Security-aware Virtual Network Embedding

Shuhao Liu*, Zhiping Cai*, Hong Xu†, Ming Xu*
*School of Computer, National University of Defense Technology, Changsha, Hunan, P.R. China
†Department of Computer Science, City University of Hong Kong, Hong Kong, P.R. China

***Abstract*—Network virtualization is a promising technology to enable multiple architectures to run on a single network. However, virtualization also introduces additional security vulnerabilities that may be exploited by attackers. It is necessary to ensure that the security requirements of virtual networks are met by the physical substrate, which however has not received much attention thus far.**

This paper represents an early attempt to consider the security issue in virtual network embedding, the process of mapping virtual networks onto physical nodes and links. We model the security demands of virtual networks by proposing a simple taxonomy of abstractions, which is enough to meet the variations of security requirements. Based on the abstraction, we formulate security-aware virtual network embedding as an optimization problem, proposing objective functions and mathematical constraints which involve both resource and security restrictions. Then a heuristic algorithm is developed to solve this problem. Our simulation results indicate its high efficiency and effectiveness.

I. Introduction

Network virtualization is one of the most important technologies for next-generation Internet. It is proposed to accelerate innovations and to provide more stable services, by enabling new protocols and topologies to be rapidly implemented upon existing network infrastructures [1]. For example, many research testbeds rely on network virtualization to experiment new architectures [2].

Making efficient use of the substrate resources in virtualization requires effective techniques of virtual network embedding [3]. Virtual network embedding is essentially a resource allocation problem, where a new virtual network, with constraints on the virtual nodes and links, is mapped onto specific physical nodes and links in the substrate network. Because of the combination of node and link constraints, and the diversity of virtual topologies, virtual network embedding is NP-hard and computationally intractable even in the offline cases [4], and many heuristic and meta-heuristic algorithms have been developed for specific formulations.

In this paper, we study virtual network embedding from a different perspective. We consider security, which is an important yet largely overlooked aspect in the literature. To protect all virtual networks from potential threats and to guarantee information confidentiality and integrity, in many cases users have specific security demands and requirements that have to be satisfied. That is, virtual networks need to be embedded onto physical nodes and links with a qualified set of protection mechanisms. For example, each virtual machine of a virtual network must be allocated onto an end system with qualified firewalls, certain data encryption functions, etc. Security requirements intuitively make the problem even more difficult due to the additional complexity of considering network resource sharing and vulnerability of current virtualization architectures. However, to our knowledge there is little work that focuses on the security aspect of virtual network embedding thus far.

This work represents our first attempt in making virtual network embedding *security-aware*. To this end, we make three concrete contributions. First, we propose a taxonomy of abstractions to properly model the security demands of virtual networks. A concept of security level is introduced to capture the availability of different protection mechanisms in the substrate. Then the security demand of a virtual node or link is expressed in terms of security levels, and can be satisfied with physical resources that can offer the same or a higher security level. This simple abstraction is general enough to embrace many distinct forms of security mechanisms and requirements.

Second, we develop an optimization framework for security-aware network embedding, by considering both the resources and security demands of virtual networks. We present three objective functions, focusing on three different major concerns of network operators. Moreover, apart from resource constraints, such as node CPU capability and link bandwidth consumptions, we propose the security constraints, based on the analysis of vulnerabilities in the current virtual network architecture.

Third, we propose a novel heuristic algorithm to solve the security-aware network embedding problem. Given the conventional embedding formulation without security constraints is NP-hard, our problem with security constraints is even more complicated. To this end, we design a novel heuristic to estimate the possibility and capability of each physical node to host a given virtual node. It involves security satisfaction and node interconnection relationship in the iterative computations. Simulation results indicate that our algorithm achieves good performance with reasonable execution time, and it is practical in large-scale real-time virtualization systems.

The rest of the paper is organized as follows. Sec. II summarizes related works in the literature. In Sec. III, the security threats and requirements of virtual networks are discussed. Then, we introduce our abstraction of security demands and formulation of the security-aware embedding problems. Sec. V proposes our algorithm with a novel heuristic. Experiment results shown in Sec. VI indicate that our method achieves good performance. Finally, Sec. VII concludes the paper.

II. Related Works

A rich literature exists for virtual network embedding. Most work focuses on the general embedding problem and propose different formulations with specific objectives or constraints. Yu et al. in [5], for example, propose a seminal algorithm that enables link splitting and migration. Su et al. in [6] focus on the energy-aware virtual network embedding problem. Cai et al. in [7] focus on redeploying virtual resources and minimizing the upgrading cost in the scenario of evolving networks.

