Copyright ⓒ2016 KSII

Virtual Network Embedding through Security Risk Awareness and Optimization

Shuiqing Gong, Jing Chen, Conghui Huang, Qingchao Zhu and Siyi Zhao
College of Information and Navigation, Air Force Engineering University
Xi'an, Shanxi 710077 - China
[e-mail: gsq0121@126.com]
*Corresponding author: Shuiqing Gong

Received March 2, 2016; revised May 22, 2016; accepted June 20, 2016; published July 31, 2016

Abstract

Network virtualization promises to play a dominant role in shaping the future Internet by overcoming the Internet ossification problem. However, due to the injecting of additional virtualization layers into the network architecture, several new security risks are introduced by the network virtualization. Although traditional protection mechanisms can help in virtualized environment, they are not guaranteed to be successful and may incur high security overheads. By performing the virtual network (VN) embedding in a security-aware way, the risks exposed to both the virtual and substrate networks can be minimized, and the additional techniques adopted to enhance the security of the networks can be reduced. Unfortunately, existing embedding algorithms largely ignore the widespread security risks, making their applicability in a realistic environment rather doubtful.

In this paper, we attempt to address the security risks by integrating the security factors into the VN embedding. We first abstract the security requirements and the protection mechanisms as numerical concept of security demands and security levels, and the corresponding security constraints are introduced into the VN embedding. Based on the abstraction, we develop three security-risky modes to model various levels of risky conditions in the virtualized environment, aiming at enabling a more flexible VN embedding. Then, we present a mixed integer linear programming formulation for the VN embedding problem in different security-risky modes. Moreover, we design three heuristic embedding algorithms to solve this problem, which are all based on the same proposed node-ranking approach to quantify the embedding potential of each substrate node and adopt the *k*-shortest path algorithm to map virtual links. Simulation results demonstrate the effectiveness and efficiency of our algorithms.

Keywords: Network Virtualization, Virtual Network Embedding, Security, Node Ranking, Risk

This work was jointly supported by the National Natural Science Foundation of China (No. 61201209, 61401499), the Natural Science Foundation of Shaanxi (2015JM6340), the Industrial Science and Technology Project of Shaanxi Province (No. 2016GY-087).

http://dx.doi.org/10.3837/tiis.2016.07.002 **ISSN : 1976-7277**

1. Introduction

Network virtualization has been propounded as a fundamental technology for the future network, which abstracts the physical substrate network (SN) and allows multiple heterogeneous virtual networks (VNs) to coexist on it [1, 2]. Each VN is a collection of virtual nodes and virtual links hosted on the physical SN, and multiple VNs are isolated from each other and can be deployed customized end-to-end services for end users. With such a dynamic and programmable network environment, network virtualization can effectively enhance the flexibility of the current network architecture, promote network innovation, and address the Internet ossification problem [3]. However, as an additional virtualization layer is injected into the network architecture, network virtualization introduces new security risks, such as user attacks on VN and SN, VN attacks on SN, and SN attacks on VN [4]. These risks may violate the information confidentiality, integrity and availability, and impede the large-scale application of network virtualization.

Several techniques such as sandboxing, encryption, and other authentication and access control mechanisms (e.g., firewall, intrusion detection system) can be implemented on the elements of network virtualization to provide a secure VN. However, these mechanisms are not guaranteed to be successful, and the additional overheads incurred by such designs may negate the performance advantages gained by the network virtualization. An appropriate solution to these problems will be integrating security factors into the VN embedding process, which is one of the main challenges in network virtualization and refers to mapping VNs onto substrate nodes and links [5]. Specifically, to protect the VNs from potential attacks, in many cases they have specific security demands that have to be satisfied. That is, VNs have to be mapped onto physical network resources with qualified security levels. For example, each virtual node in a VN should be mapped onto a substrate node with qualified firewall, certain data encryption function, etc. By performing the VN embedding in a security-aware way, the risks suffered by the VNs can be minimized, and the additional mechanisms adopted to enhance the security of the VNs can be reduced, resulting in guaranteed security for the VNs and low security overheads. Unfortunately, existing VN embedding algorithms [6-17] largely ignore the widespread security risks, making their applicability in a realistic environment rather doubtful.

In this paper, we focus on the integration of security into the VN embedding, and propose novel embedding algorithms with security support to efficiently address the security risks brought by the network virtualization. We first abstract the security assurances required by the virtual nodes as numerical concept of security demands, and the protection mechanisms provided by the substrate nodes as security levels. This simple abstraction is general enough to incorporate many different kinds of security requirements and mechanisms. Then, the security demands of virtual nodes can be fulfilled by the substrate nodes with the same or higher security levels, and we introduce three security constraints into the VN embedding to tackle the security risks in the virtualized environment. Based on the abstraction, we develop three security-risky modes (i.e., secure mode, *f*-risky mode, and risky mode) to model various levels of risky conditions, which can enable a more flexible VN embedding. In particular, the secure mode always maps the VN onto the SN strictly satisfying the security demands of virtual nodes, while the risky mode performs the VN embedding without considering the security constraints, taking whatever risks it may face. The *f*-risky mode tries to satisfy the security requirements in the VN embedding to limit the security risks to be at most certain probability *f*.

Furthermore, we formulate the VN embedding problem in three security-risky modes into a mixed integer linear programing (MILP) with optimization for minimizing the embedding cost, and design three heuristic embedding algorithms correspondingly. Specifically, our proposed algorithms are all based on the same node-ranking approach, which considers the resources of the entire network and the security satisfaction to quantify the embedding potential of each substrate node, and can enable a more efficient VN embedding. Then, based on the constructed virtual node mapping tree, our algorithms adopt a load-balanced manner to map the virtual nodes with larger resource requirements onto the substrate nodes with higher embedding potentials, and leverage the *k*-shortest path algorithm to map the virtual links. In addition, we take into account the hops of substrate paths in the node mapping stage, which further improve the resource utilization of the SN. Numerical simulations indicate that our algorithms are both effective and efficient.

In summary, the main contributions of this paper can be summarized as follows: (1) We integrate the security factors into the VN embedding, and abstract them as numerical security demand and security level. Three security constraints are introduced to enable a security-aware VN embedding. (2) We first develop three security-risky modes for embedding the VNs. These operational modes model various levels of security risk involved in the VN embedding, and can be applied to bind security in most existing heuristic embedding algorithms. (3) We formulate the VN embedding problem in three security-risky modes into a MIP formulation and design three heuristic algorithms that are based on the same proposed node-ranking approach, which can efficiently improve the resource utilization. (4) Extensive simulations are conducted to demonstrate the effectiveness and efficiency of our proposed algorithms.

The rest of this paper is organized as follows. Section 2 summarizes the related work. In Section 3, the security-aware VN embedding problem is studied, and the network models are presented. Section 4 describes the MIP formulation for the VN embedding problem with security support. We propose our novel VN embedding algorithms in Section 5 and evaluate them through extensive simulations in Section 6. Section 7 concludes the paper.

2. Related Work

In network virtualization environment, as multiple VNs coexist on the same SN and share its limited resources, it is critical to efficiently allocate substrate resources to the VNs, which is known as the VN embedding problem. Due to the multiple constraints on virtual nodes and links, VN embedding has been proved to be NP-hard [23], and numerous algorithms have been proposed to tackle it.

Most previous work relies on heuristic or meta-heuristic algorithms to balance the tradeoff between the embedding performance and the computational complexity. For example, Yu et al. [8] propose a heuristic VN embedding algorithm that allows substrate path splitting and migration to obtain better resource utilization. Chowdhury et al. [9] present the ViNEYard algorithms to coordinate the node mapping and the link mapping stages. Li et al. [10] adopt the top-k dominating model to rank the nodes and propose a novel online embedding algorithm to improve the resource allocation. Fajjari et al. [11] apply the ant colony meta-heuristic to address the VN embedding problem. Cheng et al. [12] formulate the VN embedding into an integer linear programming and devise an approximation algorithm based on the discrete Particle Swarm Optimization.

Moreover, to apply the technology to systems in practice, VN embedding with specific objectives and additional constraints are studied. Sun et al. [13] focus on reconfiguring an

existing VN with dynamic resource demands. Jarray et al. [14] propose two VN protection approaches against any single physical failure in the SN. Di et al. [15] investigate the reliable VN embedding problem for obtaining efficient resource sharing. Authors in [16] and [17] consider the minimizing of energy consumption when performing the VN embedding.

As for the security-aware VN embedding, the embedding process becomes much more challenging due to the complexity of considering both the resource sharing and the additional security requirements. Unfortunately, well-known VN embedding approaches for the network virtualization largely overlook this security factor, with only a handful of exceptions.

Fischer and de Meer [18] analyze several security risks that introduced by the network virtualization, and formulate three security constraints to enable a security-aware VN embedding. Our paper takes these constraints into consideration and integrates them into the process of VN embedding.

