Secure Virtual Network Embedding in a Multi-Cloud Environment

Max Alaluna* Luís Ferrolho* José Rui Figueira† Nuno Neves* Fernando M. V. Ramos*
LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal*,
CEG–IST, Instituto Superior Técnico, Universidade de Lisboa, Portugal†
{malaluna,lferrolho}@lasige.di.fc.ul.pt, figueira@tecnico.ulisboa.pt, nuno@di.fc.ul.pt, fvramos@ciencias.ulisboa.pt

***Abstract*—Recently-proposed virtualization platforms give cloud users the freedom to specify their network topologies and addressing schemes. These platforms have, however, been targeting a single datacenter of a cloud provider, which is insufficient to support (critical) applications that need to be deployed across multiple trust domains while enforcing diverse security requirements. This paper addresses this problem by presenting a novel solution for a central component of network virtualization – the online network embedding, which finds efficient mappings of virtual networks requests onto the substrate network. Our solution considers security as a first class citizen, enabling the definition of flexible policies in three central areas: on the communications, where alternative security compromises can be explored (e.g., encryption); on the computations, supporting redundancy if necessary while capitalizing on hardware assisted trusted executions; across multiples clouds, including public and private facilities, with the associated trust levels. We formulate the solution as a Mixed Integer Linear Program (MILP), and evaluate our proposal against the most commonly used alternative. Our analysis gives insight into the trade-offs involved with the inclusion of security and trust into network virtualization, providing evidence that this notion may enhance profits under the appropriate cost models.**

***Keywords*-virtual network embedding; multi-cloud; security**

I. Introduction

Network virtualization has emerged as a powerful technique to allow multiple heterogeneous virtual networks to run over a shared infrastructure. Nowadays, a number of production-level platforms have been proposed [2, 10], already achieving the necessary scale, performance, and required level of service. This has allowed cloud operators to start extending their service offerings of virtual storage and compute with network virtualization [10].

So far, these modern platforms have been confined to a datacenter, controlled by a single cloud operator. This restriction can be an important barrier as more critical applications start shifting to the cloud. To overcome this problem, we are developing a solution that aims to extend network virtualization across multiple cloud providers [3, 11], bringing a number of benefits in terms of cost, performance, and versatility. In particular, a multi-cloud solution may contribute to security from several perspectives. For example, a tenant[1] that needs to comply with privacy legislation can demand a certain virtual machine to remain at a specific place while the rest can go to other facilities (e.g., some services of a healthcare application, such as the analysis of patient medical images, can only be performed in pre-approved clouds). An application can also be made immune to any single datacenter (or cloud availability zone) outage by spreading its services across providers. Several incidents in cloud facilities are evidence of this increasingly acute risk [7, 15], motivating the exploration of availability-enhancing alternatives (e.g., the services deemed critical are replicated over two providers).

This paper tackles a fundamental component in our network virtualization solution – the *Virtual Network Embedding (VNE)* – from this new perspective. VNE addresses the problem of provisioning the virtual networks specified by the tenants [6]. When a virtual network request (VNR) arrives, the goal is to find an effective mapping of the virtual nodes and links onto the substrate network, while maximizing the revenue of the virtualization operator. This objective is subject to various constraints, such as the processing capacity on the substrate nodes and bandwidth resource on the links.

A mostly unexplored perspective on this problem is providing security assurances. We propose a VNE solution that considers security constraints based on indications from the tenants. These constrains address, for instance, concerns about attacks on virtual machines or on physical links (e.g., replay/eavesdropping). To further extend the resiliency properties of our solution, we support the coexistence of resources (nodes/links) in multiple clouds, both public and private, and assume that each individual cloud may have distinct levels of trust from a user standpoint. A private datacenter is in principle considered more trustworthy, while different public clouds may offer various levels of security and trust (due to their location and/or Service Level Agreements (SLA)).

We have evaluated our proposal against the most commonly used alternative [4]. The results show a similar behavior of both solutions when no security constrains are specified. Even when a significant part of the VNR impose security restrictions, the performance decrease is limited. This is a direct consequence of being harder to fulfill the requirements of the VNR, leading to lower acceptance rates. A key finding is that by rising the price of security services by a modest value, the provider is able to attain at least the same revenue as the other approach (certainly, a provider offering such added-value services will want to increase its profits further, so that figure can be seen as a baseline).

The contributions of our work can be summarized in the following: (i) We formulate the SecVNE problem and solve it as a Mixed Integer Linear Program (MILP). The novelty of our approach is in considering comprehensive security aspects over a multi-cloud model; (ii) We compare our solution with the most commonly used VNE MILP formulation [4], and analyze the trade-offs between the benefit of security and embedding efficiency.

[1]We employ the terms user and tenant interchangeably in the paper.

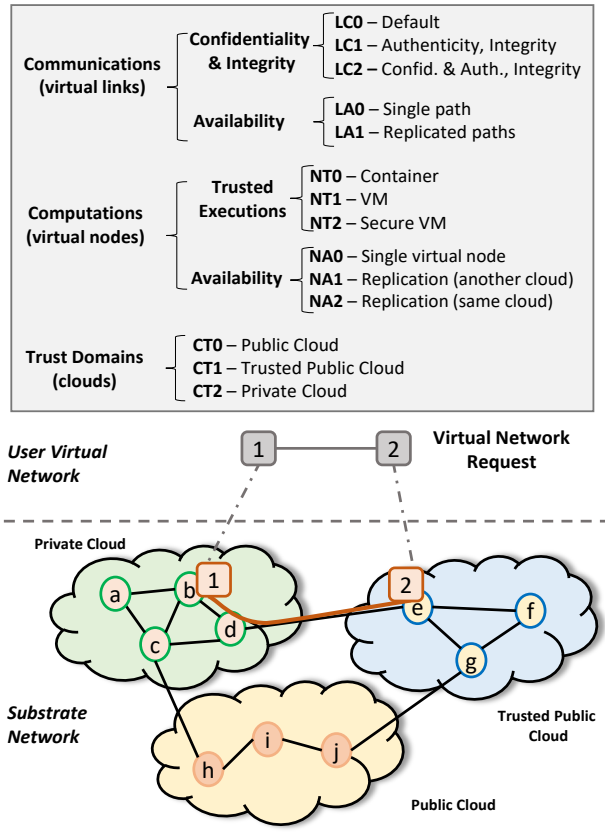


Figure 1: Model of levels of security for communications, computations and trust domains.

