Secure Virtual Network Embedding in a Multi-Cloud Environment

Max Alahuna\textsuperscript{a}, Luís Ferrolho\textsuperscript{a}, José Rui Figueira\textsuperscript{b}, Nuno Neves\textsuperscript{a},
Fernando M. V. Ramos\textsuperscript{a}

\textsuperscript{a}LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
\textsuperscript{b}CEG–IST, Instituto Superior Técnico, Universidade de Lisboa, Portugal

Abstract
Recently-proposed network 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 model considers security as a first class citizen, enabling the definition of flexible policies in three central areas: on the links, where alternative security compromises can be explored (e.g., encryption); on the switches, supporting various degrees of protection and redundancy if necessary; across multiples clouds, including public and private facilities, with the associated trust levels. We formulate the model 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 demands into network virtualization, providing evidence that this notion does not preclude high acceptance rates and efficient use of resources. In addition, by charging appropriate prices for the added security services, the providers can increase their average revenue.

Keywords: network virtualization; network embedding; security

1. 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 \cite{1,2}. 

Preprint submitted to Computer Networks September 22, 2017
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 [1].

So far, these modern platforms have been confined to a datacenter, controlled by a single cloud operator. This constraint can be an important barrier as more critical applications start shifting to the cloud. To overcome this problem, we are developing a platform that aims to extend network virtualization across multiple cloud providers [3, 4], 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 container (or 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 [5, 6], motivating the exploration of availability-enhancing alternatives (e.g., through replication 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 [7]. When a virtual network request 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 of 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 containers (e.g., covert channels) 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.

We have evaluated our proposal against the most commonly used VNE

---

1 We employ the terms user and tenant interchangeably in the paper.
alternative [8]. The results show a better behavior of our approach in terms of acceptance rate of requests. Even when a reasonable number of requests impose security constraints, the performance decrease is limited. This is a direct consequence of being harder to fulfill the requirements of the requests. With regard to resource utilization, the approach makes an efficient use of the links and nodes made available in the substrate, keeping a high average utilization.

The contributions of our work can be summarized as: (i) We formulate the SecVNE model 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 deployment; (ii) We propose a new policy language to specify the characteristics of the substrate network, and to allow the expression of user requirements; (iii) We compare our formulation with the most commonly used VNE alternative [8], and analyze various trade-offs related to embedding efficiency, costs and revenues.

2. Multi-Cloud Network Virtualization

Our multi-cloud network virtualization platform, Sirius, leverages from Software Defined Networking (SDN) to build a substrate infrastructure that spreads through both public clouds and private datacenters [3]. These resources can then be transparently shared by various users (or tenants) by allowing the definition and deployment of virtual networks (VN) composed of a number of containers arranged in an arbitrary network topology. While specifying the virtual network, it is possible to indicate several requirements for the switches and links, for example with respect to the needed bandwidth, CPU capacity, and security guarantees. These requirements are enforced during embedding by laying out the containers at the appropriate locations, where the substrate infrastructure still has enough resources to satisfy the particular demands. In addition, the datapaths are configured by the SDN controller by configuring the forwarding rules in the switches.

For example, as illustrated in Figure 1, virtual machines (VM) might be acquired at specific cloud providers to run tenants’ containers implementing distributed services. In this scenario, the most relevant security aspects that may need to be assessed are the following: First, the trust level associated to a cloud provider is influenced by various factors, which may have to be taken into consideration with more critical applications. Providers are normally better regarded if they show a good past track record on breaches and failures, have been on the market for a while, and advertise Service Level Agreements (SLAs) with stronger assurances for the users. Moreover, as the virtualization operator has full control over its own data centers, he
might employ protection features and procedures to make them compliant with regulations that have to be fulfilled by tenants (e.g., the EU GDPR that enters in force in 2018).