Security constraints for virtual network embedding have been briefly discussed in the literature. However, to the best of our knowledge, none of the existing work has proposed an effective and applicable way of either formulating or solving the problem, as we explain below.

In [8], Fischer proposes some security issues of end systems in a virtual network. Three security demands and constraints are concluded, but no solution algorithm is presented in the paper. Also, only virtual node security issues are analyzed. Link security threats, which need additional mechanisms to deal with and make the problem more complex, are ignored.

Bays et al. make progress in modeling the security-aware resource allocation in [9]. The authors enumerate several examples of security demands, such as link encryption and exclusiveness among virtual resources. These demands are then divided into different sets. However, only the correctness of the proposed model is validated, but no practical algorithm is developed.

III. The Security-aware Virtual Network Embedding Problem

A. Virtual Network Embedding Problems

Network virtualization is a powerful tool that enables multiple users to share the same physical resources simultaneously and to exploit abstracted topologies and functions. Due to the requirement of isolation and the limitation of physical resources, it is a natural question that how we would be able to embed virtual resources in an effective, expedient and serviceable way, that is, the *virtual network embedding* problems.

Typically, virtual network embedding could be abstracted as a resource allocation problem, to find the optimal mappings between a sequence of virtual network requests and a given substrate network. A single virtual network request is defined by its life span and virtual network topology. The life span indicates its demanding start and end time of occupying physical resources. The topology, including virtual nodes and links, is annotated with constraints, representing their demands for host physical resources. Note that a virtual node is typically in the form of an isolated Virtual Machine (VM), hosted by an end system that is usually abstracted as a substrate node. A virtual link is usually mapped to a substrate path.

Virtual network embedding problems are complicated. On the one hand, embedding operations shall be constrained according to the physical capabilities and the demands of virtual network requests, which makes it NP-hard and computationally intractable. QoS constraints are one of the typical forms. On the other hand, its computation efficiency is a practical requirement for network operators due to its real-time nature. Hence, designing heuristics is quite challenging but important.

B. Security Issues in Virtual Networks

Network virtualization could be a two-edged sword. As an additional virtualization layer and multiple shared VMs are introduced into the architecture of end systems, more complexity is incurred by network virtualization. Network operators are able to get more flexibility of the network in trade of potential attack vectors [8]. Apart from traditional vulnerabilities, virtual networks suffer from additional security hazards in the following aspects.

First, VMs would be easily attacked if their physical host was occupied by adversaries. The VMs being attacked would not be able to defend themselves, because VMs are always supervised by their hosts in all aspects. Second, an unauthorized VM may attack its host or another VM on the same host. The attacking VM may escape from the rigid confinement created by the virtualization process [8]. Third, the attackers can perform side-channel attacks [10] or negatively influence the whole system, such as launching Denial-of-Service attacks.

Not only nodes but also links do suffer from additional security threats. Adversaries may influence the physical links in a negative way (e.g. replay attacks). Also, it is possible that substrate routers and switches are attacked. Due to the migration and splitting nature of virtual links [5], the above two issues cannot be ignored, as a virtual link with high security demands may be embedded onto substrate resources without adequate protection. As a consequence, it is a necessity to consider security requirements of virtual networks during the process of embedding.

C. The Abstraction of Security Constraints

In order to abstract the security requirements of virtual networks, we introduce the numerical concept of security level. The security levels indicate the abstracted standard of protection, being assigned by network operators. The higher levels they are able to offer, the more protection mechanisms are available. For example, a substrate node that enables data encryption and digital signature would be assigned a higher security level than those do not.

Security demands then could be expressed in terms of security levels. Apparently, a certain demand would be satisfied by resources that have been assigned equal or higher security levels, because they are capable of a larger set of protection mechanisms.

Based on the assumptions above, we conclude four abstracted security constraints as below, and the constraints proposed in [8] have been included. Note that the security level of a substrate path will be determined by the minimum level of all links and nodes included.

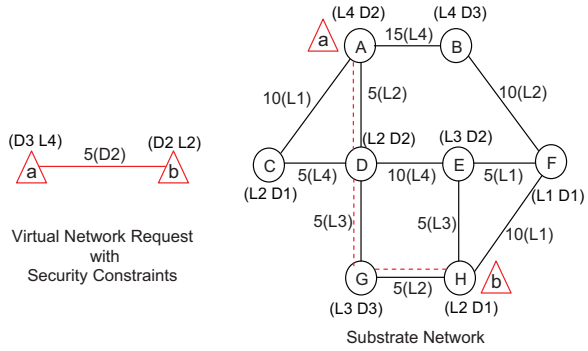

Fig. 1. An example of virtual network embedding problems with security constraints.