II. Secure VNE

Our approach to VNE supports the specification of security policies enforced on a multi-cloud deployment, enhancing the flexibility, efficiency and security of the network virtualization solution. When a user wants to instantiate a virtual network (VN), besides indicating the nodes' processing capacity and the links' bandwidth, he may also include as requirements security demands. These demands are defined with the selection of security levels/attributes associated with resources of the VN. The current collection of attributes resulted from conversations with several companies from the healthcare and energy sectors that are moving their critical services to the cloud, and they represent a balance among three goals: (i) expressive enough to represent the main security requirements when deploying a VN; (ii) easy to specify when configuring a VN (requiring a limited number of options); (iii) implementable with ready available technologies. Figure 1 displays the attributes that can be selected in our virtualization solution:

Communications. Two aspects can be tuned for each of the virtual links: (1) the user may desire to employ standard tunnels while connecting certain nodes (LC0 for default security), but in others extra safeguards may be required. Since data encryption normally imposes a non-negligible performance penalty, we let the user choose if authentication and integrity protection is enough (LC1) or if confidentiality is also necessary (LC2); (2) to tolerate failures, we allow for backup paths to be setup for extra availability assurances (LA0, where only a path is created vs. LA1, with two paths).

Computation. Virtual nodes can be embedded onto machines with distinct support for trusted executions: containers, e.g., Docker, have applications isolated directly on top of the host operating system (NT0); VMs achieve a higher level of security as isolation is ensured by the hypervisor (NT1); secure VMs could run in machines with hardware that can enhance the trust on the executions (e.g., TPMs or Intel SGX extensions)[2] (NT2). Since failure repair times are unpredictable [8], we also let the user specify that a node should be replicated. He can choose the location of each backup node, in the same cloud (NA2) or a different one (NA1) for physical failure independence.

Trust domains. The user can choose the type of cloud where virtual nodes should be located. Three types of clouds are supported: public clouds, belonging to external providers (CT0); trusted public clouds that give higher security assurances (e.g., greater penalties for SLA violations) (CT1); and private clouds, where the user has higher control on security (CT2). With these options, the user may choose private clouds for more sensitive VMs, leaving the others for public clouds to scale out.

Taking into consideration the above features, we now define the Secure Virtual Network Embedding (SecVNE) problem:

SecVNE problem: *Given a virtual network G^V with the requested resources and corresponding security requirements, and the substrate network G^S with the resources to serve incoming VNRs, can G^V be mapped to G^S with the minimum use of resources while satisfying the following constraints? (i) Each virtual link is mapped to the substrate network meeting the bandwidth and security constraints, namely related to confidentiality, integrity and availability; (ii) Each virtual node is mapped to the substrate network meeting the CPU capacity and security constraints, namely with regard to trusted executions and availability; (iii) Each virtual node is mapped to a substrate node located in a cloud that covers its trust domains requirements.*

Our solution handles the SecVNE problem, trying to map a VN onto a substrate network (SN) while respecting all the requirements and constraints. When a VNR arrives, the best mapping is searched for to decrease the costs of embedding (i.e., reduce the total quantity of substrate resources allocated to it). If there is no possible solution to embed the incoming VN, then the request is rejected. Otherwise, the quantity of resources demanded by the VN is allocated.

In our formulation, after the embedding of a VN, the SN is augmented with the virtual nodes that were embedded. These virtual nodes have meta-links to the substrate nodes on which they are mapped. Figure 2(c), illustrates the result of embedding the VNR presented in 2(b) onto the SN presented in 2(a). In the example, the VNR requires the replication of nodes and links for higher availability. Virtual node 1 has a meta-link with substrate node A, and virtual node 2 has one with E, which are their primary nodes. They also have meta-links with substrate nodes B and D, the backup nodes that can be used when a failure occurs. In this figure it is also possible to observe that the substrate paths (both working and backup) correspond to more than one substrate link (e.g., the substrate links (A,F) and (F,E) are assigned to the working path).

[2]Notice that we target also private clouds, where this support can be made readily accessible. Public clouds may have this sort of offering in the future, as out-of-the-box Intel CPUs already support SGX extensions.

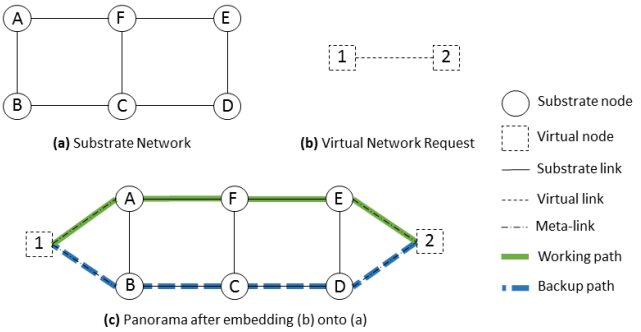


Figure 2: (a) Substrate network; (b) Virtual network request that requires node/path replication. (c) Example of the embedding result after the execution of our MILP formulation.

A. Network Model

This section describes the characteristics and attributes that are associated to a substrate and a virtual network.

1) Substrate Network

We model the substrate network as a weighted undirected graph. It is denoted by $G^S = (N^S, E^S, A_N^S, A_E^S)$, where N^S is the set of substrate nodes, E^S is the set of substrate links, A_N^S is the set of attributes of substrate nodes, and A_E^S is the set of attributes of substrate links.

A_N^S contains the following attributes for substrate nodes:

$$A_N^S = \{\{cpu^S(n), sec^S(n), cloud^S(n)\} | n \epsilon N^S\}$$

- $cpu^S(n)$ - Total CPU capacity of the substrate node *n*. This attribute can take any values greater or equal to 0;
- $sec^S(n)$ - Security provided by the substrate node *n*. This attribute can take any positive value (including zero). In the example given in Figure 1, if substrate node *n* is a normal container, then $sec^S(n) = 0$. If it is a normal VM or a secure VM, then $sec^S(n)$ is 1 or 2 respectively.
- $cloud^S(n)$ - Defines in which type of cloud the substrate node *n* is located. This attribute can take any positive value (including zero). In the example, if *n* is located in a public cloud, then $cloud^S(n)$ = 0. If it is located in a trusted public cloud, then $cloud^S(n)$ = 1. Lastly, if *n* is located at a private cloud, then $cloud^S(n)$ = 2.

A_E^S contains the following attributes for substrate links:

$$A_E^S = \{\{bw^S(l), sec^S(l)\} | l \epsilon E^S\}$$

- $bw^S(l)$ - Total bandwidth capacity of the substrate link *l*. This attribute can take any values greater or equal to 0;
- $sec^S(l)$ - Security provided by the substrate link *l*. This attribute can take any positive value (including zero). In the example, $sec^S(l) = 0$ if substrate link l might be configured with no security mechanisms (defined as default). If link *l* supports protocols that provide authenticity and integrity guarantees, then $sec^S(l) = 1$. Lastly, if *l* additionally offers confidentiality assurances (by encrypting packets), then $sec^S(l) = 2$.