Second, VMs can be configured with a mix of defense mechanisms, e.g., firewalls and antivirus, to build execution environments with stronger degrees of security at a premium price. These mechanisms can be selected by the operator when setting up the VMs, eventually based on the particular requirements of a group of tenants, or they could be sold ready to use by the cloud providers (e.g., like in Amazon\textsuperscript{2} or Azure\textsuperscript{3} offerings). Highly protected VMs arguably give more trustworthy conditions for the execution of the switch employed by the container manager, ensuring correct packet forwarding among the containers and the external network (e.g., without being eavesdropped or tampered with by malicious co-located containers).

Third, the switches can also be configured with various defenses to protect the message traffic. In particular, it is possible to setup tunnels between switches implementing alternative security measures. For instance, if confidentiality is not a concern, then it is possible to add message authentication codes (MAC) to packets to afford integrity but without paying the full performance cost of encryption. Further countermeasures could also be added,

\textsuperscript{2}For example: Trend Micro Deep Security at aws.amazon.com/marketplace.  
\textsuperscript{3}For example: Check Point vSEC at azuremarketplace.microsoft.com.
such as denial of service detection and deep packet inspection to selected flows. In some cases, if trusted hardware is accessible (such Intel CPUs with SGX extensions in the private cloud), one could leverage from it to enforce greater isolation while performing the cryptographic operations, guaranteeing that keys would never be exposed.

The reader should notice that the above discussion would also apply to other deployment scenarios. For example, the virtualization operator could offer VNs of virtual machines in distinct cloud providers, which would mean that the relevant network appliances would be the switch utilized by the virtual machine manager and the corresponding links interconnecting them.

Substrate Network Modeling. Given the envisioned scenarios, the substrate network is modeled as a weighted undirected graph, composed of a set of nodes $N^S$ (e.g., switches/routers) and edges $E^S$ connecting them, $G^S = (N^S, E^S, A^S_N, A^S_E)$. Both the nodes and edges have attributes that reflect their particular characteristics. 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: they should be (i) expressive enough to represent the main security requirements when deploying virtual networks; (ii) easy to specify when configuring a network, requiring a limited number of options; (iii) implementable with ready available technologies.

The following attributes are considered for substrate nodes:

$$A^S_N = \{\{cpu^S(n), sec^S(n), cloud^S(n)\} \mid n \in N^S\}$$

The total amount of CPU that can be allocated for the switching operations of node $n$ is given by $cpu^S(n) > 0$. Depending on the underlying machine capacity and the division of CPU cycles among the various tasks (e.g., tenant jobs, storage, network), $cpu^S(n)$ can take a greater or smaller value. The security level associated to the node is $sec^S(n) \geq 0$. Nodes that run in an environment that implements stronger protections will have a greater value for $sec^S(n)$. The trustworthiness degree associated with a cloud provider is indicated with $cloud^S(n) \geq 0$.

The substrate edges have the following attributes:

$$A^S_E = \{\{bw^S(l), sec^S(l)\} \mid l \in E^S\}$$

The first attribute, $bw^S(l) \geq 0$, corresponds to the total amount of bandwidth capacity of the substrate link $l$. The security measures enforced by
the link are reflected in $\sec^S(l) \geq 0$. If the link implements tunnels that ensure integrity and confidentiality (by resorting to MACs and encryption) then it will have a higher $\sec^S(l)$ than a default edge that simply forwards packets.

**Virtual Network Modeling.** VN have an arbitrary topology and are composed by a number of nodes and the edges that connect them. When a tenant wants to instantiate a VN, besides indicating the nodes’ required processing capacity and bandwidth for the links, she/he may also include as requirements security demands. These demands are defined by specifying security attributes values associated with the resources.

In terms of modeling, a VN is also modeled as a weighted undirected graph, $G = (N, E, A_N, A_E)$, composed by a set of nodes $N$ and edges (or links) $E$. Both the nodes and edges have attributes that portray characteristics that need to be fulfilled when embedding is performed. Both $A_N$ and $A_E$ mimic the attributes presented for the substrate network. The only exception is an extra attribute that allows for the specification of security requirements related to availability.

