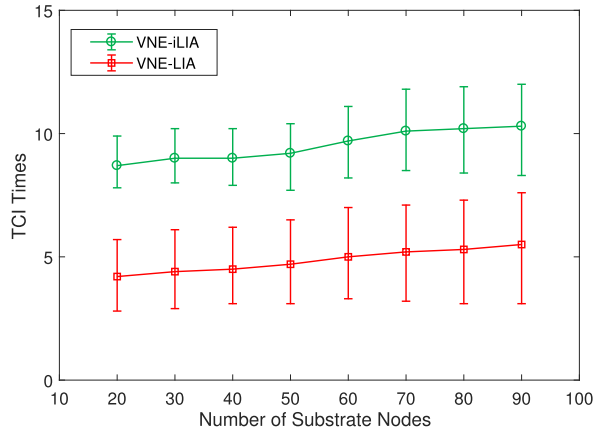


FIGURE 6. Effect of substrate node number on TCE multiple.

Besides, VNE-iLIA can make SN accommodate more VNs than VNE-LIA. The expansion capability is not obvious when the number of links is small because the upper limit of the tenant capacity is small. However, when the number of links is large, the principle of proximity is more obvious to the improvement of TCE multiple since it saves more link resources.

In order to evaluate the effect of the number of substrate nodes on TCE multiple, the number of substrate links is set to 141, and the number of substrate nodes varies from 20 to 90. The setting of LID_MAX for each SN is the same as experiment 1. As shown in Fig.6, the average TCE multiple of VNE-LIA and VNE-iLIA grows as the number of links grows, but the growth trend is more gradual than experiment 1. This can indicate that the increase of node resources can bring about an increase in tenant capacity to a certain extent, but the ability to expand is limited.

In order to evaluate the effect of the number of LID_MAX on TCE multiple, the number of substrate links is set to 141, and the number of substrate nodes is set to 50. As shown in Fig.7, the average TCE multiple of VNE-LIA and VNE-iLIA grows as the LID_MAX grows, and converges to a steady point, for the reason that the definition of TCE multiple weakens the influence of LID_MAX. Compared with substrate links, the impact from LID_MAX is limited.

In general, compared with the number of substrate nodes and labels, the number of substrate links is the major factor of TCI Times. So VNE-LIA and VNE-iLIA have better effects on dense networks, especially those with a large number of links.

3) EFFECT OF INCREASING VN REQUEST'S CPU AND BW ON R/C

In this simulation, we intend to figure out the impact of different algorithms on R/C in different conditions. This experiment focuses on the effectiveness of different algorithms for the allocation of CPU and BW resources, so LID_MAX is set as 4096 so that labels will not become restricted resources. We record the R/C during 500 time windows to ensure the system to enter a steady state, and the average duration of each virtual network is 10 time windows.

In order to evaluate the effect of increasing VN request's CPU, the $bw(l^V)$ is set as 50% of $bw(l^S)$ and the $cpu(n^V)$ increases from 10% to 90% $cpu(l^S)$. As shown in Fig.8, baseline algorithm and VNE-LIA have similar resource utilization

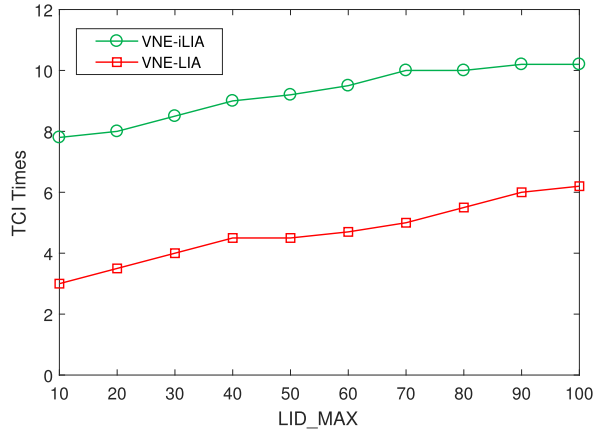


FIGURE 7. Effect of LID_MAX number on TCE multiple.