1) A substrate node should guarantee a security level that is higher than the demand of every guest node.
2) A virtual node should guarantee a security level that is higher than the demand of its host node.
3) Each virtual node should provide all other virtual nodes of the same host with an adequate security level.
4) A virtual link with a certain security demand should be hosted by a substrate path with an adequate security level.

Figure 1 depicts a simple example of the security-aware virtual network embedding problem. We assume an adequate CPU capability of every physical node, and we omit them in the figure to emphasize the security constraints. In the figure, `(D3, L4)` at side of a node indicates that the corresponding resource ensure a security level of 4, and require its host or guest to offer a security level not lower than 3. `5(D2)` indicates that the annotated link require 5 unit bandwidth and demand for a host path that offers a security level of 2.

Without the security constraints, the example virtual network request could be embedded onto any pair of physical nodes and any path between them in the figure. However, when considering security, the mapping result is shown by the red dotted line. Virtual nodes a, b and link ab are mapped onto A, H and the path $ADGH$, respectively. C cannot be a host of a because of insufficient security level. $ABFH$ would be a host of ab only if HF offered a security level of 2 or higher.

IV. Formulation of the Security-aware Virtual Network Embedding Problem

A. Mathematical Definitions

1) Substrate Networks: A substrate network can be abstracted as a weighted undirected graph $G^S = (N^S, L^S, A_N^S, A_L^S)$, where N_S is the set of all substrate nodes and L_S is the set of all substrate links. The annotation A_N^S and A_L^S denote the attributes of substrate nodes and links, respectively. When discussing the security problem, it is reasonable to simplify the distinct attributes to the following settings:

$$A_N^S = \{\{cpu^S(n), dem^S(n), lev^S(n)\}|n \in N^S\}$$
$$A_L^S = \{\{bw^S(l), lev^S(l)\}|l \in L^S\}$$

For a given $n \in N^S$, $cpu^S(n), dem^S(n)$ and $lev^S(n)$ denote the CPU capacity, security demand and security level of n, respectively. Similarly, $bw^S(l)$ and $lev^S(l)$ are the available bandwidth and the security level of each substrate link l ($\forall l \in L^S$). Additionally, We use notation P^S to represent the set of all paths in G^S.

2) Virtual Network Requests: Virtual network requests are organized in order of arriving time:

$$G^V = \{G_1^V, G_2^V, \ldots, G_k^V\}$$

The No.i request can be described as another weighted undirected graph $G_i^V = (N_i^V, L_i^V, Time_i^V, Dur_i^V, C_{i,N}^V, C_{i,L}^V)$. It arrives at time $Time_i^V$ and lasts a time period of Dur_i^V. Similar to the description of substrate resource attributes, the virtual network requirements are represented as follows:

$$C_{i,N}^V = \{\{cpu_i^V(n), dem_i^V(n), lev_i^V(n)\}|n \in N_i^V\}$$
$$C_{i,L}^V = \{\{bw_i^V(l), dem_i^V(l)\}|l \in L_i^V\}$$

For a given $n \in N_i^V$, $cpu^V(n), dem^V(n)$ and $lev^V(n)$ denote the demanded CPU, required security demand and security level, respectively. $bw^V(l)$ and $dem^V(l)$ are the bandwidth demand and the security demand of each virtual link l ($\forall l \in L_i^V$).

3) The Embedding Operations: Based on the descriptions above, we can define the embedding operations as a sequence of mappings:

$$\mathbf{M} = \{M_1, M_2, \ldots, M_i, \ldots\}$$

$\forall i \in \{1, 2, 3, \ldots\}$, M_i is the mapping of request G_i^V.

$$M_i : G_i^V \rightarrow G_i^S = (N_i, P_i, A_{i,N}, A_{i,L}),$$

where $N_i \subseteq N^S, P_i \subseteq P^S$. $A_{i,N}$ and $A_{i,L}$ represent the attributes of substrate nodes and links in G_i^S, the redundant network of G^S at $Time_i^V$, just before trying to embed G_i^V.

Additionally, we use $M_{i,N} : N_i^V \rightarrow (N^S, A_{i,N})$ and $M_{i,L} : L_i^V \rightarrow (P^S, A_{i,L})$ to describe the two stages of M_i, that is, node mapping and link mapping, respectively.