The attribute $\avail^V(n)$ indicates that a particular node should have a backup replica to be used as a cold spare. This causes the embedding to allocate an additional node and the necessary links to connect it to the other nodes. These resources will only be used in case the virtualization platform detects a failures in the primary (or working) node / links. $\avail^V(n)$ defines where the backup of virtual node $n$ should be mapped. Typically, it would take value 0 if no backup is necessary. If virtual node $n$ should have a backup in another cloud (e.g., to survive cloud outages), then $\avail^V(n) = 1$. If $n$ should have the backup placed in the same cloud then $\avail^V(n) = 2$.

**Virtual Network Request.** VNRs are defined by the tenants of the system. They are modeled as a VN with two additional parameters, $VNR = (N, E, A^V_N, A^V_E, \text{Time}^V, \text{Dur}^V)$, where $\text{Time}^V$ is the arrival time of the VNR and $\text{Dur}^V$ is the interval of time during which the VN is valid.

3. Secure Virtual Network Embedding Problem

Our approach to VNE enables the specification of VNs to be mapped over a multi-cloud substrate, enhancing the security and flexibility of network virtualization. More precisely, we can define the Secure Virtual Network Embedding (SecVNE) problem as follows:

**SecVNE problem:** Given a virtual network request with resources and
security demands, $G^V$, and a substrate network $G^S$ with the resources to serve incoming VNRs, can $G^V$ be mapped to $G^S$ ensuring an efficient use of resources while satisfying the following constraints? (i) Each virtual edge is mapped to the substrate network meeting the bandwidth and security constraints; (ii) Each virtual node is mapped to the substrate network meeting the CPU capacity and security constraints (including node availability and cloud trust domain requirements).

Our approach handles the SecVNE problem, mapping a VN onto a substrate network respecting all constraints. When a VNR arrives, the optimal embedding is searched for to decrease the costs, i.e., reduce the total quantity of substrate resources allocated to it. It can happen that there are not enough substrate resources available at a certain instant, and in that case the incoming request has to be rejected. In order to increase the acceptance rate, we will allocate resources that are at least as secure as the ones specified in the VNR. This means that a VN might end up being mapped onto substrate resources deemed “more secure” than what is required. We find this option an acceptable trade-off as the alternative would cause resources to be under utilized if for instance during a period tenants only had VNRs with weak security demands. If it is possible to solve SecVNE for a request, then the asked resources will be consumed from the substrate for the period $Dur^V$ defined in the VNR.

Figure 2 illustrates the result of embedding a VNR (displayed on top) onto the substrate of Figure 1 (represented at the bottom). The VNR has node 1 that requires a medium level of security ($sec^V(1) = 3$) on a default trust cloud ($cloud^V(1) = 1$). The other node needs to be replicated ($avail^V(2) = 1$), in such a way that the primary and backup are in different clouds. It has a similar security demand ($sec^V(2) = 3$) but asks for more trusted clouds ($cloud^V(2) = 2$).

The chosen embedding guarantees that all requirements are satisfied. Node 1 is mapped on the left public cloud to a substrate node with a security level equal to the one requested ($sec^S(b) = 3$ and $cloud^S(b) = 1$). The other virtual node is embedded on more trustworthy clouds (with respectively $cloud^S(c) = 3$ and $cloud^S(e) = 5$). It is also possible to observe that one of the substrate paths (the primary/working) corresponds to more than one substrate edge (e.g., edge $(b,d)$ plus $(d,c)$), but all with the necessary security level (2 in this case). The figure also displays meta-links that connect the virtual nodes to the substrate nodes where they are mapped (e.g., line between 1 and $b$). This is an artifact in our modeling that is going to be explored in the MILP formulation.
4. A Policy Language to Specify SecVNE

Currently, we support two alternative ways for a user to indicate the information necessary to solve the SecVNE problem, namely give a description of the substrate network and the VNRs. The first is based on a graphical interface where the user can draw arbitrary substrates, with nodes and links and the associated attributes. The tenants can then depict the VNRs, which are then embedded into the substrate by our solution.