2) Virtual Network Requests

VNRs are defined by the clients of the system. Similar to the substrate network, the VNRs are also modeled as weighted undirected graphs. Each virtual network request is denoted by $G^V = (N^V, E^V, Time^V, Dur^V, A_N^V, A_E^V)$, where N^V is the set of virtual nodes, E^V is the set of virtual links, $Time^V$ is the arrival time of the VNR, Dur^V is the time period for which the VN is valid, A_N^V is the set of attributes of substrate nodes, and A_E^V is the set of attributes of substrate links. A_N^V contains the attributes demanded by virtual nodes and links, quite similar to the Substrate Network ones, except for:

- $avail^V(n)$ - Defines where the backup of virtual node *n* should be mapped. This attribute can take any positive value (including zero). Considering the same example, if replication is not needed for the VNR , then $avail^V(n)$ = 0. If virtual node *n* should have a backup in the same cloud, then $avail^V(n)$ = 1. Finally, if *n* should have a backup located in other cloud (e.g., in order to survive from a cloud outage), then $avail^V(n)$ = 2.

3) Measurement of Substrate Network Resources

The residual capacity (or available capacity) of a substrate node, $R_N(n^S)$, is defined as the available CPU capacity of the substrate node $n^S \epsilon N^S$.

$$R_N(n^S) = cpu^S(n^S) - \sum_{\forall n^V \uparrow n^S} cpu^V(n^V),$$

where $n^V \epsilon N^V$ and $x \uparrow y$ denotes that the virtual node *x* is hosted on the substrate node *y*.

Similarly, the residual capacity of a substrate link, $R_E(e^S)$, is defined as the total amount of bandwidth available on the substrate link $e^S \epsilon E^S$.

$$R_E(e^S) = bw^S(e^S) - \sum_{\forall e^V \uparrow e^S} bw^V(e^V),$$

where $e^V \epsilon E^V$ and $x \uparrow y$ denotes that the flow of the virtual link *x* traverses the substrate link *y*.

Since a virtual link can be mapped onto multiple substrate links, i.e., mapped onto a substrate path, it is also important to define the available bandwidth capacity of a substrate path *P*. This corresponds to the minimum available bandwidth among all the substrate links belonging to *P*.

$$R_E(P) = \min_{e^S \epsilon P} R_E(e^S)$$

4) Objectives

One of the main goals of VNE is to maximize the profit of the virtualization provider. For this purpose, and similar to [4, 17], the revenue of accepting a VNR is proportional to the acquired resources. However, in our case, we have to take into consideration that stronger security protection measures maybe are charged at a higher premium value:

$$\mathbb{R}(G^V) = \lambda_1 \sum_{e^V \epsilon E^V} bw^V(e^V)\ sec^V(e^V) + \lambda_2 \sum_{n^V \epsilon N^V} cpu^V(n^V)\ sec^V(n^V)\ cloud^V(n^V),$$

where λ_1 and λ_2 are weight coefficients that denote the relative proportion of each revenue component to the total revenue.

Although the revenue gives us an idea of how much a virtualization provider will gain by accepting a certain VNR, it is also important to understand the cost the provider will incur for embedding that request. The cost of embedding a VNR is proportional to the total sum of substrate resources

allocated to that VN. In particular, this cost has to take into consideration that virtual links may be embedded to one or more physical links. In our work, the cost may also increase if the VNR requires higher security for its virtual nodes and links. We define the cost of embedding a VNR as:

$$\mathbb{C}(G^V) = \lambda_1 \sum_{e^V \epsilon E^V} \sum_{e^S \epsilon E^S} f_{e^S}^{e^V} \; sec^S(e^S) + \lambda_2 \sum_{n^V \epsilon N^V} \sum_{n^S \epsilon N^S} cpu_{n^S}^{n^V} \; sec^S(n^S) \; cloud^S(n^S),$$

where $f_{e^S}^{e^V}$ denotes the total amount of bandwidth allocated on the substrate link e^S for virtual link e^V. Similarly, $cpu_{n^S}^{n^V}$ corresponds to the total amount of CPU allocated on the substrate node n^S for virtual node n^V (either working or backup parts). As above, λ_1 and λ_2 are the same weights, which denote the relative proportion of each cost component to the total cost.

III. MILP Formulation

We have developed a MILP formulation to solve the SecVNE problem. This section starts by explaining the decision variables used in our formulation, the objective function, and finally the constraints that were defined to model the problem.

A. Decision variables

Table I explains the variables that are used in our MILP formulation. Briefly, $wf_{u,v}^{i,j}$, $bf_{u,v}^{i,j}$, $wl_{u,v}^{i,j}$, $bl_{u,v}^{i,j}$ and $rl_{u,v}$ are related to working and backup links; $wn_{i,v}$, $bn_{i,v}$ and rn_v are associated with the working and backup nodes; $wc_{i,c}$ and $bc_{i,c}$ are related to the embedding location of virtual nodes.

B. Objective Function

The objective function of our formulation has three goals: to minimize 1) the sum of all computing costs, 2) the sum of all communication costs, and 3) the overall number of hops of the substrate paths for the virtual links.

Since we have different objectives, and these objectives are measured in different units, we have to unify them. Thus, in our formulation we consider a weighted-sum function with three different coefficients, β_1, β_2, and β_3, which should be reasonably parameterized for each objective.

The first and the second parts of Eq. 1 are the sum of all working and backup link bandwidth costs, respectively. The third and the fourth parts are the sum of all working and backup computing node costs. In this function, the level of security provided by the physical resources is considered. The parameter α is a weight for each physical link that assumes some value defined previously and that depends of (*u*,*v*) being an inter-cloud connection (link between two clouds) or an intra-domain link (link inside a cloud). This is due to the expectation that virtual links that use links connecting two clouds (inter-domain links) will have a higher cost (monetary, delay, or other). Also, mapping a VN onto substrate resources that provide higher security are expected to increase costs. The fifth and last parts of the objective function achieve the third goal presented above.

Symbol	Meaning
$wf_{u,v}^{i,j} \geqslant 0$	The amount of working flow, i.e., bandwidth, on the physical link (*u*,*v*) for the virtual link (*i*,*j*)
$bf_{u,v}^{i,j} \geqslant 0$	The amount of backup flow, i.e., backup bandwidth, on the physical link (*u*,*v*) for the virtual link (*i*,*j*)
$wl_{u,v}^{i,j} \in \{0,1\}$	Denotes whether the virtual link (*i*,*j*) is mapped onto the physical link (*u*,*v*). (1 if (*i*,*j*) is mapped on (*u*,*v*), 0 otherwise)
$bl_{u,v}^{i,j} \in \{0,1\}$	Denotes whether the backup of virtual link (*i*,*j*) is mapped onto the physical link (*u*,*v*). (1 if backup of (*i*,*j*) is mapped on (*u*,*v*), 0 otherwise)
$rl_{u,v} \geqslant 0$	The reserved backup resources on a physical link (*u*,*v*), i.e., the total quantity of bandwidth that is allocated for backup flows.
$wn_{i,v} \in \{0,1\}$	Denotes whether virtual node *i* is mapped onto the physical node *v*. (1 if *i* is mapped on *v*, 0 otherwise)
$bn_{i,v} \in \{0,1\}$	Denotes whether virtual node *i*'s backup is mapped onto the physical node *v*. (1 if *i*'s backup is mapped on *v*, 0 otherwise)
$rn_v \geqslant 0$	The reserved backup resource on a physical node *v*, i.e., the total quantity of CPU that is allocated to backups.
$wc_{i,c} \in \{0,1\}$	Denotes whether virtual node *i* is mapped on cloud *c*. (1 if *i* is mapped on *c*, 0 otherwise)
$bc_{i,c} \in \{0,1\}$	Denotes whether virtual node *i*'s backup is mapped on cloud *c*. (1 if *i*'s backup is mapped on *c*, 0 otherwise)

Table I: List of all the Domain Constraints (decision variables) used in our MILP formulation.