Xing et al. [19] present two security threats exposed by the VN. Two security constraints based on the trust value and the security protection level are modeled to guarantee the security of the VN. However, they fail to consider the security of SN, and the proposed embedding approach is based on the simple greedy algorithm, leading to low resource utilization. Liu et al in [20] and [21] model the security requirements by the analysis of the security vulnerabilities suffered by nodes and links in virtualized environment. Based on the security constraints on nodes and links, an optimization of VN embedding is formulated and heuristic algorithms are developed to ensure the security of the VNs. However, as the VN requests are mapped onto the SN strictly meeting the security requirements of VN, many of them may be dropped for being over-demanding.

In this paper, in addition to the security constraints, we develop three security-risky modes for the VN embedding, i.e., secure mode, risky mode, and *f*-risky mode. By specifying these security-risky modes to model various levels of risky conditions in the virtualized environment, we can enable a more flexible security-aware VN embedding. Furthermore, the heuristic VN embedding algorithms proposed herein combine the security satisfaction and the resource of the entire network together, which can efficiently increase the resource utilization as well as satisfy the security requirements.

3. Problem Statement and Network Model

To design the mapping of VNs with security support, in this section, we first study the security-aware VN embedding problem, and then provide the network model in the following.

3.1 Security-aware VN embedding problem

3.1.1 Security risks in virtualized environment

Network virtualization promises to address several problems considering the flexibility of the current network architecture. However, as an additional virtualization layer is injected in the network architecture and multiple VNs cohost on a shared network infrastructure, network virtualization gives rise to a new array of security vulnerabilities unlike seen in the current network. Specifically, the security risks that brought by the network virtualization can be divided into three categories [4, 18, 22, 23].

Physical hosts attacks on their virtual machines (VMs). Physical hosts in the SN are responsible for offering resources to VMs in the VN under certain service level agreement (SLA), and all the services and applications operated on the VMs are ultimately carried out by the physical hosts. Thus, when a physical host is controlled by malicious attackers, it is able to change all aspects of the hosted VMs, including monitoring or snooping traffic associated to

the hosted VMs, modifying legitimate traffic or injecting malicious traffic that can disrupt the functionality of the VMs. The VMs being attacked by their physical hosts are incapable of defending themselves as they are supervised by their hosts all the time.

VMs attacks on their physical hosts. A malicious VM can access the vulnerabilities of its host through the allocated resources, and break the rigid isolation created by the virtualization process. By intruding or even taking the control of the physical host, the malicious VM is able to attack the network infrastructure to disrupt the services hosted by a competing VN, e.g. the VMs can launch a denial-of-service (DoS) attack against their physical hosts, which will take all the available resources from the hosts, and eventually bring down the entire network infrastructure.

VMs attacks on other VMs coexisted on the same physical host. In the network virtualization environment, VNs are logically isolated from each other, and VM nodes in different VNs may share the same resources of the physical hosts. A malicious VM is able to take advantage of the shared resources to access the vulnerabilities of the cohosted VMs, or launch a cross-VM side channel attack to steal information from the vulnerable VMs that coexist on the same physical host.

3.1.2 Security constraints for the VN embedding

To address the security risks that suffered by both the VMs (virtual nodes) and the physical hosts (substrate nodes) in the virtualized environment, one way is to implement additional security techniques on virtual and substrate nodes, such as intrusion detection system and anti-virus software. However, the adoption of these additional techniques may incur high security overhead. Besides, due to the dynamic, heterogeneous and open nature of resources in the virtualized environment, these passive defense techniques can't guarantee the security of user information.

By integrating security factors into the VN embedding and allocating SN resources to VNs in a security-aware way, the risks exposed to both the virtual and substrate nodes can be minimized, and the additional techniques to enhance their security can be reduced. Specifically, to protect a virtual node from the security risks, it should demand some form of security assurance from the substrate node and other cohosted virtual nodes. Simply put, such a situation can be modeled by a parameter called security level (SL), which quantifies how much protection mechanisms the substrate node and other virtual nodes can provide for this virtual node. Correspondingly, a virtual node can be assigned a security demand (SD) value, which quantifies how much security assurances it needs to defend attacks. A virtual node is expected to run successfully without any security risks when SD is not greater than SL during the VN embedding. Otherwise, it may fail and has to be remapped to another substrate node. Likewise, a substrate node also has SD for the hosted virtual nodes to protect it from VM attacks. Note that this generic security model is only used for illustration, and the parameters SL and SD are not necessarily scalar. They could be composite structure (e.g., vector) to capture a more complex model of security assurances, and their settings are large administrative issues, which are beyond the scope of this paper. For example, SL and SD could be a weighted sum of several security parameters such as anti-virus capability, firewall capability and IDS related capability.

Based on the assumptions and abstractions above, three security constraints are identified below, which are in line with three security-risky categories mentioned in the previous section, and can be incorporated in the VN embedding to enhance the security of the VNs and SN [18].

1. The security demand of a virtual node should not be higher than the security level of its mapped substrate node.

2. The security demand of a substrate node should not be higher than the security level of its

hosted virtual nodes.

3. The security level of a virtual node should not be lower than the security demands of other virtual nodes cohosted on the same substrate node.

3.1.3 Security-risky modes for the VN embedding

In this paper, we assume that a virtual node is absolutely safe if it is mapped onto a substrate node with a SL not lower than its SD, and a virtual node may fail if it takes the risk of being mapped onto a substrate node with a SL lower than its SD. Thus, we define the failure model of a virtual node as a function of the difference SD-SL between its security demand and the security level of its mapped substrate node. Specifically, the failure probability of a virtual node n_v is modeled by an exponential distribution [24], which is formulated as

$$FP(n_v)=\begin{cases}0 & if\ sd(n_v)-sl(n_s)\le 0\\ 1-e^{-\rho(sd(n_v)-sl(n_s))} & if\ sd(n_v)-sl(n_s)>0\end{cases} \tag{1}$$

where $FP(n_v)$ denotes the failure probability of the virtual node n_v, n_s denotes the substrate node that the virtual node n_v is mapped onto, $sd(n_v)$ and $sl(n_s)$ denote the security demand of n_v and the security level of n_s, respectively. ρ is a real number deciding the growth of the failure probability against the difference SD-SL.

Different VN requests have different security demands. If all current arriving VN requests are mapped onto the SN strictly satisfying the security constraints, the resources of key substrate nodes may be exhausted quickly due to being over-demanding, resulting in the refusals of subsequent VN requests and eventually the low resource utilization of the SN. Besides, in a realistic scenario, some VN requests such as military communication, financial transaction, and remote medical procedures, need more security supports. As a result, the security demands must be satisfied during the VN embedding process to protect them from any potential risks. While for other VN requests, to reduce the security overheads, they can take some or even all possible security risks, and thus the security level of the SN can be lower than the security demand of these VN requests during the VN embedding.

Based on the node failure model and the analyses above, we develop three security-risky modes to model various levels of risky conditions in the virtualized environment, which can enable a more flexible VN embedding with security support. These risky modes are designed to be applied in realistic network virtualization platforms.

Secure mode: Each virtual node in the VNs can be mapped onto a substrate node in the SN, only if the security constraints are satisfied during the VN embedding process. The secure mode is considered as a conservative way of security-aware VN embedding, which ensures that the mapped VNs are risk-free, that is, the probability of node failure is 0.

Risky mode: Each virtual node in the VNs can be mapped onto any substrate node that can satisfy its resource requirements, without considering the security constraints. The risky mode is considered as an aggressive way of VN embedding. It overlooks the security risks factor completely, that is, its tolerance of node failure probability is 1. Obviously, most previous VN embedding algorithms assumed risky mode.

***f*-risky mode:** Each virtual node in the VNs can be mapped onto a substrate node with SD > SL during the VN embedding, provided that the probability of node failure is assured to be less than f ($0 < f < 1$). Virtual nodes are taking partially security risks in *f*-risky mode.

3.2 Network model

Substrate Network A substrate network can be modeled as a weighted undirected network graph $G_S=(N_S,L_S)$, where N_S and L_S represent the set of substrate nodes and links,

respectively. For a given substrate node $n_s \in N_S$, we take the typical CPU and the security as its attributes. $cpu(n_s)$, $sd(n_s)$ and $sl(n_s)$ denote the available CPU resource, security demand and security level of n_s, respectively. For a given substrate link $l_s \in L_S$, we take the typical bandwidth as its attributes, and $bw(l_s)$ denote the available bandwidth resource of l_s. Since each virtual link is mapped onto a substrate path that consisted of one or several substrate links, we denote the set of substrate paths by P_S.

Virtual Network Request Similar to the SN, a VN request can also be modeled as a weighted undirected graph $G_V = (N_V, L_V)$, where N_V and L_V represent the set of virtual nodes and links, respectively. The requirements on virtual nodes and links are expressed in terms of the attributes of substrate nodes and links, where $cpu(n_v)$, $sd(n_v)$, and $sl(n_v)$ denote the required CPU resource, security demand and security level of n_v, respectively, and $bw(l_v)$ denote the required bandwidth resource of l_v.