4) Other Functions: In order to describe our model explicitly, a two-value function $\rho(i)$ is defined to indicate whether a single request is successfully embedded or not.

$$\forall i \in \{1, 2, \ldots, |M|\}, \rho(i) = \begin{cases} 1, \text{if request No.}i \text{ accepted} \\ 0, \text{if request No.}i \text{ denied} \end{cases} \quad (1)$$

Additionally, we propose two sets of variants $X_i = \{x_{i,qr}|n_q \in N_i^V, n_r \in N^S\}$ and $Y_i = \{y_{i,qr}|l_q \in L_i^V, p_r \in P^S\}$ to express the formulation in a simple way. For a given virtual network request G_i^V, we define $x_{i,qr} \in \{0, 1\}$ as an element of X_i to indicate the node relationships. If a virtual node n_q is mapped onto the substrate node n_r, then $x_{i,qr} = 1$. Otherwise, $x_{i,qr} = 0$. Moreover, $y_{i,qr} \in Y_i$ is defined in a similar way, indicating the ratio of bandwidth allocation, so we have $y_{i,qr} \in [0, 1]$. For example, $y_{i,qr} = 0.5$ means that half bandwidth of virtual link request l_q is mapped onto each substrate link of path p_r.

B. The Objective Functions

Operators of virtual networks always try to maximize the long-term profit of virtual network embedding operations, which involves both revenue and cost.

1) The Revenue Functions: The revenue of a mapping $M_i \in \mathbf{M}$ can be described as below:

$$Rev(M_i) = \rho(i) \cdot Dur_i^V \cdot [\sum_{n_i^V \in N_i^V} dem^V(n_i^V) cpu^V(n_i^V) + \sum_{l_i^V \in L_i^V} dem^V(l_i^V) bw^V(l_i^V)] \quad (2)$$

Intuitively, a request with higher security demands achieves a higher revenue. Therefore, we take both node and link security demands into account, as is shown in the equation.

2) The Cost Functions: The cost of a mapping $M_i \in \mathbf{M}$ can be described as follows:

$$Cost(M_i) = \rho(i) \cdot Dur_i^V [\sum_{n_i^V \in N_i^V} lev^S(M_{i,N}(n_i^V)) cpu^V(n_i^V) + \sum_{l_i^V \in L_i^V} lev^S(M_{i,L}(l_i^V)) len(M_{i,L}(l_i^V)) bw^V(l_i^V)] \quad (3)$$

The function $len(p)$ in the equation indicates the number of hops through the path $p \in P^S$. Obviously, it would be a waste and would cost more to occupy substrate resources with an unnecessarily long path and an exorbitant security level.

3) The Objectives: Based on the assumptions above, we can conclude three different objectives of the security-aware virtual network embedding problem as below:

1) to maximize the request acceptance ratio, that is, the ratio of virtual network requests being successfully embedded;
2) to maximize the long-term revenue of the network;
3) to maximize the long-term Revenue to Cost Ratio (R/C Ratio) of the network.

Generally, to evaluate the embedding results, all of the three objectives are considered with different priorities, which vary among different practical situations.

C. The Model of the Security-aware Virtual Network Embedding Problem

Described as an optimization problem, the security-aware virtual network embedding problem is formulated with specific objectives and constraints. The objective is:

$$\max \quad \lim_{T \to \infty} \frac{\sum_{i=1}^{|\mathbf{M}|} Rev(M_i)}{T} \quad (4)$$

and the constraints are:

$$\sum_{r=1}^{|N^S|} x_{i,qr} = 1 \qquad \forall n_{i,q} \in N_i^V \quad (5)$$

$$\sum_{r=1}^{|P^S|} y_{i,qr} = 1 \qquad \forall l_{i,q} \in L_i^V \quad (6)$$

$$\sum_{i=1}^{|N|} x_{i,qr} cpu_i^V(n_{i,q}) \le cpu^S(n_r) \quad \forall n_{i,q} \in N_i^V, n_r \in N^S \quad (7)$$

$$\sum_{i=1}^{|N|} y_{i,qr} bw_i^V(l_{i,q}) \le \min_{l_j \in p_r} bw^S(l_j) \quad \forall l_{i,q} \in L_i^V, p_r \in P^S \quad (8)$$

$$x_{qr} dem^S(n_r) \le lev^V(n_q) \quad \forall n_q \in N_i^V, n_r \in N^S \quad (9)$$

$$x_{qr} dem^V(n_q) \le lev^S(n_r) \quad \forall n_q \in N_i^V, n_r \in N^S \quad (10)$$

$$\max\{dem^S(n_r), \max_{x_{qr}=1} dem^V(n_q)\} \le \min\{lev^S(n_r), \min_{x_{qr}=1} lev^V(n_q)\} \quad \forall n_q \in N_i^V, n_r \in N^S \quad (11)$$

$$dem^V(l_q) \le \min_{l_i \in p_r, y_{qr}>0} lev^S(l_i) \quad \forall l_q \in L_i^V, p_r \in P^S \quad (12)$$