Intuitively, with our formulation when a VNR arrives to our system, the embedder will try to match the request to the resources in such a way that more "powerful" resources are saved (e.g., those with higher security levels) for the virtual resources that require them explicitly. For instance, virtual nodes with $sec^V = 1$ will be mapped onto substrate nodes with $sec^S = 2$ if and only if there are no other substrate nodes with $sec^S = 1$ available.

$$\begin{aligned} min \quad & \beta_1 \sum_{(i,j)\epsilon E^V} \sum_{u,v \epsilon N^S} \alpha_{u,v} \;\; wf_{u,v}^{i,j} \;\; sec^S(u,v) \\ & + \beta_1 \sum_{u,v \epsilon N^S} rl_{u,v} \;\; sec^S(u,v) \\ & + \beta_2 \sum_{i \epsilon N^V} \sum_{v \epsilon N^S} cpu^V(v) \; wn_{i,v} \; sec^S(v) \; cloud^S(v) \\ & + \beta_2 \sum_{v \epsilon N^S} rn_v \;\; sec^S(v) \;\; cloud^S(v) \\ & + \beta_3 \sum_{(i,j)\epsilon E^V} \sum_{u,v \epsilon N^S} wl_{u,v}^{i,j} \\ & + \beta_3 \sum_{(i,j)\epsilon E^V} \sum_{u,v \epsilon N^S} bl_{u,v}^{i,j} \end{aligned} \tag{1}$$

C. Typical Constraints

Besides the Domain Constraints of Table I, in this section we define the other constraints that often have to be considered in most VNE MILP formulations.

Link Mapping for Working Traffic

Constraints 2, 3 and 4 refer to the working flow conservation conditions, which denote that the overall network flow to a node is zero, except for the source node and the sink node, respectively (i.e., no flow appears or disappears in any node, unless it is a source or a sink node).

$$\sum_{u\epsilon N^S \cup N^V} wf_{u,v}^{i,j} - \sum_{u\epsilon N^S \cup N^V} wf_{v,u}^{i,j} = 0, \forall (i,j)\epsilon E^V, v\epsilon N^S \backslash \{s_i, t_i\} \quad (2)$$

$$\sum_{v\epsilon N^S} wf_{i,v}^{i,j} - \sum_{v\epsilon N^S} wf_{v,i}^{i,j} = bw^V(i,j), \forall (i,j)\epsilon E^V \quad (3)$$

$$\sum_{v\epsilon N^S} wf_{j,v}^{i,j} - \sum_{v\epsilon N^S} wf_{v,j}^{i,j} = -bw^V(i,j), \forall (i,j)\epsilon E^V \quad (4)$$

Eq. 5 and 6 guarantee that the working flow of a virtual link (i, j) always departs from the correspondent working node of *i* and arrives to the correspondent working node of *j*.

$$wn_{i,v} \quad bw^V(i,j) = wf_{i,v}^{i,j}, \forall v\epsilon N^S, (i,j)\epsilon E^V \quad (5)$$

$$wn_{j,v} \quad bw^V(i,j) = wf_{v,j}^{i,j}, \forall v\epsilon N^S, (i,j)\epsilon E^V \quad (6)$$

Node Capacity Constraints

Substrate nodes can map nodes from different VNRs. For instance, they can be the working node corresponding to a virtual node i from a VNR x and simultaneously be the corresponding backup node of a virtual node j from a VNR y. Considering this, for a substrate node, the total allocated capacity depends on the total capacity that is allocated for working nodes, plus the total capacity that is allocated for backup nodes, which should be less than the current capacity of the substrate node. This is represented by Eq. 7 and 8.

$$\sum_{u\epsilon N^V} bn_{u,v} \quad cpu^V(u) \leqslant rn_v, \forall v\epsilon N^S \quad (7)$$

$$\sum_{u\epsilon N^V} wn_{u,v} \quad cpu^V(u) + rn_v \leqslant R_N(v), \forall v\epsilon N^S \quad (8)$$

Link Capacity Constraints

Like substrate nodes, substrate links can also map virtual links from different VNRs. Eq. 9 and 10 define the allocated link capacity of a substrate link as the sum of the capacity allocated for the active flows and the reserved resources for backup. The allocated capacity of a substrate link should be less than the residual capacity of that physical link.

$$\sum_{(i,j)\epsilon E^V} (bf_{u,v}^{i,j} + bf_{v,u}^{i,j}) \leqslant rl_{u,v}, \forall u, v\epsilon N^S \quad (9)$$

$$\sum_{(i,j)\epsilon E^V} (wf_{u,v}^{i,j} + wf_{v,u}^{i,j}) + rl_{u,v} \leqslant R_E(u,v), \forall u, v\epsilon N^S \quad (10)$$

D. Security Constraints

Our VNE formulation includes security constraints, for both the nodes, links, and clouds.

Node Security Constraints

The security restrictions for nodes are defined as:

$$wn_{u,v} \quad sec^V(u) \leqslant sec^S(v), \forall u\epsilon N^V, v\epsilon N^S \quad (11)$$

$$bn_{u,v} \quad sec^V(u) \leqslant sec^S(v), \forall u\epsilon N^V, v\epsilon N^S \quad (12)$$

Eq. 11 guarantees that a virtual node u is only mapped to a physical node that has an equal or higher security level than u's security demand. Eq. 12 ensure the same as the previous one, but for backup nodes. This ensures, returning to our initial example, that a secure VM from the physical infrastructure can map virtual nodes requesting normal containers, VMs, or secure VMs, whereas a physical container can only map virtual nodes that are looking for normal containers.

Link Security Constraints

The following equations refer to the working and the backup link security constraints:

$$wl_{u,v}^{i,j} \quad sec^V(i,j) \leqslant sec^S(u,v), \forall (i,j)\epsilon E^V, u, v\epsilon N^S \quad (13)$$

$$bl_{u,v}^{i,j} \quad sec^V(i,j) \leqslant sec^S(u,v), \forall (i,j)\epsilon E^V, u, v\epsilon N^S \quad (14)$$

Here, it is necessary to ensure that each virtual link is mapped to one or more physical links that provide a security level equal or higher than the security demand of the virtual link. Similarly to the previous case, a physical link that provides default security can only map virtual links that demand for that low level of security, while a physical link that provides authenticity, integrity and confidentiality guarantees can map virtual links with any security demand.