Virtual Network Embedding The general security-aware VN embedding consists of a node mapping stage and a link mapping stage, which can be denoted by $M_N : (N_V, C_V^n) \to (N_S^*, R_S^n)$ and $M_L : (L_V, C_V^l) \to (P_S^*, R_S^l)$, respectively, where C_V^n and C_V^l refer to the requirements on virtual nodes and links, respectively; $N_S^* \subseteq N_S$ and $P_S^* \subseteq P_S$; R_S^n and R_S^l refer to the resources of substrate nodes and links that allocated to the VN request, respectively. **Fig. 1** illustrates an example of the security-aware VN embedding problem. We assume that substrate nodes and links have sufficient CPU and bandwidth resources, and we omit the resource constraints in the figure to emphasize the security constraints. For the simple VN request presented in the figure, (SD, SL) next to the virtual node indicates that it can offer a security level of SL, and require a security level of its mapped substrate node and co-hosted virtual nodes not lower than SD. For the SN presented in the figure, (SD, SL) next to the substrate node indicates that it can guarantee a security level of SL for virtual nodes, and require the security level of the hosted virtual node not lower than SD.

For the VN embedding in risky mode, without considering the security constraints, virtual nodes could be mapped onto any substrate nodes in the SN. While for the VN embedding in secure mode, virtual node a and b should be mapped onto a substrate node with a security level not lower than 0.8 and 0.7, respectively. Since high security level incurs high security overhead, in order to reduce the cost, we map the VN request ab to CDE, which is shown by the red dotted line in the figure. For the VN embedding in *f*-risky mode, we set *f* to 0.5, and assume that the probability of node failure is less than 0.5, only if SD – SL ≤ 0.1. Therefore, virtual node a and b should be mapped onto a substrate node with a security level not lower than 0.7 and 0.6, respectively. Obviously, the optimal mapping for VN request ab in *f*-risky mode is AB, which is shown by the blue dotted line in the figure.

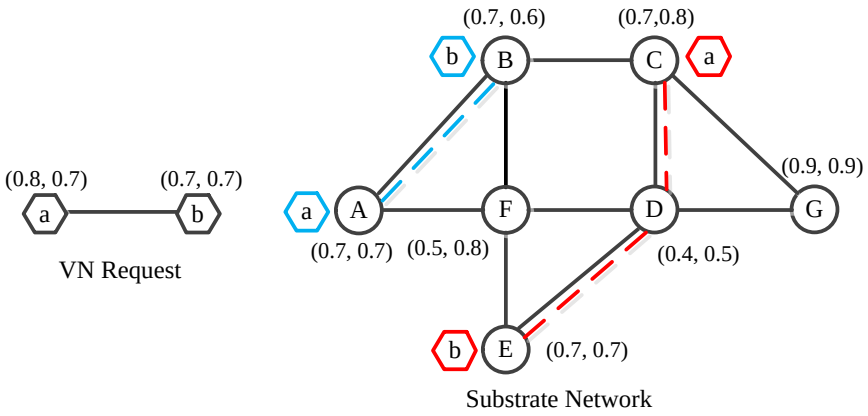


Fig. 1. An example of security-aware VN embedding

4. Problem Formulation

As discussed in Section 3.1.3, the main differences of the VN embedding in three security-risky modes are the security constraints. Thus, in this section, we model the VN embedding problem in three security-risky modes by the same MILP formulation, which employs the following variables.

x_j^i: Binary variable denoting the mapping between a virtual node and a substrate node, i.e., $x_j^i = 1$ if virtual node n_i is mapped onto substrate node n_j, and 0 otherwise.

f_{ij}^{uv}: Binary variable denoting the mapping between a virtual link and a substrate link, i.e., $f_{ij}^{uv} = 1$ if the substrate path that hosts virtual link l_{uv} goes through link l_{ij}, and 0 otherwise.

Therefore, the security-aware VN embedding problem is formulated with specific objectives and constraints, which are shown as follows.

Objective:

$$\min \quad \sum_{n_i \in N_V} \sum_{n_j \in N_S} x_j^i \cdot (1+sl(n_j)) \cdot cpu(n_i) + \sum_{l_{uv} \in L_V} \sum_{l_{ij} \in L_S} f_{ij}^{uv} \cdot bw(l_{uv}) \tag{2}$$

Subject to:

$$\forall n_i \in N_V, \forall n_j \in N_S: \; x_j^i \cdot cpu(n_i) \le cpu(n_j) \tag{3}$$

$$\forall l_{ij} \in L_S: \; \sum_{l_{uv} \in L_V} f_{ij}^{uv} \cdot bw(l_{uv}) \le bw(l_{ij}) \tag{4}$$

$$\forall n_j \in N_S, \forall l_{uv} \in L_V: \; \sum_{l_{ji} \in L_S} f_{ji}^{uv} - \sum_{l_{ij} \in L_S} f_{ij}^{uv} = \begin{cases} 1, & if \; x_j^u = 1 \\ -1, & if \; x_j^v = 1 \\ 0, & otherwise \end{cases} \tag{5}$$

$$\forall n_i \in N_V, \forall n_j \in N_S: \; x_j^i \cdot sd(n_i) - sl(n_j) \le \Delta \tag{6}$$

$$\forall n_i \in N_V, \forall n_j \in N_S: \; x_j^i \cdot sd(n_j) - sl(n_i) \le \Delta \tag{7}$$

$$\forall n_i \in N_V, \forall n_j \in N_S: \; x_j^i \cdot sd(n_i) - \min_{n_k \in \Omega(n_j)} sl(n_k) \le \Delta \tag{8}$$

$$\forall n_i \in N_V: \; \sum_{n_j \in N_S} x_j^i = 1 \tag{9}$$

$$\forall n_j \in N_S: \; \sum_{n_i \in N_V} x_j^i \le 1 \tag{10}$$

$$\forall n_i \in N_V, \forall n_j \in N_S: \; x_j^i \in \{0,1\} \tag{11}$$

$$\forall l_{uv} \in L_V, \forall l_{ij} \in L_S: \; f_{ij}^{uv} \in \{0,1\} \tag{12}$$

Similar to the previous work [9], the objective of the MILP tries to minimize the cost of embedding the VN request. Obviously, a substrate node with higher security level incurs higher security cost. Therefore, we take both the occupied resources and security levels into account, as is shown in Eq. (2). Constraint (3) and (4) are the capacity constraints,which ensuse that the required resources of a new arriving VN request mustn't exceed the available resources of the SN. Constraint (5) is the connectivity constraint, which refers to the flow conservation constraint for routing one unit of net flow from virtual node *u* to node *v*. Constraints (6) - (8) denote the three security constraints, where $\Omega(n_j)$ is the set of virtual nodes cohosted on the same substrate node n_j, *sd* and *sl* of the virtual and substrate nodes can be vectors or scalars, and Δ is a parameter that is used to adapt the MIP model to the VN embedding in different security-risky modes. In particular, when the secure mode is active, Δ is set to zero, which means that the security demand must be strictly satisfied to protect the VN requests from any security risks; when the *f*-risky mode is active, Δ is set to η, which is a real

number calculated by Eq. (1) based on the probability of node failure f; Δ is set to infinity in risky mode, which means that virtual nodes can be mapped ingoring the security constraints. Constraint (9) ensures that each virtual node in the VN request must be mapped onto just one substrate node, and Constraint (10) makes sure that no more than one virtual node in the same VN request is mapped onto the same substrate node. Constraints (11) and (12) denote the binary domain for the variables x_j^i and f_{ij}^{uv}.

5. Heuristic Algorithm Design

Solving a MILP is known to be NP-hard [25]. Even though exact algorithms (e.g., cutting plane, branch and bound) can obtain optimal results, they may incur exponential increasing running time. Consequently, they cannot scale to address large VN embedding problems. In this section, we propose novel heuristic algorithms to solve the optimization problems formulated in Section 4. More specifically, we first design a novel node-ranking approach, which takes into account the global resource information and the security satisfaction to estimate the embedding potential of each node in the SN. Then, based on our comprehensive node ranks, we devise three heuristic algorithms to tackle the VN embedding problems in different security-risky modes.

5.1 Node ranking

The general VN embedding incorporates a mapping of virtual nodes and a mapping of virtual links. Node mapping can be achieved by selecting substrate nodes with sufficient CPU resources, and link mapping aims to select substrate paths with enough bandwidth resources to host virtual links. Node-ranking approach is the basis of node mapping. It is designed for each node in the SN to evaluate its embedding potential of hosting a given node in the VN. Then the node mapping can be performed according to the node ranks based on their estimated embedding potential. Obviously, a more comprehensive and precise evaluation of embedding potential will not only simplify the VN embedding, but also promote the efficiency of subsequent link mapping, resulting in lower embedding cost for the operators of the SN.