The other approach is based on a policy language that lets the user describe both the substrate and VNRs in a computer friendly manner, allowing scripts and tools to process them. The production rules of the grammar were kept relatively simple, but the achieved level of expressiveness is greater than what is attained with the graphical interface. As the characteristics of the substrate and VNRs are distinct, we explain them separately.

The substrate part of SecVNE policy grammar (top rows of Table 1) enables the listing of resources that compose the substrate. There are only functions and values to represent the current status of the network. For example, the leftmost cloud of the Figure 2 is specified as:

$$\text{substrate} \rightarrow \text{cpu}^S(a) = 80 \& \text{sec}^S(a) = 1 \& \text{cloud}^S(a) = 1 \&$$
$$\text{cpu}^S(b) = 80 \& \text{sec}^S(b) = 3 \& \text{cloud}^S(b) = 1 \&$$
Substrate Specification

\[ S \rightarrow func^S(\text{parameter}) = value_num \]
\[ S \rightarrow S \& S \]

Virtual Network Specification

\[ V \rightarrow func^V(\text{parameter}) = value_num \]
\[ V \rightarrow func^V(\text{parameter}) \geq value_num \]
\[ V \rightarrow !V; (V); V \& V; V | V \]

Table 1: Policy grammar to define SecVNE parameters.

\[ \text{bw}^S(a,b) = 100 \& \text{sec}^S(a,b) = 2 \& \ldots \]

In the virtual part of SecVNE policy grammar (bottom rows of Table 1),
the relations dictate the requirements for each node and link of the VNR.
As the grammar supports boolean operations, such as or ("|")
and ("&"), and not ("!")
, it is possible to express alternative constraints for the resources.
When processing a VNR containing a VN with several optional demands,
we generate all possible requests that would satisfy the tenant. Then, we
evaluate each one and select the solution with lowest cost. There are two
main benefits of this approach: (i) the acceptance rate grows because a
request may be mapped in more ways; (ii) the tenants can explore different
trade offs with respect to security (e.g., replication is only necessary if clouds
are not highly trusted).

As an example, consider the following VN with two nodes and one edge:
\[ VN \rightarrow (CPU^V(1) = 2 \& sec^V(1) = 3 \& cloud^V(1) \geq 1 \&
\quad avail^V(1) = 0 ) \& (CPU^V(2) = 3 \& avail^V(2) = 1 \&
\quad ((sec^V(2) \geq 1 \& cloud^V(2) \geq 4) \mid (sec^V(2) \geq 4 \& cloud^V(2) \geq 1))) \&
\quad bw^V(1,2) = 4 \& sec^V(1,2) = 2 \]

Node 1 needs to have security degree o 3 but the cloud trustworthiness can
be 1 or more. For node 2 there is a compromise between node security and
the degree of cloud trust, establishing two acceptable options where either
of the attributes needs to have a higher value (sec^V(2) or cloud^V(2) should
be larger or equal to 4). In this case, the VNR would be converted into two
requests, the first with the constraint (sec^V(2) \geq 1 \& cloud^V(2) \geq 4) (plus
other attributes) and the other with (sec^V(2) \geq 4 \& cloud^V(2) \geq 1) (plus
other attributes).