Cloud Security Constraints

Eq. 15 ensures that a virtual node u is mapped to a certain physical node v only if the cloud where v is located is of a type of equal or higher security than the type of cloud demanded by node u. For instance, a virtual node that requires to be mapped on a public cloud may be mapped to either a public, a trusted public or a private cloud, considering again our example. On the other side, a virtual node that requires the highest level of security can only be mapped to nodes located in a private cloud. Eq. 16 guarantees the same for the backup nodes.

$$wn_{u,v} \quad cloud^V(u) \leqslant cloud^S(v), \forall u\epsilon N^V, v\epsilon N^S \quad (15)$$

$$bn_{u,v} \quad cloud^V(u) \leqslant cloud^S(v), \forall u\epsilon N^V, v\epsilon N^S \quad (16)$$

E. Availability Constraints

Finally, we define the constraints related to fault-tolerance and availability.

Link Mapping for Backup Traffic

For the backup traffic, it is necessary to define the same set of flow constraints defined for working traffic, but using the variables $bf_{u,v}^{i,j}$ and $bn_{u,v}$. Constraints 17, 18 and 19 refer to the backup flow conservation conditions, which denote that the network flow to a node is zero, except for the source node and the sink node, respectively. $wantBackup$ is a parameter defined by the tenant and it assumes the value 1 if backups are needed or the value 0 otherwise.

$$\sum_{u\epsilon N^S \cup N^V} bf_{u,v}^{i,j} - \sum_{u\epsilon N^S \cup N^V} bf_{v,u}^{i,j} = 0, \forall (i,j)\epsilon E^V, v\epsilon N^S \quad (17)$$

$$\sum_{v\epsilon N^S} bf_{i,v}^{i,j} - \sum_{v\epsilon N^S} bf_{v,i}^{i,j} = bw^V(i,j) * wantBackup, \forall (i,j)\epsilon E^V \quad (18)$$

$$\sum_{v\epsilon N^S} bf_{j,v}^{i,j} - \sum_{v\epsilon N^S} bf_{v,j}^{i,j} = -bw^V(i,j) * wantBackup, \forall (i,j)\epsilon E^V \quad (19)$$

Eq. 20 and 21 guarantee that the backup flow of a virtual link (i, j) always departs from the corresponding backup node of *i* and arrives to the corresponding backup node of *j*. Normally, the backup path only carries information of a virtual link if a failure in the working substrate path has occurred.

$$bn_{i,v}\ bw^V(i,j) = bf^{i,j}_{i,v} * wantBackup, \forall v \epsilon N^S, (i,j) \epsilon E^V \tag{20}$$

$$bn_{j,v}\ bw^V(i,j) = bf^{i,j}_{v,j} * wantBackup, \forall v \epsilon N^S, (i,j) \epsilon E^V \tag{21}$$

The equation below guarantees that the meta-links only carry working or backup traffic to their correspondent virtual nodes. This means that, if a virtual node 1 needs to send information to virtual node 2, the data does not need to pass through the meta-links of a virtual node 3.

$$\sum_{j,k!=i \ \ j,k \epsilon N^V} wf^{j,k}_{i,v} + wf^{j,k}_{v,i} + bf^{j,k}_{i,v} + bf^{j,k}_{v,i} = 0, \forall v \epsilon N^S, i \epsilon N^V \tag{22}$$

Virtual Node Mapping

Eq. 23 and 24 state that each virtual node has to be mapped to exactly one working node and $wantBackup$ backup nodes in the substrate node, i.e., if $wantBackup = 0$, virtual nodes will not have backup, if $wantBackup = 1$, virtual nodes will have backup. For the same VN, Eq. 25 guarantees that *(i)* two different virtual working nodes are not mapped to the same substrate node; and *(ii)* a substrate node that is the backup for a virtual node does not have any virtual working nodes on it. Eq. 26 guarantees that a substrate node can be the backup of a single virtual node, for the same VN.

$$\sum_{v \epsilon N^S} wn_{u,v} = 1, \forall u \epsilon N^V \tag{23}$$

$$\sum_{v \epsilon N^S} bn_{u,v} = wantBackup, \forall u \epsilon N^V \tag{24}$$

$$\sum_{u \epsilon N^V} wn_{u,v} + bn_{z,v} \leqslant 1, \forall v \epsilon N^S, z \epsilon N^V \tag{25}$$

$$\sum_{u \epsilon N^V \setminus \{z\}} bn_{u,v} + bn_{z,v} \leqslant 1, \forall v \epsilon N^S, z \epsilon N^V \tag{26}$$

Note that we define Eq. 25 because we want to minimize the number of virtual resources of a VN affected if a failure occurs in a certain substrate node. Figures 3 clarifies this idea. In this example if a failure occurs in the physical node A, and nodes 1 and 2 are mapped onto it, all the virtual links will be affected. Instead, if all the nodes of the VN are mapped onto different physical nodes and the failure occurs in node A, only the virtual link (1,3) will be affected.

Since in our work we allow the user to choose between having no replication, replication in the same cloud or in different clouds, it is necessary to specify these restrictions.

$$\sum_{c \epsilon C} wc_{u,c} = 1, \forall u \epsilon N^V \tag{27}$$

$$\sum_{c \epsilon C} bc_{u,c} = wantBackup, \forall u \epsilon N^V \tag{28}$$

$$wantBackup * wc_{u,c} + bc_{u,c} \leqslant dep^V(u), \forall u \epsilon N^V, c \epsilon C \tag{29}$$

$$wc_{u,c} \geqslant bc_{u,c} * dep^V(u) - 1, \forall u \epsilon N^V, c \epsilon C \tag{30}$$

Eq. 27 states that each virtual node is mapped on exactly one cloud. Eq. 28 ensures that, when a VN needs backup ($wantBackup = 1$), the backup of each virtual node is mapped to exactly one cloud. Eq. 29 and 30 are restrictions that address if a virtual node *u* and its correspondent backup will be on the same cloud or in different clouds, depending on the availability level required by *u* ($dep^V(u)$).

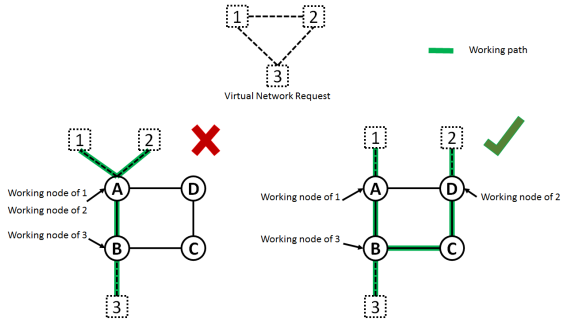


Figure 3: Example of an embedding that respects the first part of eq. 25.