Intuitively, substrate nodes with larger local resources (i.e., CPU resources and bandwidth resources of adjacent links) are more likely to host virtual nodes, and should be assigned higher node ranks. The resources of neighboring nodes will also affect the rank of a substrate node. For example, node A and B have the same local resources, but the neighbors of node A have more resources than those of node B, and so mapping a virtual node onto A has a higher chance to achieve a successful link mapping. In addition, a higher security level of a substrate node also contributes to its node rank since it can satisfy the higher security demand of a virtual node. However, as high security level incurs high embedding cost according to Eq. (2), it would be better to exactly match the security demands of virtual nodes during the node mapping stage. To simplify the embedding problem, we consider the security demand and the security level as scalars in our heuristic algorithm.

Taking the aforementioned node-ranking factors into consideration, we design a novel ranking approach to evaluate the embedding potentials of substrate nodes for hosting the virtual nodes. Specifically, given a security demand k, the uniformed CPU resource of a substrate node $n_s \in N_S$ is defined as follows:

$$cpu'(n_s,k)=\begin{cases}0 & sl(n_s)<k\\ cpu(n_s)\cdot e^{-\beta(sl(n_s)-k)} & sl(n_s)\geq k\end{cases} \tag{13}$$

where $cpu'(n_s,k)$ denotes the uniformed CPU resource of substrate node n_s at certain security

demand k, $\beta \in (0,1)$ is damping factor. Eq. (13) indicates that a substrate node with a security level exactly matching the security demand has larger uniformed CPU resource.

Based on the formulation of the uniformed CPU resource, the embedding potential $ep(n_s)$ of node $n_s \in N_S$ at security demand k is calculated as follows:

$$ep(n_s,k) = \lambda \cdot \overline{cpu'}(n_s,k) + (1-\lambda) \sum_{n_j \in Nb(n_s)} \frac{bw(l_{js})}{\sum_{n_k \in Nb(n_j)} bw(l_{jk})} \cdot ep(n_j,k) \tag{14}$$

where $\lambda \in (0,1)$ is a bias factor, $Nb(n_s)$ denotes the set of neighboring nodes that directly connect node n_s by a link. $cpu'(n_s,k)$ denotes the normalized uniformed CPU resource of node n_s, which is formulated as follows:

$$\overline{cpu'}(n_s,k) = \frac{cpu'(n_s,k)}{\sum_{n_i \in N_S} cpu'(n_i,k)} \tag{15}$$

Obviously, the embedding potential of a substrate node increases with the increasing of its available CPU resource, the bandwidth of its adjacent links and the embedding potentials of its neighboring nodes.

Using a vector format, we can calculate the embedding potentials for all substrate nodes as

$$\mathbf{ep} = \lambda \cdot \mathbf{c} + (1-\lambda) \cdot \mathbf{M} \cdot \mathbf{ep} \tag{16}$$

where $\mathbf{ep} = (ep(n_1,k), ep(n_2,k), \ldots, ep(n_{|N_S|},k))^T$, $\mathbf{c} = (\overline{cpu'}(n_1,k), \overline{cpu'}(n_2,k), \ldots, \overline{cpu'}(n_{|N_S|},k))^T$. $\mathbf{M}$ is a row-stochastic matrix with the dimension of $|N_S| \times |N_S|$, and each element in $\mathbf{M}$ is defined as follows:

$$m_{ij} = \begin{cases} 0 & l_{ij} \notin L_S \\ \dfrac{bw(l_{ij})}{\sum_{n_k \in Nb(n_j)} bw(l_{jk})} & l_{ij} \in L_S \end{cases} \tag{17}$$

Since the calculation of node-ranking vector with Eq. (17) could be time consuming as the size of SN gets large, we develop a simple iterative computation process as follows:

$$\mathbf{ep}^{(t+1)} = \lambda \cdot \mathbf{c} + (1-\lambda) \cdot \mathbf{M} \cdot \mathbf{ep}^{(t)} \tag{18}$$

where $\mathbf{ep}^{(t)}$ is the embedding potential vector after t iteration. The embedding potential of each node considers the global resource information in a recursive manner, and we set the initial $\mathbf{ep}^{(0)}$ as

$$\mathbf{ep}^{(0)} = (\overline{cpu'}(n_1,k), \overline{cpu'}(n_2,k), \ldots, \overline{cpu'}(n_{|N_S|},k))^T \tag{19}$$

Algorithm 1 Node-ranking Algorithm

input: Substrate network $G_S = (N_S, L_S)$, security demand k, $\mathbf{ep}^{(0)}$, maximum iterations MAX_IT, small positive real number μ

Output: Embedding potential vector $\mathbf{ep}$

1. $\mathbf{ep} = \mathbf{ep}^{(0)}$, $t = 0$, $\Delta = \infty$;
2. **while** $t <$ MAX_IT and $\Delta \geq \mu$ **do**
3. $\mathbf{ep}^{(t+1)} = \lambda \cdot \mathbf{c} + (1-\lambda) \cdot \mathbf{M} \cdot \mathbf{ep}^{(t)}$;
4. $\Delta = \left\| \mathbf{ep}^{(t+1)} - \mathbf{ep}^{(t)} \right\|$;
5. $t = t+1$;
6. **end while**
7. $\mathbf{ep} = \mathbf{ep}^{(t)}$;

The iteration will be terminated when the maximum number of iteration is reached or the difference of two adjacent iterations is less than a small enough positive threshold, e.g., when $\left\|\mathbf{ep}^{(t+1)}-\mathbf{ep}^{(t)}\right\|<0.0001$. Note that the maximum eigenvalue of matrix $\mathbf{M}$ is not larger than 1 as $\|\mathbf{M}\|\le 1$ according to Eq. (17). This guarantees that our iterative computation can always converge to a stationary solution. The details of our node-ranking algorithm are given by Algorithm 1.

5.2 S-VNE: VN embedding algorithm in secure mode

To perform an efficient VN embedding in secure mode, we propose a novel heuristic embedding algorithm called S-VNE, which consists of the node mapping stage based on the node ranking and the link mapping stage based on the *k*-shortest path algorithm.

5.2.1 Node mapping

Most existing algorithms merely adopt the greedy strategy to map the virtual nodes requiring more resources to the substrate nodes with larger available resources, while the hops of substrate paths that are used to host the virtual links are ignored in the node mapping stage. In order to avoid this problem, S-VNE takes into account not only the embedding potential values, but also the hops of the substrate paths in the node mapping stage.

Specifically, we first sort the virtual nodes according to their required resource capacity (RRC) in descending order. The RRC of a virtual node n_v is defined as follows:

$$RRC(n_v)=cpu(n_v)\cdot e^{sd(n_v)}\cdot\sum_{l_v\in L(n_v)}bw(l_v) \tag{20}$$

where $L(n_v)$ denotes the set of adjacent virtual links that connect directly to virtual node n_v. The larger the RRC value of node n_v is, the more resources including the CPU, security demand and the bandwidth of adjacent links it requires, so that it should be mapped in priority due to the limited available resources in the SN.

Then, we construct the virtual node mapping tree (VNMT) for the VN, which can effectively decrease the hops of substrate paths that virtual links are mapped onto. VNMT is constructed based on the RRC values of virtual nodes and the topology of the VN. In particular, the construction of VNMT works as follows. For a given VN, we first select the virtual node with the largest RRC value as the root node. Then those virtual nodes that directly connect to the root by virtual links become its children from left to right according to the descending order of their RRC values. Other virtual nodes in the VN are constructed recursively in a similar way.

After constructing the VNMT, S-VNE adopts the breadth first search (BFS) strategy to map virtual nodes. For the root node of the VNMT, S-VNE maps it to the substrate node with the largest embedding potential value while satisfying the resource constraint (Eq. (3)) and the security constraints (Eq. (6) - (8)). For the other virtual nodes in the VNMT, S-VNE maps them to the substrate nodes with the largest NR, which is a metric for selecting the substrate nodes in the node mapping stage, and is defined according to the embedding potential values of the substrate nodes and the hops of the substrate paths that are assigned to the virtual links. In particular, when we map virtual node n_v, the NR of the substrate node n_s is defined as

$$NR(n_s)=\frac{ep(n_s,sd(n_v))}{hops(M_N(f(n_v)),n_s)} \tag{21}$$

where $ep(n_s,sd(n_v))$ denotes the embedding potential value of n_s according to the security demand of n_v, $f(n_v)$ denotes the father node of n_v in the VNMT, $M_N(f(n_v))$ denotes the substrate node that $f(n_v)$ is mapped onto, and $hops(M_N(f(n_v)),n_s)$ denotes the hops of the shortest path from $M_N(f(n_v))$ to n_s in the SN.