The equation (4) is the formulation of the revenue objective addressed in Sec. IV-B3. The constraint (5) restricts each virtual node to be mapped onto a single substrate node, while (6) ensures that the bandwidth demand of each virtual link is shared by several substrate paths. The constraint (7) ensures that the host of virtual nodes can satisfy the guests' CPU requirements. The constraint (8) ensures that the bandwidth of each substrate link is not over-subscribed. The constraints (9) to (12) are highlighted: they are four security constraints corresponding to the constraint list in Sec. III-C.

V. Security-aware Virtual Network Embedding Algorithm

In this section, we propose a two-stage security-aware virtual network embedding algorithm, based on the formulation presented in Sec. IV-C. The two stages of the algorithm correspond to the problem decomposition of node mapping stage and link mapping stage, respectively. The separation of two stages simplifies the problem, but special mechanisms are required to avoid bad results.

The embedding problem is capable of being decomposed into a sequence of sub-problems, which is to embed one single virtual network onto the redundant substrate. In the following description, we focus on a single sub-problem to describe two stages in detail, then the framework of solving the overall problem is given.

A. The Heuristics

The principle of our algorithm is to design a heuristic for each substrate node to estimate its availability of hosting a given virtual node, and to guide the node mapping operations. Based on this estimated value, we are able to sort the substrate nodes in a reasonable order and then to perform a best-first search. Apparently, a more precise estimation will result in a better result, that is, achieving more optimized objectives. Also, the complexity of calculating this value cannot be too high because of frequent refreshment.

Intuitively, a substrate node with more CPU capability and broader outgoing bandwidth would be more available to host virtual nodes. Offering a higher security level would also

contribute, but, to avoid high cost based on equation (3), it would be the best to exactly match the security demands. As a result, the estimated value varies with different levels of security demands, and we need to compute each for different requests.

For a given security demand k, We design two variants to properly introduce the factors mentioned above. In particular, let n be the substrate node we are focusing on, and $Link(n)$ denotes the set of outgoing links of n, so $Link(n) \subseteq L^S$. $\forall l \in Link(n)$, we define the uniformed bandwidth bw_u_k:

$$bw\_u(l,k) = \begin{cases} bw(l)e^{lev(l)-k} & \text{if } lev(l) \geq k \\ 0 & \text{if } lev(l) < k \end{cases}$$

Similarly, the uniformed CPU of a substrate node n is defined:

$$cpu\_u(n,k) = \begin{cases} cpu(n)\frac{1-(lev(n)-k)^2}{\delta} & \text{if } lev(n) \geq k \\ 0 & \text{if } lev(n) < k \end{cases}$$

δ is the coefficient to ensure its positive value.

As is done in [5], we model the intuitive estimated value of a given substrate node n at certain security demand k by using the product of its uniformed CPU and collective bandwidth of outgoing links, that is,

$$H^{(0)}(n,k) = cpu\_u(n,k) \sum_{l \in Link(n)} bw\_u(l,k) \tag{13}$$

The result of $H^{(0)}$ is called basic estimation, depending on a single node's related resources.

Furthermore, to make the estimation more accurate, additional information should be included. We take both virtual and substrate global topologies into consideration. Neighboring nodes interconnection factors in both virtual and physical networks are introduced after analyzing the following examples. First, assuming that a virtual node a has already been embedded onto substrate node A. For virtual node b, which connect to a directly in the same virtual network request, the availability of the neighbors of A would undoubtedly get a promotion. The second case is that a substrate node B with little basic estimated value would seem to be more available, only if B had a neighboring C with a high rank of estimation. Moreover, with broader interconnection bandwidth and a higher link security level, B would be more influenced by its neighbor, because the additional cost of link BC would be less.