Eq. 31 and 32 establish a relation between the virtual and physical nodes and the clouds (the first one related to working nodes, and the second related with the backup nodes).

$$\sum_{v \epsilon N^S} (wn_{u,v} * doesItBelong_{c,v}) \geqslant wc_{u,c}, \forall u \epsilon N^V, c \epsilon C \tag{31}$$

$$\sum_{v \epsilon N_S} (bn_{u,v} * doesItBelong_{c,v}) \geqslant bc_{u,c}, \forall u \epsilon N_V, c \epsilon C \tag{32}$$

The aim of these equations is the following. If a virtual node *u* is mapped onto a physical node *v* and *v* belongs to cloud *c* ($doesItBelong_{c,v} = 1$ if substrate node v belongs to cloud c, 0 otherwise), then *u* is mapped on cloud *c*.

In a similar fashion, we also need restrictions to create relationships between the variables *wf*, *wl* and wn, and *bf*, *bl* and bn:

$$wn_{i,v} * bw^V(i,j) \geqslant wl^{i,j}_{i,v}, \forall (i,j) \epsilon E^V, v \epsilon N^S \tag{33}$$

$$wn_{j,v} * bw^V(i,j) \geqslant wl^{i,j}_{v,j}, \forall (i,j) \epsilon E^V, v \epsilon N^S \tag{34}$$

$$bw^V(i,j) * wl^{i,j}_{u,v} \geqslant wf^{i,j}_{u,v}, \forall (i,j) \epsilon E^V, u, v \epsilon N^S \cup N^V \tag{35}$$

Eq. 33 and 34 are constraints that ensure that if a meta-link is established between a virtual node i and a physical node v, then it means that i is mapped onto v. For instance, if $wl^{i,j}_{i,v} = 1$, then $wn_{i,v} = 1$. Eq. 35 ensures that if there is a flow between nodes u and v for a virtual link (i,j), then this means that (i,j) is mapped to a meta-link or a physical link whose end-points are u and v. For example, if $wf^{i,j}_{u,v} = 1$, then $wl^{i,j}_{u,v} = 1$.

Eq. 36, 37 and 38 achieve the same goals as before, but for the backup:

$$bn_{i,v} * bw^V(i,j) \geqslant bl^{i,j}_{i,v}, \forall (i,j) \epsilon E^V, v \epsilon N^S \tag{36}$$

$$bn_{j,v} * bw^V(i,j) \geqslant bl^{i,j}_{v,j}, \forall (i,j) \epsilon E^V, v \epsilon N^S \tag{37}$$

$$bw^V(i,j) * bl^{i,j}_{u,v} \geqslant bf^{i,j}_{u,v}, \forall (i,j) \epsilon E^V, u, v \epsilon N^S \cup N^V \tag{38}$$

Finally, we include two binary constraints to guarantee the symmetric property of the binary variables related with links.

$$wl_{u,v}^{i,j} = wl_{v,u}^{i,j}, \forall (i,j) \epsilon E^V, u, v \epsilon N^S \cup N^V \quad (39)$$

$$bl_{u,v}^{i,j} = bl_{v,u}^{i,j}, \forall (i,j) \epsilon E^V, u, v \epsilon N^S \cup N^V \quad (40)$$

Nodes and Links Disjointness

Since any substrate nodes and links of a working path can fail, we have to ensure that backup paths connecting the backups of the virtual nodes are disjoint from the substrate resources that are being used for the working part (otherwise a backup path can be compromised if a physical resource belonging to the working and backup part fails).

$$BigConstant * working_u \geqslant \sum_{v \epsilon N^S} wl_{u,v}^{i,j}, \forall (i,j) \epsilon E^V, u \epsilon N^S \quad (41)$$

$$BigConstant * backup_u \geqslant \sum_{v \epsilon N^S} bl_{u,v}^{i,j}, \forall (i,j) \epsilon E^V, u \epsilon N^S \quad (42)$$

$$backup_u = 1 - working_u, \forall u \epsilon N^S \quad (43)$$

Equations 41 to 43, together with Eq. 25, ensure path disjointness between the working and the backup parts. As we already observed, Eq. 25 ensures that a substrate node mapping the working virtual node can not be a backup of any other node, and vice-versa. Relatively to Equations 41 to 43 we ensure that if a substrate node u is an end point of a certain link that is being used as a working resource, u can not be an end point of a link that is being used as a backup resource, and vice-versa. Variables $working$ and $backup$ are auxiliary variables that define if a certain physical node u belongs to the working or backup part. $BigConstant$ is a constant big enough to ensure that the restriction is valid when its rightmost part is greater than 0.

IV. Evaluation

This chapter presents performance results of our solution. We have implemented a simulator to reproduce an environment where VNRs with different requirements arrive over time. Section IV-A presents the simulation setup. In Section IV-B we describe the different experiments and also briefly introduce the algorithm that was the basis for comparison. Finally, in Section IV-C we present the evaluation results and discuss them.

A. Simulation Setup

We have implemented an event simulator to evaluate the performance of our algorithm. Our tool was based on the implementation presented in [1, 4] and, in short, it simulates the dynamic arrival of VNRs to the system.

For the evaluation, the SN topology was randomly generated with 25 nodes using the GT-ITM tool [18] in (10x10) grids. Each pair of substrate nodes was randomly connected with probability between 0.1 and 0.3. The CPU and bandwidth resources of the substrate nodes and links were real numbers uniformly distributed between 50 and 100. We assumed that VNRs arrivals ($Time^V$) are modeled as a Poisson process with an average rate of 4 VNRs per 100 time units, each one having an exponentially distributed lifetime (Dur^V) with an average of $\mu = 1000$ time units. In each VNR, the number of virtual nodes was randomly determined by a uniform distribution between 2 and 4. Each pair of virtual nodes was randomly connected with probability between 0.1 and 0.3. The CPU and bandwidth capacity requirements of the virtual nodes and links, respectively, were real numbers uniformly distributed between 10 and 20. We chose to only address a small scale environment (25 nodes to the SN and 2-4 nodes to the VNs) because optimal solutions, such as the MILP used in SecVNE, do not scale for large networks. One way to deal with this problem is to devise heuristics, something we leave for future work.