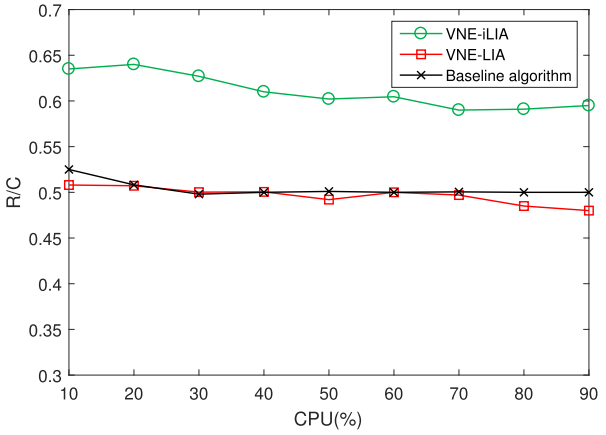


FIGURE 8. Effect of virtual nodes' CPU on R/C.

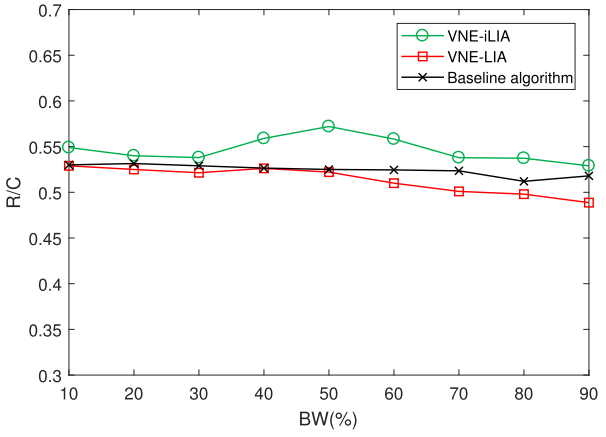


FIGURE 9. Effect of virtual links' BW on R/C.

rates for the SN, which is reflected by the R/C ratio. And the VNE-iLIA can make much better use of SN resources.

In order to evaluate the effect of increasing VN request's BW, the $cpu(l^V)$ is set as 50% of $cpu(l^S)$ and the $bw(n^V)$ increases from 10% to 90% $bw(l^S)$. As shown in Fig.9, the difference between R/C of baseline algorithm and VNE-LIA is slight, but R/C of VNE-iLIA increases when $bw(n^V)$ increases from 30% to 50% of $bw(l^S)$ because the advantage of the proximity principle is more obvious when $bw(n^V)$ is moderate.

V. CONCLUSION

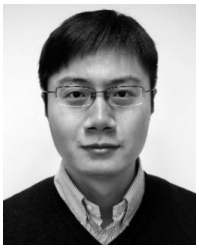
In this paper, we investigate the location-based identifier allocation problem of virtual network embedding. We consider a new way called Label-combination method to generate virtual network identifiers by combining the link-grained labels with their location information, and we analyze the upper bound of the tenant capacity and the reasons for the performance bottleneck. Our objective is to maximize the tenant capacity with a fewer bit width of the label in the packet header. Based on the greedy algorithm and the proximity principle, we propose two window-based online allocation algorithms called VNE-LIA and VNE-iLIA to allocate the labels together with the CPU and BW resources. We also conducted extensive simulations and our results show that the proposed algorithms can increase tenant capacity and R/C ratio under the different resource conditions of substrate networks.

In future work, we will continue to investigate other issues in the VNE model considering location-based identifier allocation. In the next step, we plan to further explore the impact of specific network topologies on our algorithms' performance. In addition, the imbalance of bandwidth and label resource usage on the same link can also make this link a bottleneck, so we should further optimize the algorithms according to the current usage of the substrate link resources and the request of virtual link resources. Finally, we will integrate the algorithm with a real SDN system.