Thus, an iterative mechanism is introduced. We propose a variant named propagate coefficient of each substrate link, to measure the neighboring nodes' influence on each other. For the nodes at both end of link l, $PC(l,k)$ is defined to evaluate propagate coefficient at the given security level k.

$$PC(l,k) = \frac{bw\_u(l,k)}{Max\_Prob\_BW}$$

The constant `Max_Prob_BW` denotes the maximum probable bandwidth of all links. The iterative computation process of the heuristic is defined as below:

$$H^{(t+1)}(n,k) = \lambda \sum_{(m,n) \in Link(n)} PC((m,n),k) H^{(t)}(m,k) + (1-\lambda) H^{(t)}(n,k) \tag{14}$$

where $t = 0, 1, 2, \ldots,$ `MaxIte` $- 1$. λ is a bias factor, and we typically set it to 0.15. Additionally, we take the uniformed link bandwidth into account, instead of just focusing on neighboring node resources in [11]. `MaxIte` defines the number of iteration rounds, and we set it to $\lfloor \sqrt{|N^S|} \rfloor$, the square root of the number of substrate nodes, so that topology information would be able to spread out the network.

B. Node Mapping Algorithm

The node mapping algorithm is described in Algorithm 1. It is called to try to properly embed all virtual nodes of a single virtual network request G_i^V to the redundant substrate G_i^S, while ensuring high revenue and low cost during the second stage of link mapping. That is, for a given request G_i^V, Algorithm 1 aims at getting a proper mapping $M_{i,N}$ based on the heuristic $H^{(MaxIte)}$. $H^{(MaxIte)}$ is calculated by using equation (13) and (14) in terms of current security demand k.

Algorithm 1 Node Mapping Algorithm

1: For each possible security demand k, sort all substrate nodes in a candidate node queue $queue(k)$ in descending order of $H^{(MaxIte)}$.
2: For all nodes $m \in G_i^S$, initial their state by setting $Occupied(m) =$ FALSE.
3: **repeat**
4: Get an unmapped node n randomly from G_i^V.
5: $k = dem^V(n)$.
6: **if** exist the first node m in $queue(k)$ satisfying $Occupied(m) =$ FALSE **and** $dem^S(m) \leq lev^V(n)$ **and** $cpu^S(m) \geq cpu^V(n)$ **then**
7: $Occupied(m) =$ TRUE.
8: Map the virtual node n onto the substrate node m.
9: **else**
10: Release all resources occupied by G_i^V.
11: **return** `MAP_FAILED`.
12: **until** all nodes in G_i^V are mapped successfully.
13: **return** `NODE_MAP_SUCCESS`.

C. Link Mapping Algorithm

If node mapping stage ends up with a success, we turn to map the virtual links, using Algorithm 2. As virtual nodes have become fixed in the substrate, what we need to do is to find out the available substrate paths among there hosts. There is no doubt that our goal is to get the substrate path between virtual nodes with the lowest cost instead of smallest number of paths hops. To this end, a variant named Path Cost Coefficient (PCC) is designed in the place of paths hops. Considering link security demand k, the Path Cost Coefficient of a substrate path $p \in P^S$ is defined by following equation:

$$PCC(p,k) = \sum_{l \in L^S, l \in p} [lev^S(l) - k + 1]$$

The prerequisite of link mapping algorithm is that the state of current request is already NODE_MAP_SUCCESS. The constant MAX_SPLIT_TIME, which indicates the upper limit of link split times, is defined to avoid the fragmentation of splittable link mapping and to limit the algorithm complexity. The variant $flag$ is an indicator of overall link mapping state.

Algorithm 2 Link Mapping Algorithm

for each unmapped virtual link $l \in G_i^V$ **do**
  $bw_r = bw^V(l), k = lev^V(l), flag = 0.$
  if l is splittable **then**
    $split = 1.$
  else
    $split =$ MAX_SPLIT_TIME.
  Let $m_1, m_2 \in G^S$ be the hosts of both ends of l.
  repeat
    if exist $p \in P^S$ **and** $p : m_1 \rightarrow m_2$, with the minimum $PCC(p, k)$ **then**
      $bw^S(p) = \min_{t \in p} bw^S(t), t \in L^S.$
      Map the remaining resources of l onto p.
      $bw_r = bw_r - bw^S(p).$
      if $bw_r \geq$ **then**
        $flag = 1.$
    else
      $split =$ MAX_SPLIT_TIME $+ 1.$
  until $flag = 1$ **or** $split >$ MAX_SPLIT_TIME
  if $flag = 0$ **then**
    return MAP_FAILED.
return MAP_SUCCESS.

D. The Algorithm Framework

1) Framework Description: To be applied in real-time scenario, dealing with both new-coming and suspended requests, our algorithm framework is designed to be called once in every fixed time interval.

First, the algorithm scans all of the online virtual network requests. Second, these requests are sorted in descending order of their revenues. This process is the preparing work of greedily deciding embedding priorities. Third, the sub-algorithms of both node and link mapping are called in turn, to try embedding the awaiting virtual network request with the maximum revenue and then to refresh the redundant networks. Finally, after trying all requests, the procedure is terminated.