Notation	Algorithm description
D-ViNE	VNE MILP model presented in [4]
NoSec	Our SecVNE with no security requirements for VN and SN
SecL+0	SecVNE with all VNRs having security requirements (excluding availability) for $1/3$ of their resources (nodes and links)
SecH+0	Similar to SecL+0, but with security requirements (excluding availability) for $2/3$ of the resources
SecL+5	SecVNE with all VNRs having security requirements (excluding availability) for $1/3$ of their resources. In addition, 5% of the requests require one replica of each resource for increased availability
SecH+5	Similar to SecL+5, but with security requirements (excluding availability) for $2/3$ of the resources
SecL+10	Similar to SecL+5, but 10% of the requests require one replica
SecH+10	Similar to SecL+10, but with security requirements (excluding availability) for $2/3$ of the resources
SecL+20	Similar to SecL+5, but 20% of the requests require one replica
SecH+20	Similar to SecL+20, but with security requirements (excluding availability) for $2/3$ of the resources

Table II: Configurations evaluated

For SecVNE, it is necessary to include security requirements. In the simulation, the substrate nodes were divided between three clouds, each one with a different security level (public, trusted public and private). The parameters sec and $cloud$ were set in increasing order of security with the values $\{1.0, 1.1, 1.2\}$. The rationale for these values was to find a good adjustment of the different security levels to their price. For example, a higher level of security is considered 20% more expensive than having no security. The choice of these specific values was based on an empirical analysis we performed on the pricing schemes of Amazon EC2. The probability of substrate nodes and links having $sec^S = 1.0$ was 0.05, $sec^S = 1.1$ was 0.4, and $sec^S = 1.2$ was 0.55. The weight (α) of all substrate links was 1. The $cloud^V$ of the virtual nodes was distributed uniformly by the three different cloud types.

For the variables γ_1, γ_2, γ_3 to be parameterized appropriately, it is necessary to consider:

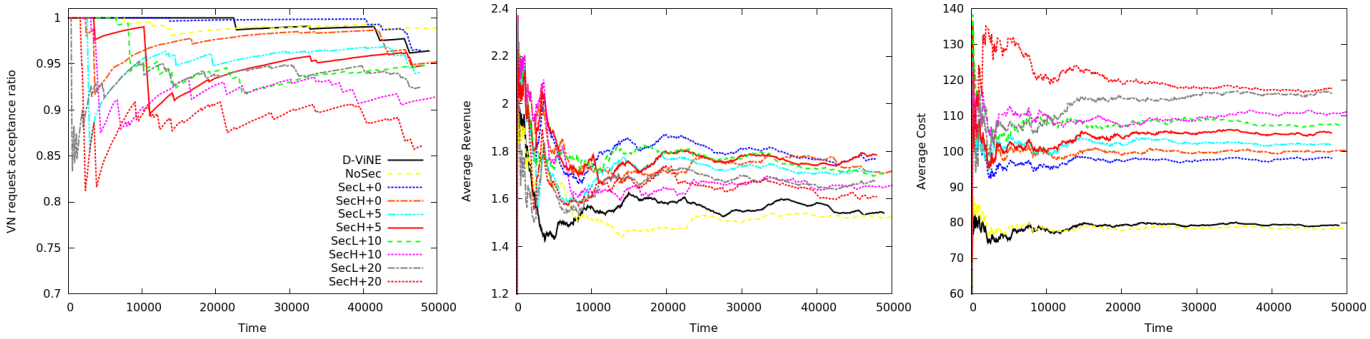
$$\gamma_1 \in]0,1[,$$
$$\gamma_2 \in]0,1[,$$
$$\gamma_3 \in]0,1[,$$
$$\gamma_1 + \gamma_2 + \gamma_3 = 1 \text{ [14]}$$

As such, in our evaluation we considered that $\gamma_1 = 1/3$, $\gamma_2 = 1/3$ and $\gamma_3 = 1/3$ to find one supported Pareto solution.

To solve the MILP, we used the open source library GLPK [9]. The simulation ran for 50000 time units, and during this period the MILPs tried to embed 1000 VNRs. The order of arrival of VNRs and the capacity requirements of each VNR were the same for both algorithms, ensuring that both solved equivalent problem instances.

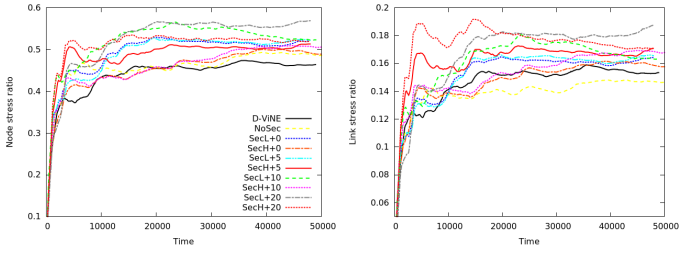
B. Evaluation Method

In our evaluation, we compared the algorithm D-ViNE [4] with ours, SecVNE. D-ViNE was chosen because it has been considered as the baseline for most VNE work and due to its availability as open-source software. D-ViNE requirements



(a) (b) (c)

Figure 4: Performance results: (a) VNR acceptance ratio over time; (b) Time average of generated revenue; (c) Average cost of accepting VNRs over time.



(a) (b)

Figure 5: Performance results: (a) Average node utilization; (b) Average link utilization.

are only based on CPU and bandwidth capacities, while our algorithm adds to these requirements also security demands, including availability needs, and cloud preferences. Furthermore, the comparison of SecVNE against D-ViNE is interesting in order to understand what are the implications of introducing security constraints on a traditional embedding problem. The objective function of D-ViNE is presented in Eq. 44:

$$min \sum_{uv \epsilon E^S} \frac{\alpha_{uv}}{R_E(u,v)+\delta} \sum_i f_{uv}^i + \sum_{w \epsilon N^S} \frac{\beta_w}{R_N(w)+\delta} \sum_{m \epsilon N^{S'} \setminus N^S} x_{mw} c(m) \quad (44)$$

Besides the minimization of CPU and bandwidth resources allocated to a VN (i.e., minimization of costs), the objective function also tries to balance the load by introducing weights.

For evaluation, we run nine different setups of our algorithm, reflecting alternative needs of the tenants. Table II summarizes the various setups and includes the notations used to refer to the experiments.

C. Evaluation Results

We used several performance metrics (the same as in [4]) for evaluation in our experiments. Namely, we have considered:

- VNR acceptance ratio: the percentage of requests accepted over time;
- Average revenue: the revenue that the infrastructure provider obtains over time;
- Average cost of accepting a VNR: the cost the provider will incur by embedding a request;
- Average node utilization: the load of the SN nodes over time;
- Average link utilization: the load of SN links over time.

To calculate the revenue $\mathbb{R}$ and costs $\mathbb{C}$ we used the equations presented in Section II-A4. We now present all results from the simulations and discuss each outcome.