Algorithm 2 Node Mapping Algorithm

input: Substrate network $G_S = (N_S, L_S)$, the arriving VN request $G_V = (N_V, L_V)$

Output: Node mapping M_N

1. **for** each possible security demand *k* **do**
2. Sort all the substrate nodes according to their embedding potential values calculated by Algorithm 1;
3. **end for**
4. Calculate the RRC for each virtual node $n_v \in N_V$;
5. Construct the VNMT for G_V according to the RRC values of virtual nodes in descending order;
6. Map the root node of VNMT to the substrate node with the largest embedding potential value;
7. **for** other unmapped nodes in the VNMT **do**
8. Choose the virtual node n_v using BFS strategy, and compute the NR values of substrate nodes;
9. Map n_v to the substrate node n_s with the largest NR value while satisfying the resource constraint (Eq. (3)) and the security constraints (Eq. (6) - (8)), namely $M_N(n_v) = n_s$;
10. **if** node constraints are not satisfied **then**
11. **return** NODE_MAPPING_FAILED;
12. **end if**
13. **end for**
14. **return** NODE_MAPPING_SUCCESS;

The reasons why we take the NR as the node selection metric are as follows. Firstly, the substrate node with larger embedding potential value indicates that the node's available resources are greater, and selecting it for mapping virtual node helps balance the stress on substrate nodes. In addition, as the virtual node $f(n_v)$ has been mapped onto the SN, if $hops(M_N(f(n_v)), n_s)$ is too large, the cost of mapping the virtual link between n_v and $f(n_v)$ is also too large, resulting in low resource utilization of the SN. Therefore, based on the VNMT and NR, our node mapping algorithm can keep the mapped substrate nodes connected closely to each other, and is favourable for achieving larger successful probability in the following link mapping stage. The details of the node mapping algorithm is shown in Algorithm 2.

Algorithm 3 Link Mapping Algorithm

Input : Substrate network $G_S = (N_S, L_S)$, the VN request $G_V = (N_V, L_V)$, node mapping M_N

Output: Link mapping M_L

1. Rank virtual links in current VN request according to the required bandwidth in descending order;
2. **for** each unmapped virtual link $l_{uv} \in L_v$ in order **do**
3. Search the *k*-shortest paths between the selected nodes in the SN by increasing k;
4. **if** find a path $p \in P_S$ that can satisfy the bandwidth constraint of l_{uv} **then**
5. Map l_{uv} to this path, namely $M_L(l_{uv}) = p$;
6. **else**
7. **return** LINK_MAPPING_FAILED;
8. **end if**
9. **end for**
10. **return** LINK_MAPPING_SUCCESS;

5.2.2 Link mapping

In the link mapping stage, similar to the previous work [6, 21], S-VNE adopts the *k*-shortest path [26] algorithm to map the virtual links to the substrate paths, as shown in Algorithm 3. Since different substrate paths that virtual links are mapped onto may share the same substrate links and compete for their limited bandwidth resources, it is difficult or even impossible to

map virtual links with large bandwidth requirements due to the lack of bandwidth resources in the SN. Therefore, virtual links with large required bandwidth should be mapped in priority.

The time complexity of node-ranking algorithm is $O(|N_S|^2\log(1/\mu))$ [27], the complexity of constructing the VNMT is $O(|N_S||N_V|)$, and the complexity of computing NR values is $O(|N_S|^2)$. Thus, Algorithm 2 can be computed in polynomial time in terms of $|N_S|$, $|N_V|$ and $\log(1/\mu)$. Algorithm 3 adopts the *k*-shortest path algorithm to map the virtual links, which can also be solved in polynomial time [26]. Therefore, S-VNE is a polynomial-time algorithm.

5.3 FR-VNE: VN embedding algorithm in *f*-risky mode

For the VN embedding in *f*-risky mode, the node mapping is performed by taking the maximum probability of node failure *f*, that is, each virtual node $n_v \in N_V$ can be mapped onto a substrate node $n_s \in N_S$ with $sd(n_v) - sl(n_s) \le \eta$, where η is a real number calculated by our node failure model (Eq. (1)) according to the probability of node failure *f*. Based on the aforementioned analyses, we can see that the actual minimum security demand of a virtual node $n_v \in N_V$ is $sd(n_v) - \eta$ for the VN embedding in *f*-risky mode. Therefore, in order to reduce the embedding cost, it would be better to exactly match the actual security demands of virtual nodes during the node mapping stage. The steps of FR-VNE algorithm are similar to those of S-VNE in Section 5.2. The main difference is the definition of the uniformed CPU resource of the substrate node (Eq. (13)) in the node-ranking approach in Section 5.1. In particular, given a security demand *k* in *f*-risky mode, the uniformed CPU resource of a substrate node $n_s \in N_S$ is redefined as follows:

$$cpu'(n_s, k) = \begin{cases} 0 & sl(n_s) < k - \eta \\ cpu(n_s) \cdot e^{-\beta(sl(n_s)-k+\eta)} & sl(n_s) \ge k - \eta \end{cases} \tag{22}$$

where η is a real number calculated by Eq. (1), and is used to guarantee the maximum probability of node failure *f* at the security demand *k*.

5.4 R-VNE: VN embedding algorithm in risky mode

For the VN embedding in risky mode, as the maximum tolerance of node failure probability is 1, virtual nodes could be mapped onto any substrate nodes satisfying their resource requirements without considering the security constraints. Thus, to reduce the embedding cost, substrate nodes with lower security levels have higher priority to be selected in the node mapping stage. R-VNE is a heuristic algorithm designed for the VN embedding in risky mode, and its main steps are also similar to those of S-VNE. However, as the security constraints are neglected in the node mapping stage, the node-ranking approach of R-VNE is slightly different from that of S-VNE. Specifically, the uniformed CPU resource of a substrate node $n_s \in N_S$ is redefined without considering different security demands, i.e.

$$cpu'(n_s) = cpu(n_s) \cdot e^{-\beta sl(n_s)} \tag{23}$$

where we can see that the lower the security level of a substrate node is, the larger its uniformed CPU resource is. Thus, it would have higher rank after the calculating of our node-ranking method. In addition, as the virtual nodes with larger resource requirements are more difficult to be mapped due to the finite substrate resource, the RRC of a virtual node n_v for the construction of VNMT is redefined as follows:

$$RRC(n_v) = cpu(n_v) \cdot \sum_{l_v \in L(n_v)} bw(l_v) \tag{24}$$

which guarantees that the virtual nodes with greater resource demand have higher mapping priorities in the node mapping stage.

6. Performance Evaluation

In this section, we first introduce the network settings in our simulations and the performance metrics for the evaluation of our proposed algorithms, and then present the main evaluation results. Our evaluation focuses primarily on quantifying the effectiveness and efficiency of our embedding algorithms.

6.1 Simulation environment

6.1.1 Network settings

In this paper, we adopt the similar simulation model as those of previous work [12, 16, 21]. We randomly generate the topologies of VN requests and SN by using GT-ITM tool [28]. The SN is configured with 100 nodes and about 500 links, which corresponds to the size of a medium-sized ISP. The initial available CPU resource of substrate nodes and the bandwidth resource of substrate links are real numbers following a uniform distribution between 50 and 100. For each VN request, the number of nodes is uniformly distributed between 2 and 10, and the average link connectivity rate is set to 0.5. The required CPU and bandwidth resource are real numbers uniformly distributed between 0 and 50. The arrivals of VN requests follow a Poisson process with an average arrival rate of 5 per 100 time units, and the lifetime of each VN request are modeled by an exponential distribution with an average of 1000 time units. In addition, the security level and the security demand of both the substrate and virtual nodes are real numbers uniformly distributed between 0 and 1, and the value of f for the f-risky mode is set to 0.5. We run our simulations for 50000 time units, so that the performance can reach a stable state. Each simulation is performed ten instances and we take the average values as the final results.

6.1.2 Performance metrics and comparisons

Below are the metrics that we use to evaluate our VN embedding techniques:

Long-term average revenue The revenue of accepting a VN request at time t is defined as the total resource it requires [12, 20]. Since high security demand incurs high revenue, the long-term average revenue is thus defined as

$$R = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \Big(\sum_{n_i \in N_V} (1+sd(n_i)) \cdot cpu(n_i) + \sum_{l_{uv} \in L_V} bw(l_{uv}) \Big) \tag{25}$$

Resource utilization Resource utilization is defined as the long-term revenue to cost ratio [6, 12], which is formulated as

$$RU = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} \Big(\sum_{n_i \in N_V} (1+sd(n_i)) \cdot cpu(n_i) + \sum_{l_{uv} \in L_V} bw(l_{uv}) \Big)}{\sum_{t=0}^{T} \Big(\sum_{n_i \in N_V} \sum_{n_j \in N_S} x_j^i \cdot (1+sl(n_j)) \cdot cpu(n_i) + \sum_{l_{uv} \in L_V} \sum_{l_{ij} \in L_S} f_{ij}^{uv} \cdot bw(l_{uv}) \Big)} \tag{26}$$

Acceptance ratio Acceptance ratio is used to evaluate the ratio of the VN requests that are successfully mapped. Let VNR_{suc} be the number of VN requests that are successfully mapped, and VNR_{sum} be the total arrival VN requests, the acceptance ratio is formulated as

$$AR = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} VNR_{suc}}{\sum_{t=0}^{T} VNR_{sum}} \tag{27}$$

Node failure rate A virtual node suffers security risks when its security demand is lower than the security level of its mapped substrate node, and may fail during the operation of VN. Let N_{fail} be the number of failed virtual nodes, and N be the number of total virtual nodes that have been mapped onto the substrate nodes, the node failure rate is thus defined as

$$NFR = \frac{N_{fail}}{N} \tag{28}$$

Our simulation experiments evaluate six algorithms. S-VNE, FR-VNE and R-VNE are the VN embedding algorithms proposed in this paper. G-SP is the baseline algorithm presented in [8] without considering the path-spliting, which adopts the greedy strategy to map the virtual nodes and the *k*-shortest path algorithm to map virtual links. Obviously, G-SP assume the risky mode. On the basis of considering the security constraints, we propose other two algorithms improved from G-SP, i.e., S-G-SP and FR-G-SP. S-G-SP is a simple extension to G-SP by adding the security constraints for the VN embedding in secure mode, and FR-G-SP is revised by taking into account the security constraints in *f*-risky mode. Besides, the calculation of embedding revenue and cost for all three comparing algorithms are updated by using Eq. (2) and (25).