2) Time Complexity Analysis: As a sub-process to embed a single virtual network request onto the redundant substrate network, both node and link mapping algorithm need to be called once. Based on the descriptions above, we conclude that node mapping is a polynomial-time algorithm. The sub-process of calculating estimated value and sorting has the time complexity of $O(|N^S|^{\frac{3}{2}})$, while node mapping is $O(|N^S| \cdot |N^V|)$. Also, link mapping can be solved in $O(|N^S|^3 \cdot |L^V|)$. Finally, considering the procedures of detecting request states and allocating, releasing resources have linear complexity, our framework is a polynomial-time algorithm in terms of $|N^S|$, $|N^V|$, $|L^V|$ and the number of virtual requests.

VI. Performance Evaluation

In this section, the performance of our security-aware virtual network embedding algorithm is evaluated. We first describe the different simulation settings, and then present our result of evaluation.

A. Evaluation Environments

1) Network Settings: We generate all virtual network requests and substrate network topologies by using the GT-ITM tool [12], adopting similar parameters with [5].

The substrate is set to have 100 nodes and around 500 links, a scale that corresponds to a medium sized ISP. The CPU and bandwidth capability of the substrate nodes and links are real numbers uniformly distributed between 50 and 100. The security level of both nodes and links are integer numbers uniformly distributed between 0 and 4. The security demands of substrate nodes are also integers varying from 0 to 4. They are not distributed uniformly, because the security demand of a single node cannot be higher than its security level.

The number of nodes in each request topology is uniformly distributed between 2 and 20. The average link connectivity rate is 50%, which is determined by the α parameter of GT-ITM. The CPU and bandwidth requirements of virtual nodes and links are real numbers uniformly distributed between 0 and 50. The virtual network requests are reconfigured by an extension to the GT-ITM tool, which arranges the topologies into a sequence and gives each of them a request time and duration. We assume that requests arrive following a Poisson process with an average arrival rate of 5 requests per 100 time units. The duration of each follows an exponentially distribution with an average of 500 time units. Our simulation involves 1500 requests per instance, so that the total time of simulation would be about 30000 time units.

2) Comparison and Objectives: Three algorithms listed below are included in our evaluation to test the performances. We will compare the evaluation results of all three objectives listed in Sec. IV-B3, based on the same hardware and software platform.

SAV: Our Security-Aware Virtual network embedding algorithm, with detailed description in Sec. V. The two-stage algorithm is based on the heuristic of $H^{(MaxIte)}$, which is calculated by using Equation 13 and 14.

NI-SAV: The Non-Iterative SAV, that is, using $H^{(0)}$ instead of $H^{(MaxIte)}$ as heuristic. Without considering factors of neighboring nodes, it is used to test the effectiveness of iterative computation of estimated values.

BL: The Baseline algorithm. It is an extension to Yu's project in [5]. We have simply revised it by adding constraints and updating the computation of embedding revenue and cost, using Equation 2 and 3.

B. Evaluation Results and Discussion

1) Objectives Comparison: We generated 11 different test sets of virtual network requests (with link splittable ratio at 60%) and substrate networks to evaluate the performance of

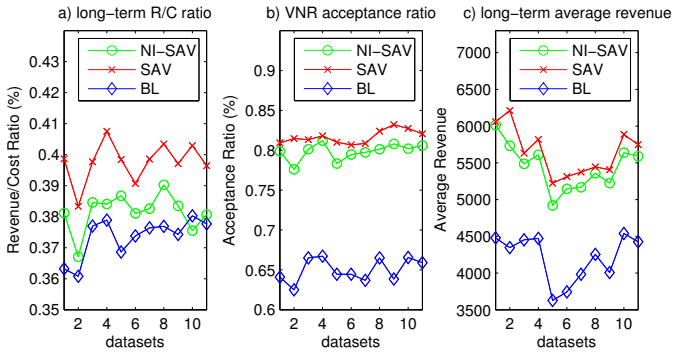

Fig. 2. Simulation results of 11 data sets. Splittable Rate = 0.6.