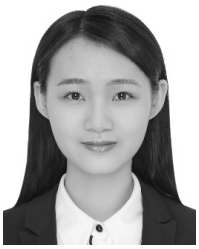
REFERENCES

[1] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, no. 5, pp. 862–876, Apr. 2010.
[2] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.
[3] M. F. Bari *et al.*, "Data center network virtualization: A survey," *IEEE Commun. Surv. Tuts.*, vol. 15, no. 2, pp. 909–928, 2nd Quart., 2013.
[4] S. R. Chowdhury *et al.*, "ReViNE: Reallocation of virtual network embedding to eliminate substrate bottlenecks," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 116–124.
[5] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 358–380, Mar. 2015.
[6] D. Qiang, N. Ansari, and M. Toy, "Software-defined network virtualization: An architectural framework for integrating SDN and NFV for service provisioning in future networks," *IEEE Netw.*, vol. 30, no. 5, pp. 10–16, Sep./Oct. 2016.
[7] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quart., 2013.
[8] S. R. Chowdhury *et al.*, "Protecting virtual networks with DRONE," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2016, pp. 78–86.
[9] P. Zhang, H. Yao, and Y. Liu, "Virtual network embedding based on the degree and clustering coefficient information," *IEEE Access*, vol. 4, pp. 8572–8580, 2016.
[10] H. Cao, Y. Zhu, L. Yang, and G. Zheng, "A efficient mapping algorithm with novel node-ranking approach for embedding virtual networks," *IEEE Access*, vol. 5, no. 1, pp. 22054–22066, 2017.
[11] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 1–9.
[12] S. Haeri and L. Trajković, "Virtual network embedding via monte carlo tree search," *IEEE Trans. Cybern.*, vol. 48, no. 2, pp. 510–521, Feb. 2018.
[13] F.-T. Hsu and C.-H. Gan, "Resource allocation with spectrum aggregation for wireless virtual network embedding," in *Proc. IEEE 82nd Veh. Technol. Conf.*, Sep. 2015, pp. 1–5.
[14] Y. Yuan, C. Wang, B. Zhang, S. Zhu, and N. Zhu, "A novel algorithm for embedding dynamic virtual network request," in *Proc. 2nd Int. Conf. Inf. Sci. Control Eng.*, Apr. 2015, pp. 28–32.
[15] S. Gong, C. Jing, X. Yin, and Q. Zhu, "Survivable virtual network embedding across multiple domains," in *Proc. 2nd IEEE Int. Conf. Comput. Commun.*, Oct. 2016, pp. 2391–2396.
[16] K. Guo, Y. Wang, X. Qiu, W. Li, and A. Xiao, "Particle swarm optimization based multi-domain virtual network embedding," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 798–801.
[17] M. Shen, K. Xu, K. Yang, and H.-H. Chen, "Towards efficient virtual network embedding across multiple network domains," in *Proc. IEEE 22nd Int. Symp. Quality Service*, May 2014, pp. 61–70.
[18] S. Jia, G. Jiang, P. He, and J. Wu, "Efficient algorithm for energy-aware virtual network embedding," *Tsinghua Sci. Technol.*, vol. 21, no. 4, pp. 407–414, Aug. 2016.
[19] M. H. Dahir, H. Alizadeh, and D. Gözüpek, "Energy efficient virtual network embedding in federated software defined networks," in *Proc. 25th Signal Process. Commun. Appl. Conf. (SIU)*, May 2017, pp. 1–4.
[20] V. Lira, E. Tavares, M. Oliveira, Jr., E. Sousa, and B. Nogueira, "Virtual network mapping considering energy consumption and availability," *Computing*, no. 2, pp. 1–31, 2018. [Online]. Available: https://link.springer.com/article/10.1007/s00607-018-0620-y
[21] P. Zhang, H. Yao, and Y. Liu, "Virtual network embedding based on computing, network, and storage resource constraints," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3298–3304, Oct. 2018.
[22] A. Blenk, P. Kalmbach, J. Zerwas, M. Jarschel, S. Schmid, and W. Kellerer, "NeuroViNE: A neural preprocessor for your virtual network embedding algorithm," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 405–413.
[23] J. Wang, L. Sun, X. Jiang, and Z. Wu, "IGMP snooping: A VLAN-based multicast protocol," in *Proc. IEEE 5th Int. Conf. High Speed Netw. Multimedia Commun.*, Jul. 2002, pp. 335–340.
[24] S. Kikuchi, Y. Imai, K. Fukui, and S. Kotabe, "A network virtualization method using l2-tunneling for cloud data centers and its evaluation," *IEICE Tech. Rep.*, vol. 110, no. 167, pp. 43–48, Aug. 2010.
[25] O. Bello and S. Zeadally, "Intelligent device-to-device communication in the Internet of Things," *IEEE Syst. J.*, vol. 10, no. 3, pp. 1172–1182, Sep. 2016.
[26] P. Bosshart *et al.*, "Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.
[27] J. Liu, T. Huang, Y. Xin, J. Zhang, F. R. Yu, and Y. Liu, "VLAN-reusing: A novel solution for efficient network virtualization," *Intell. Automat. Soft Comput.*, vol. 22, no. 4, pp. 543–549, 2016.
[28] J. Tantsura *et al.*, "Segment routing with MPLS data plane," Internet Draft Internet Eng. Task Force, Sacramento, CA, USA, Tech. Rep. draft-ietf-spring-segment-routing-mpls-04, 2016.
[29] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, Apr. 2008.
[30] J. Liu, T. Huang, J.-Y. Chen, and Y.-J. Liu, "A new algorithm based on the proximity principle for the virtual network embedding problem," *J. Zhejiang Univ. Sci. C*, vol. 12, no. 11, p. 910, 2011.
[31] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an Internetwork," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2002, pp. 594–602.