1) **The embedding performance of SecVNE without security and availability requirements is similar to the embedding performance of D-ViNE.** When security and availability are not taken into account, our algorithm performs similar to D-ViNE, as can be seen in all figures and it shows our baseline to be identical to the most commonly used VNE algorithm.
2) **A richer set of features (namely, security and availability) decreases the acceptance ratio.** Figure 4a shows that D-ViNE leads to a higher acceptance ratio over time when compared with our SecVNE *when the security requirements increase over a certain threshold*. This was expected, as SecVNE is richer in terms of the features provided (security and availability, in addition to CPU and bandwidth). Consequently, in SecVNE to accept a VNR the number of constraints is higher and more conditions need to be satisfied. Interestingly, the reduction in acceptance ratio is more pronounced when the number of VNRs requiring availability increases (compared with the other security features). This is due to the higher use of substrate resources, as more resources need to be allocated to these VNRs.
3) **A richer set of features (security and availability) increases the revenue until the point when the acceptance ratio becomes too low.** Figure 4b illustrates the time average of generated revenue. When our algorithm is set without security and availability requirements (NoSec), it generates nearly the same revenue as D-ViNE. In this figure it is also possible to observe that all other configurations generate more revenue than D-ViNE. This is due to security having a weight coefficient (i.e., a price) that is considered in the revenue function (richer resources are expected to be more more expensive). Since the acceptance ratio is not far from those of D-ViNE, the revenue increases. It is to expect, therefore, a turning point where infrastructure providers may need to increase the price of their resources to generate a profit, as a natural consequence of the balance between demand and supply.

4) **Security requirements increase the costs of embedding a VNR.** Figure 4c shows that the costs of embedding a VNR with the SecVNE algorithm are higher when compared to D-ViNE, as providing security has a cost. It is clear the cost of availability to be higher than that of the other secure properties because when backups are required by a VNR, the CPU and bandwidth resources (with the security constraints) demanded by a virtual node are always allocated twice: one allocation for the working node and another to the backup node.
5) **Security requirements increase substrate resources utilization.** Figure 5a and Figure 5b show the average substrate node and link utilization, respectively. We observe in both figures that more resources are allocated in the SN with SecVNE than with D-ViNE. Between all security properties providing availability remains costlier, leading to higher average utilization of both nodes and links. The reasons are the same as in the previous point.

V. Related Work

There is already a wide literature on this problem [6]. Yu et al. [17] where the first ot solve it efficiently, by assuming the capability of path splitting (multi-path) in the substrate network, which enable the computationally harder part of the problem to be solved as a multicommodity flow (MCF), for which efficient algorithms exist. The authors solve the problem considering two independent phases – an approach commonly used by most algorithms. In the first phase, a greedy algorithm is used for virtual node embedding. Then, to map the virtual links, either efficient MCF solutions or k-shortest path algorithms can be used. In [4], Chowdhury et al. proposed two algorithms for VNE that introduce coordination between the node and link mapping phases. The main technique proposed in this work is to augment the substrate graph with meta-nodes and meta-links that allow the two phases to be well correlated, achieving more efficient solutions. Neither of these works considers security.

As failures in networks are inevitable, the issue of failure recovery and survavibility in VNE has gained attention recently. H. Yu et al. [16] have focused on the failure recovery of nodes. They proposed to extend the basic VNE mapping with the inclusion of redundant nodes. Rahman et al. [13] formulated the survivable virtual network embedding (SVNE) problem to incorporate single substrate link failures. Contrary to our work, these proposals target only availability.

A mostly unexplored perspective on the VNE problem is providing security guarantees. Fischer et al. [5] have introduced this problem with a position paper where was proposed the assignment of security levels in the physical resources and virtual network requests. No algorithms were presented. Liu et al. [12] have afterwards proposed a VNE algorithm based on this idea. Their simple model does not support the detailed specification of security we propose, and does not consider availability nor a multi-cloud setting with different trust domains.

VI. Conclusions

This paper proposes a VNE solution that addresses a diverse set of security requirements, applied both to communications and virtual nodes. These requirements enable performance trade-offs to be explored (e.g., by avoiding packet encryption) and allow trusted executions assisted by hardware (e.g., Intel SGX extensions). In addition, the model was extended by considering multiple clouds with distinct levels of trust, either private (possibly belonging to the tenant) or public. By not relying on a single cloud provider we avoid internet-scale single points of failures, taking care of outages by replicating workloads over various clouds. Privacy issues can also be accommodated by constraining the mapping of particular virtual nodes to specific classes of clouds (e.g., private).

The results from our experiments show that there is a cost in providing security assurances. However, a relatively small increase in the price of the new features (security resources, for instance), coupled with efficient techniques to reduce the embedding cost (e.g., pooling of backup resources), enables the offering of security with an increased profit.

References

[1] *ViNE-Yard*. http://www.mosharaf.com/ViNE-Yard.tar.gz.

[2] Al-Shabibi, Ali, *et al.* . 2014. OpenVirteX: Make Your Virtual SDNs Programmable. *In: HotSDN*.

[3] Alaluna, Max, *et al.* . 2016. (Literally) above the clouds: virtualizing the network over multiple clouds. *In: IEEE NetSoft*.

[4] Chowdhury, Mosharaf, *et al.* . 2012. ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping. *IEEE/ACM Transactions on Networking*, Feb.

[5] Fischer, Andreas, *et al.* . 2011. Position paper: Secure virtual network embedding. *Praxis der Informationsverarbeitung und Kommunikation*.

[6] Fischer, Andreas, *et al.* . 2013. Virtual Network Embedding: A Survey. *IEEE Communications Surveys Tutorials*, **15**(4).

[7] Froehlich, A. 2015 (July). *9 Spectacular Cloud Computing Fails*. Information Week.

[8] Gill, Phillipa, *et al.* . 2011. Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications. *ACM SIGCOMM*.

[9] GLPK. 2008. *GNU Linear Programming Kit*. http://www.gnu.org/software/glpk/.

[10] Koponen, Teemu, *et al.* . 2014. Network Virtualization in Multi-tenant Datacenters. *In: USENIX NSDI*.

[11] Lacoste, M., *et al.* . 2016. User-Centric Security and Dependability in the Clouds-of-Clouds. *IEEE Cloud Computing*, **3**(5).

[12] Liu, Shuhao, *et al.* . 2014. Security-aware virtual network embedding. *In: IEEE ICC'14*.

[13] Rahman, Muntasir Raihan, *et al.* . 2010. Survivable virtual network embedding. *In: International Conference on Research in Networking*.

[14] Steuer, Raphael E. 1986. *Multiple Criteria Optimization: Theory, Computation and Application*. John Wiley, New York, 546 pp.

[15] Tsidulko, J. 2016 (July). *The 10 Biggest Cloud Outages Of 2016 (So Far)*. The Channel Company.

[16] Yu, H., *et al.* . 2011. Cost Efficient Design of Survivable Virtual Infrastructure to Recover from Facility Node Failures. *In: IEEE ICC'11*.

[17] Yu, Minlan, *et al.* . 2008. Rethinking virtual network em-

bedding: substrate support for path splitting and migration. *ACM SIGCOMM*.

[18] Zegura, Ellen W., *et al.* . 1996. How to model an internetwork. *In: IEEE INFOCOM*.