6.2 Evaluation Results

In our evaluation, we first simulate the online VN embedding to evaluate the efficiency of six algorithms in terms of the aforementioned metrics, and then we vary the security demand of VN requests and the value of f for the f-risky mode to evaluate their impacts on the performance of the embedding algorithms.

6.2.1 Evaluation with general VN requests

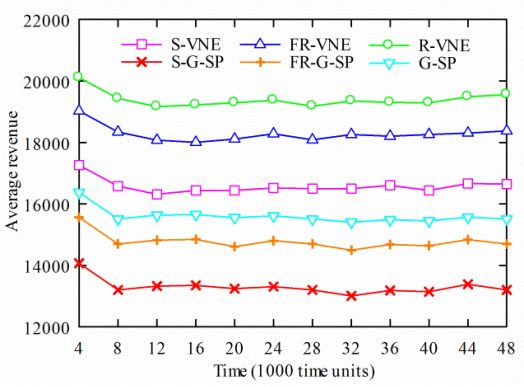
The attributes of general VN requests and SN are the same as those in the network settings, and would not change. The comparison results are shown in **Fig. 2**, and we summarize the following key observations.

Figs. 2(a) and **(b)** illustrates the average revenue and the acceptance ratio of six algorithms in stable state, respectively. From the figures, we can see that R-VNE produces the largest average revenue and the highest acceptance ratio. For the algorithms in the same security-risky mode, the average revenue and acceptance ratio of our proposed algorithms (i.e. S-VNE, FR-VNE and R-VNE) are much larger than S-G-SP, FR-G-SP and G-SP, respectively. Among three security-risky modes, the risky mode performs better than the other two modes, and the secure mode is the worst. The reasons are as follows: The revenue of embedding the same VN request is fixed according to its definition (Eq. (25)). Since our proposed algorithms consider minimizing the hops of substrate paths that virtual links are mapped onto by taking NR as the node selection metric in the node mapping stage, they can effectively reduce the bandwidth resource allocated to the VN requests, which makes the subsequent link mapping easier and leads to accepting more VN requests. Therefore, the revenue and acceptance ratio of our proposed algorithms are larger than the others. In addition, due to considering the additional security constraints, the mapping of VN requests in secure and *f*-risky modes are more difficult than that in risky mode. Thus, without regarding the security constraints, algorithms in risky mode can obtain larger average revenue and higher acceptance ratio.

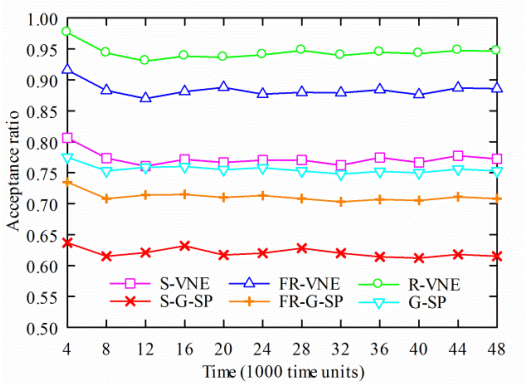
Fig. 2(c) illustrates the resource utilization of six algorithms in stable state. It can be seen that the relative results are similar to the average revenue and the acceptance ratio. Specifically, the resource utilization of R-VNE is higher than those of other embedding algorithms, and our proposed algorithms in three security-risky modes perform better than the G-SP based algorithms, respectively. The reasons are as follows: Since the revenue of mapping a VN request is fixed, the resource utilization is mainly determined by the embedding cost according to Eq. (26). Thus, our proposed algorithms can achieve higher resource utilization because they take into account reducing the hops of substrate paths in the node mapping stage. Besides, without regarding the security constraints, algorithms in risky mode can reduce the

consumption of bandwidth resource more efficiently than these in secure and *f*-risky modes, leading to higher resource utilization.

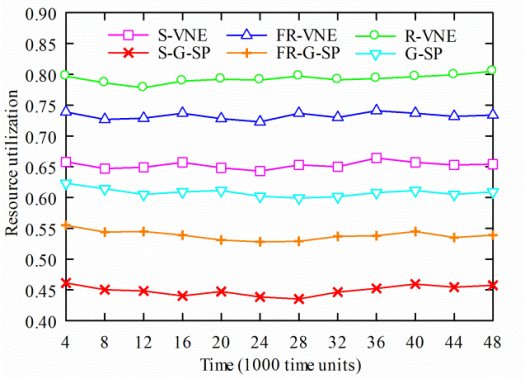
Fig. 2(d) illustrates the node failure rate of six algorithms in stable state. As the VN requests are mapped onto the SN strictly satisfying the security demands, the mapped VNs are risk-free, so that the node failure rate is 0. For the other four algorithms, it can be seen that the node failure rates of algorithms in *f*-risky mode are much lower than those of the risky mode ones. The reasons are as follows: According to our proposed node failure model, when the security demand of a virtual node is lower than the security level of its mapped substrate node, the virtual node suffers security risks and may fail during the operation of the VN. The lower the security level of the substrate node is, the higher the failure probability of virtual node is. Thus, algorithms in risky mode prefer to map the virtual nodes to the substrate nodes with lower security level to reduce the embedding cost, which results in higher node failure rate. In addition, from the figure we can see that the node failure rate of FR-VNE and R-VNE is slightly higher than that of FR-G-SP and G-SP, respectively. This is because that FR-VNE and R-VNE consider the security level of substrate nodes in the node-ranking approach, and prefer to select the substrate node with lower security level in the node mapping stage, which lead to higher node failure rate than other algorithms.



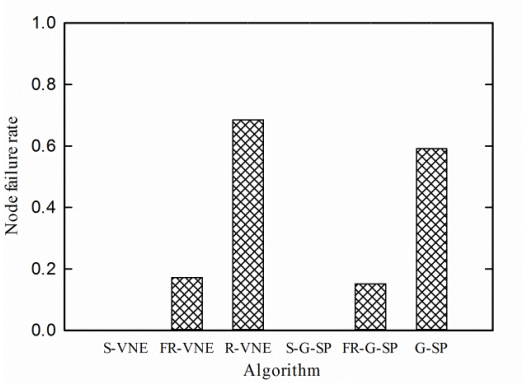
(a) Long-term average revenue



(b) Acceptance ratio



(c) Resource utilization



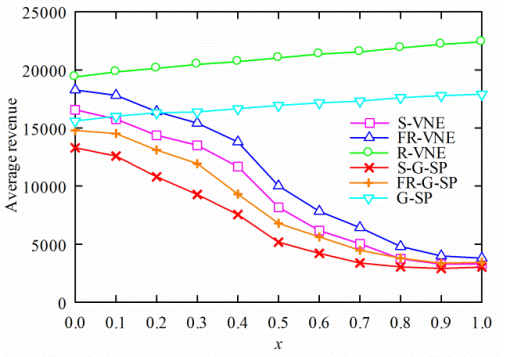
(d) Node failure rate

Fig. 2. Comparison between our algorithms and others on general VN requests in stable state

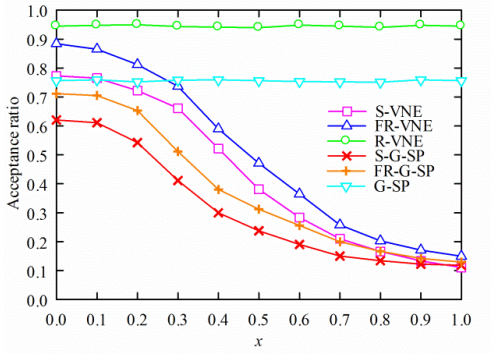
6.2.2 Evaluation with varied security demand of VN requests

To evaluate the impact of varied security demand on the performance of algorithms, we set the

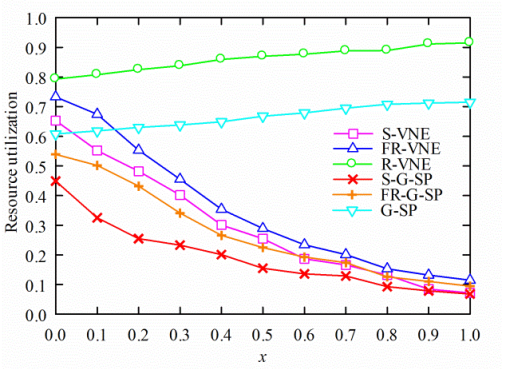
security demand and level of virtual nodes to follow a uniform distribution between x and 1, where x is the lower bound of security demand and level, and is set to range from 0 to 1 in our simulations. Other parameters are the same as those in Section 6.1. The comparison results are shown in **Fig. 3.** From these results, we conclude the following observations.