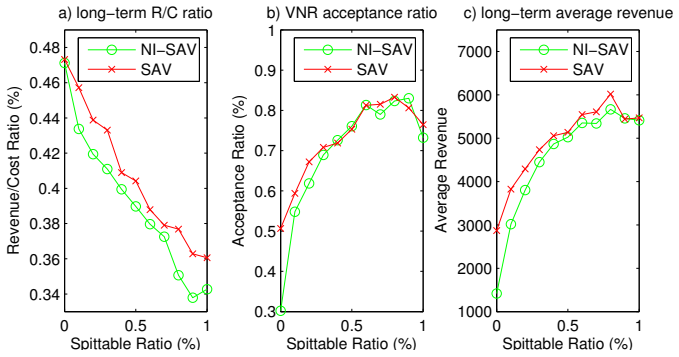

Fig. 3. Algorithm performance under different splittable rate.

different algorithms. Figure 2 depicts that both of our algorithms (SAV and NI-SAV) achieve higher R/C Ratio (Figure 2 (a)), higher request acceptance ratio (Figure 2 (b)) and higher long-term average revenue (Figure 2 (c)), compared with BL.

Besides, the iterative computation process of estimated values has much elevated the accuracy of estimation. SAV always achieves the best performances among the three.

2) Performance with Varied Splittable Ratio: In order to evaluate the influence of splittable link ratio on the algorithm process, we generated another 11 sets of requests, with splittable ratios ranging from 0% to 100%. The simulation results are shown in Figure 3. The figure indicates the poor performance of our algorithms when few virtual link allows splitting. When no link is splittable, SAV performs much better because of the accuracy of estimation. When splittable rate is above 30%, both SAV and NI-SAV are able to achieve acceptable performance.

Note that there is a drop in both request acceptance ratio and long-term average revenue when the splittable rate is very high ($> 80\%$). We think the cause of this phenomenon is that the setting of the constant `MAX_SPLIT_TIME` is too small. If our virtual network requests have high splittable rate, we need to increase the value properly, hence to achieve a higher acceptance ratio and more revenue. However, More virtual links being split means more high-cost substrate paths being utilized, therefore the R/C Ratio keeps decreasing, as is depicted in Figure 3 (a).

3) Execution Time Comparison: Figure 4 indicates that our algorithms need much less execution time, which ensures real-time performance in large-scale scenarios. The iterative computation of estimated value greatly enhances the performance

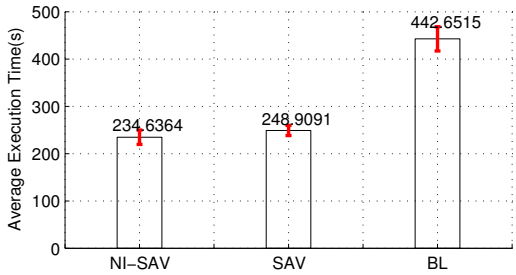

Fig. 4. Algorithm execution time comparison.

at the cost of reasonable time complexity.

VII. Conclusion

In this paper, we address the security requirements of virtual network embedding. The numerical concept of security levels and security demands are proposed to properly abstract the security requirements. We formulate the problem as an optimization problem, proposing three objectives with both resource and security constraints. In our design of the heuristic algorithm, the innovation of incurring global topologies and interconnection information is highlighted. The evaluation results demonstrate its effectiveness and practicality.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant No. 61070198, 61379145, 61170288, 61379144.

References

[1] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, pp. 1–19, 2013.

[2] J. Carapinha and J. Jiménez, "Network virtualization: a view from the bottom," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*. ACM, 2009, pp. 73–80.

[3] A. Haider, R. Potter, and A. Nakao, "Challenges in resource allocation in network virtualization," in *20th ITC Specialist Seminar*, vol. 18, 2009.

[4] D. G. Andersen, "Theoretical approaches to node assignment," *Computer Science Department*, p. 86, 2002.

[5] 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, 2008.

[6] S. Su, Z. Zhang, X. Cheng, Y. Wang, Y. Luo, and J. Wang, "Energy-aware virtual network embedding through consolidation," in *INFOCOM 2012 Computer Communications Workshops*. IEEE, 2012, pp. 127–132.

[7] Z. Cai, F. Liu, N. Xiao, Q. Liu, and Z. Wang, "Virtual network embedding for evolving networks," in *Global Telecommunications Conference (GLOBECOM 2010)*. IEEE, 2010, pp. 1–5.

[8] A. Fischer and H. de Meer, "Position paper: Secure virtual network embedding," *Praxis der Informationsverarbeitung und Kommunikation*, vol. 34, no. 4, 2011.

[9] L. Bays, R. Oliveira, L. Buriol, M. Barcellos, and L. Gaspary, "Security-aware optimal resource allocation for virtual network embedding," in *Network and service management (cnsm)*, 2012, pp. 378–384.

[10] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 199–212.

[11] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, 2011.

[12] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *INFOCOM'96*, vol. 2. IEEE, 1996, pp. 594–602.