TIANJIAO CHEN received the B.S. degree in electronic and information engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2016, where he is currently pursuing the Ph.D. degree in network engineering with the State Key Laboratory of Network and Switching Technology. His research interests include network virtualization and software-defined networking.



JIANG LIU received the B.S. degree in electronics engineering from the Beijing Institute of Technology, Beijing, China, in 2005, the M.S. degree in communication and information system from Zhengzhou University, Zhengzhou, China, in 2009, and the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, in 2012, where he is currently an Associate Professor. His current research interests include network architecture, network virtualization, software-defined networking, information centric networking, and tools and platforms for networking research and teaching.



QINQIN TANG received the B.S. degree in information engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2016, where she is currently pursuing the Ph.D. degree in network engineering with the State Key Laboratory of Network and Switching Technology. Her research interests include mobile edge networks, traffic offloading, and resource management and allocation.

TAO HUANG received the B.S. degree in communication engineering from Nankai University, Tianjin, China, in 2002, and the M.S. and Ph.D. degrees in communication and information system from the Beijing University of Posts and Telecommunications, Beijing, China, in 2004 and 2007, respectively, where he is currently a Professor with the Beijing University of Posts and Telecommunications. His current research interests include network architecture, routing and forwarding, and network virtualization.

RU HUO received the B.S. degree in electronics and information engineering from Harbin Engineering University, Heilongjiang, China, in 2011, and the Ph.D. degree in information and communication engineering from the Beijing University of Posts and Telecommunications, Beijing, China. From September 2015 to September 2016, she studied at The University of British Columbia, Vancouver, Canada, as a visiting Ph.D. student. She is currently a Lecturer with the Beijing University of Technology. Her current research interests include software-defined networking, information-centric networking, multi-access edge computing, blockchain, and resource scheduling.