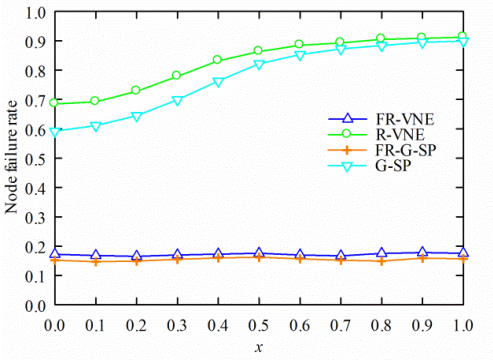
(a) Long-term average revenue



(b) Acceptance ratio



(c) Resource utilization



(d) Node failure rate

Fig. 3. Comparison between our algorithms and others on VN requests with varied security demand

First, as the security demand and level of virtual nodes increase, from **Fig. 3(a)** and **(b)**, we can see the average revenue and the acceptance ratio of algorithms in secure and f-risky modes decrease, while S-VNE and FR-VNE always achieve relatively larger revenue and higher acceptance ratio than S-G-SP and FR-G-SP, respectively, which demonstrate that our proposed algorithms are more efficient than the others in terms of the average revenue and acceptance ratio. Besides, with the increasing of security demand and level, the average revenues of R-VNE and G-SP increase, while the acceptance ratios remain almost unchanged. This is because R-VNE and G-SP perform the VN embedding without considering the security constraints. As a result, the variations of the security demand and level of virtual nodes have almost no effect on the acceptance ratio of VN requests. However, according to the definition of revenue, higher security demands of virtual nodes lead to larger revenues, so that the revenues of R-VNE and G-SP increase due to the unchangeable acceptance ratio and the increasing of security demand of VN requests.

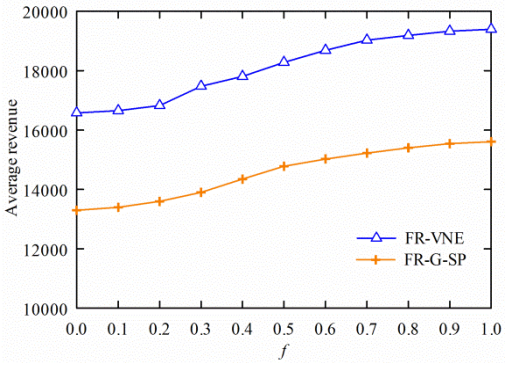
Second, with the increasing of the security demand and level, from **Fig. 3(c)**, we can see that the resource utilization decreases for all algorithms in secure and f-risky modes, and increases in risky mode. But S-VNE, FR-VNE and R-VNE still achieve relatively higher resource utilization than that of S-G-SP, FR-G-SP and G-SP, respectively. It can be understood as

follows: For the algorithms in secure and *f*-risky mode, as the security demand increases, virtual nodes are more difficult to map due to the unchangeable security levels of the SN. As a result, they might be mapped onto the substrate nodes that are far away from each other, which will lead to higher cost of link mapping and lower resource utilization. For the algorithms in risky mode, without regarding the security constraints, the resource utilization increases due to the increasing of revenue and the almost unchangeable embedding cost.

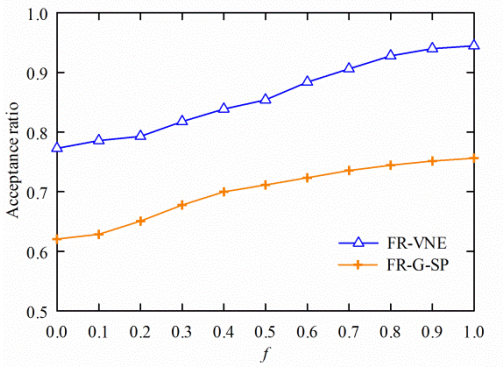
Third, as the security demand and level of virtual nodes increase, from **Fig. 3(d)**, it can be seen that the node failure rate of algorithms in risky mode increases, while the node failure rate of algorithms in *f*-risky mode remain almost unchanged. Besides, G-SP always obtains a lower node failure rate than R-VNE. Note that the node failure rate of S-VNE and S-G-SP is 0 as they are risk-free, and we don't illustrate them in **Fig. 3(d)**. It can be understood as follows: For R-VNE and G-SP, since the security level of substrate nodes remain unchanged, the difference of the security demand of a virtual node and the security level of its mapped substrate node become larger with the increasing of security demand of virtual nodes. Thus, the node failure rate will increase according our proposed node failure model. For FR-VNE and FR-G-SP, they can guarantee a maximum node failure probability at *f* when performing the VN embedding, which leads to their almost unchangeable node failure rate.

6.2.3 Evaluation with varied *f* for the *f*-risky mode

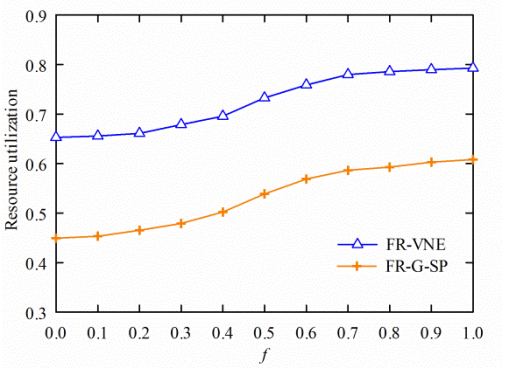
To evaluate the impact of varied *f* on the performance of algorithms in the *f*-risky mode, we set the value of *f* as increasing from 0 to 1. Other parameters are the same as those in Section 6.1. The comparison results are shown in **Fig. 4**.



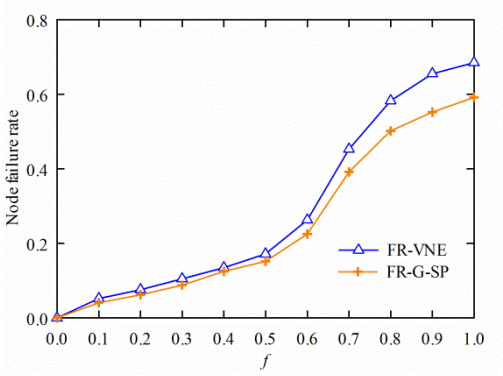
(a) Long-term average revenue



(b) Acceptance ratio



(c) Resource utilization



(d) Node failure rate

Fig. 4. Comparison between the algorithms in *f*-risky mode with varied *f*

Figs. 4(a), (b) and **(c)** show the long-term average revenue, the acceptance ratio and the resource utilization of the comparing algorithms in *f*-risky mode, respectively. From the figures, we can see that the values of aforementioned performance metrics increase as *f* increases. Moreover, our proposed FR-VNE always outperforms FR-G-SP. The reasons are as follows: As *f* increases, the VN requests can bear more security risks. As a result, virtual nodes can be mapped onto substrate nodes with lower security levels according to the security constraints (Eq. (6) – (8)), which increases the success rate of the node mapping and eventually results in higher average revenue, acceptance raito and resource utilization. Besides, since our proposed FR-VNE coordinates the node mapping stage and the link mapping stage by taking into account the hops of substrate paths in the node mapping, it can achieve a better performance than FR-G-SP.

Fig. 4(d) show the node failure rate of the comparing algorithms in *f*-risky mode. From the figure we can see that the node failure rates of both FR-VNE and FR-G-SP increase as *f* increases. Moreover, FR-G-SP always produces a slightly lower failure rate than FR-VNE. The reasons are as follows: As *f* increases, in order to reduce the embedding cost, both FR-VNE and FR-G-SP try to map the virutal nodes to the substrate nodes with lower security levels while satisfying the securtiy constraints. Thus, the differences between the security demands of virutal nodes and the security levels of substrate nodes will become larger and larger, resulting in higher node failure rate according to the node failure model (Eq. (1)). Besides, FR-VNE prefer to map virtual nodes to substrate nodes with lower security levels by considering the security level in the node-ranking approach, which results in a slightly higher node failure rate than FR-G-SP.

7. Conclusion

In this paper, we study the VN embedding problem from the perspective of considering the security. We abstract the security requirements and the protection mechanisms as numerical concept of security demands and security levels, and the corresponding security constraints are introduced into the VN embedding. We develop three security-risky modes to enable a more flexible VN embedding with security support, and the VN embedding problem in these modes are formulated into a MILP with optimization of minimizing the embedding cost. Based on our novel node-ranking approach, we propose three heuristic algorithms for the VN embedding in different security-risky modes. We compare the performance of the proposed algorithm with other algorithms. Our observations are summarized as follows:

- The proposed R-VNE algorithm achieves higher average revenue, acceptance ratio and resource utilization than any other improved algorithms, but it produces the highest node failure rate.
- The proposed algorithm (i.e. S-VNE, FR-VNE and R-VNE) in each security-risky mode performs much better than the improved baseline algorithm (i.e. S-G-SP, FR-G-SP and G-SP) in terms of the average revenue, acceptance ratio and resource utilization, respectively. However, the better performance of FR-VNE and R-VNE is achieved at the cost of a slightly higher node failure rate than that of FR-G-SP and G-SP.
- Among the three security-risky modes, risky mode performs better than f-risky mode, and secure mode is the worst in terms of the average revenue, acceptance ratio and resource utilization. However, the better performance of the risky mode is achieved at the cost of higher node failure rate.

In a realistic network virtualization scenario, as the SN would host different types of VNs with different levels of security demand, the operator of SN will adopt different security-risky

modes to embedding the VN requests according to their specific contexts. In particular, the proposed algorithm in secure mode is adopted to embed the VN requests with high security demand (e.g., military communication, financial transaction), the risky mode is used for the VN requests that can bear any potential security risks, and otherwise the *f*-risky mode is adopted. The value of *f* for the *f*-risky mode can be changed dynamically to adapt to the specific types of the VN requests, and such study is left for further development. Moreover, as the nodes may fail during the operation of VNs in *f*-risky and risky modes, in the future work, we will extend our work by considering VN reconfiguration (reassignment or migration) to recover the failure nodes.

References

[1] N. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862-876, 2010. Article (CrossRef Link).

[2] A.J. Wang, M. Iyer, R. Dutta, G.N. Rouskas and I. Baldine, "Network virtualization: technologies, perspectives, and frontiers," *Journal of Lightware Technology*, vol. 31, no. 4, pp. 523-537, 2013. Article (CrossRef Link).

[3] T. Anderson, L. Peterson, S. Shenker and J. Turner, "Overcoming the Internet impasse through virtualization," *IEEE Computer Magazine*, vol. 38, no. 4, pp. 34-41, 2005. Article (CrossRef Link).

[4] S. Natarajan and T. Wolf, "Security issues in network virtualization for the future Internet," in *Proc. of IEEE ICNC*, pp. 537-543, 2012. Article (CrossRef Link).

[5] A. Fischer, J. F. Botero, M. Till Beck, H. De Meer and X. Hesselbach, "Virtual network embedding: a survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888-1906, 2013. Article (CrossRef Link).

[6] J. Liao, M. Feng, T. Li, J, Wang and S. Qing, "Topology-aware virtual network embedding using multiple characteristics," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 1, pp. 145-164, 2014. Article (CrossRef Link).

[7] D. Liao, G. Sun, V. Anand and H, Yu, "Efficient provisioning for multicast virtual network under single regional failure in cloud-based datacenters," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 7, pp. 2325-2349, 2014. Article (CrossRef Link).

[8] 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, no. 2, pp. 17-29, 2008. Article (CrossRef Link).

[9] M. Chowdhury, M.R. Rahman and R. Boutaba, "ViNEYard: virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206-219, 2012. Article (CrossRef Link).

[10] X. Li, H. Wang, B. Ding, X.Y. Li and D. Feng, "Resource allocation with multi-factor node ranking in data center networks," *Future Generation Computer Systems*, vol. 32, no. 2, pp. 1-12, 2014. Article (CrossRef Link).

[11] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "VNE-AC: Virtual network embedding algorithm based on ant colony meta-heuristic," in *Proc. of IEEE ICC*, pp. 1 –6, 2011. Article (CrossRef Link).

[12] X. Cheng, S. Su, Z. Zhang et al., "Virtual network embedding through topology awareness and optimization," *Computer Networks*, vol. 56, no. 6, pp. 1797-1813, 2012. Article (CrossRef Link).

[13] G. Sun, H. Yu, A. Vishal and L. Li. "A cost efficient framework and algorithm for embedding dynamic virtual network requests," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1265-1277, 2013. Article (CrossRef Link).

[14] A. Jarray and A. Karmouch. "Cost-Efficient Mapping for Fault-Tolerant Virtual Networks," *IEEE Transactions on Computers*, vol. 64, no. 3, pp. 668-681, 2015. Article (CrossRef Link).

[15] H. Di, A. Vishal and H. Yu. "Design of reliable virtual infrastructure with resource sharing," *Computer Networks*, vol. 62, no. 5, pp. 137-151, 2014. Article (CrossRef Link).

[16] S. Su, Z. Zhang, A. Liu *et al*., "Energy-aware virtual network embedding," *IEEE/ACM Translations on Networking*, vol. 22, no. 5, pp. 1607-1620, 2014. Article (CrossRef Link).
[17] Z. Zhang, S. Su, J. Zhang, *et al*. "Energy aware virtual network embedding with dynamic demands: Online and offline," *Computer Networks*, vol. 93, pp. 448-459, 2015. Article (CrossRef Link).
[18] A. Fischer and H. de Meer, "Position paper: Secure virtual network embedding," *Praxis der Informationsverarbeitung und Kommunikation*, vol. 34, no. 4, pp. 190–193, 2011. Article (CrossRef Link).
[19] C. Xing, J. Lan and Y. Hu, "Virtual Network with Security Guarantee Embedding Algorithms," *Journal of Computers*, vol. 8, no. 11, pp. 2782-2788, 2013. Article (CrossRef Link).
[20] S. Liu, Z. Cai, H. Xu and M. Xu, "Security-aware virtual network embedding," in *Proc. of IEEE ICC*, pp. 834-840, 2014. Article (CrossRef Link).
[21] S. Liu, Z. Cai, H. Xu and M. Xu, "Towards security-aware virtual network embedding," *Computer Networks*, vol. 91, pp. 151-163, 2015. Article (CrossRef Link).
[22] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran and S. Guizani, "Securing software defined networks: taxonomy, requirements, and open issues," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 36-44, 2015. Article (CrossRef Link).
[23] A. Akhunzada , A. Gani, N.B. Anuar, A. Abdelaziz, M.K. Khan, A. Hayat, et al., "Secure and dependable software defined networks," *Journal of Network & Computer Applications*, vol. 61, pp. 199-221, 2015. Article (CrossRef Link).
[24] S. Song, K. Hwang and Y. Kowk, "Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling," *IEEE Transactions on Computers*, vol. 55, no. 6, pp. 703-719, 2006. Article (CrossRef Link).
[25] A. Schrijver, "Theory of linear and integer programming," NewYork, NY, USA: Wiley, 1998.
[26] D. Eppstein, "Finding the k shortest paths", in *Proc. of the 35th IEEE Annual Symposium on Foundations of Computer Science*, pp. 154–165, 1994. Article (CrossRef Link).
[27] M. Bianchini, M. Gori, and F. Scarselli, "Inside PageRank," *ACM Transactions on Internet Technology*, vol. 5, no. 1, pp. 92–128, 2005. Article (CrossRef Link).
[28] E. Zegura, K. Calvert and S. Bhattacharjee, "How to model an Internetwork," in *Proc. of IEEE INFOCOM*, March 24-28, 1996. Article (CrossRef Link).

Shuiqing Gong is a Ph.D. candidate in Computer Application Technology from Air Force Engineering University, China. His research interests include network virtualization, software defined networking and cloud computing.

Jing Chen is a professor at Air Force Engineering University, China. She received her B.S. degree in Computer Science and Technology in 1985 from Air Force Engineering University, China, her M.S. degree in Computer Science and Technology in 1994 from Northwestern Polytechnical University, China, and her Ph.D. degree in Communication and Information System in 2005 from Xi'an Jiaotong University, China. From 2011 to 2012, she was a visiting scholar at the Department of Computer Science, Ohio State University. Her research interests include next generation Internet, cloud computing and software defined networking.

Conghui Huang is an engineer at Unit 94543 of PLA, China. He recevied his Ph.D. degrees in Computer Application Technology from Air Force Engineering University, China. His research interests include network virtualizaiton and cloud computing.

QingChao Zhu is a Ph.D. candidate in Computer Application Technology from Air Force Engineering University, China. His research interests include network virtualization and cloud computing.

Siyi Zhao is a postgraduate in Computer Science from Air Force Engineering University, China. Her research interests include network virtualization and software defined networking..