5. MILP Formulation

We have developed a MILP formulation to solve the SecVNE problem. The
section starts by explaining the decision variables used in the formulation,
Symbol | Meaning
---|---
\(w_{i,j}^{p,q} \geq 0\) | The amount of working flow, i.e., bandwidth, on the physical link \((p,q)\) for the virtual link \((i,j)\)
\(b_{i,j}^{p,q} \geq 0\) | The amount of backup flow, i.e., backup bandwidth, on the physical link \((p,q)\) for the virtual link \((i,j)\)
\(w_{i,j}^{p,q} \in \{0,1\}\) | Denotes whether the virtual link \((i,j)\) is mapped onto the physical link \((p,q)\). (1 if \((i,j)\) is mapped on \((p,q)\), 0 otherwise)
\(b_{i,j}^{p,q} \in \{0,1\}\) | Denotes whether the backup of virtual link \((i,j)\) is mapped onto the physical link \((p,q)\). (1 if backup of \((i,j)\) is mapped on \((p,q)\), 0 otherwise)
\(w_{n_i,p} \in \{0,1\}\) | Denotes whether virtual node \(i\) is mapped onto the physical node \(p\). (1 if \(i\) is mapped on \(p\), 0 otherwise)
\(b_{n_i,p} \in \{0,1\}\) | Denotes whether virtual node \(i\)'s backup is mapped onto the physical node \(p\). (1 if \(i\)'s backup is mapped on \(p\), 0 otherwise)
\(w_{c_i,c} \in \{0,1\}\) | Denotes whether virtual node \(i\) is mapped on cloud \(c\). (1 if \(i\) is mapped on \(c\), 0 otherwise)
\(b_{c_i,c} \in \{0,1\}\) | Denotes whether virtual node \(i\)'s backup is mapped on cloud \(c\). (1 if \(i\)'s backup is mapped on \(c\), 0 otherwise)

Table 2: Domain constraints (decision variables) used in the MILP formulation.

the objective function, and finally the constraints required to model the problem.

5.1. Decision variables

Table 2 presents the variables that are used in our MILP formulation. Briefly, \(w_{i,j}^{p,q}\), \(b_{i,j}^{p,q}\), \(w_{i,j}^{p,q}\) and \(b_{i,j}^{p,q}\) are related to working and backup links; \(w_{n_i,p}\) and \(b_{n_i,p}\) are associated with the working and backup nodes; \(w_{c_i,c}\) and \(b_{c_i,c}\) are related to the location of virtual node embedding in clouds.

\[
\begin{align*}
\hat{N}^V &= \{ i \in N^V : \text{avail}^V(i) = 0 \} \\
\bar{N}^V &= N^V \setminus \hat{N}^V \\
\hat{E}^V &= \{ (i,j) \in E^V : \text{avail}^V(i) = 0 \text{ and } \text{avail}^V(j) = 0 \} \\
\bar{E}^V &= E^V \setminus \hat{E}^V
\end{align*}
\]

Table 3: Auxiliary sets to facilitate the description of the constraints.

The formulation also employs a few auxiliary sets whose value depends on the VNR, as shown in Table 3. For example, \(\bar{N}^V = \emptyset\) means that no virtual node requires a backup. When this happens, we only model a working network in the substrate, making every backup related decision variable \((b_{f_{p,q}}^{i,j}, b_{b_{p,q}}^{i,j}, b_{n_i,p}, b_{c_i,c})\) become 0. On the other hand, if \(\bar{N}^V \neq \emptyset\), then we
model both a working and a backup network in the substrate. In this case, the decision variables take different values depending on the tenant request. If the virtual node $i$ has $\text{avail}^V(i) = 0$, indicating that there is no need to replicate, then both the working and the backup nodes of $i$ are placed in the same substrate node $p$ (i.e., $\text{wn}_{i,p} = \text{bn}_{i,p} = 1$), but the backup does not consume resources (e.g., CPU). When a virtual node $j$ has $\text{avail}^V(j) > 0$, it is necessary to locate the working and backup in different substrate nodes, belonging eventually to distinct clouds. Here, the backup will reserve the resources to be able to substitute the primary in case of failure.

5.2. Objective Function

The objective function wants to minimize three aspects (see Eq. [1]): 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 these objectives are measured in different units, we resort to a composite function, which can be parametrized and used to compute different solutions (others approaches could be used, see Steuer [9]). Thus, the formulation is based on a weighted-sum function with three different coefficients, $\beta_1$, $\beta_2$, and $\beta_3$, which should be reasonably parameterized for each objective.

$$
\begin{align*}
\min & \quad \beta_1 \left[ \sum_{i \in \mathcal{N}_V} \sum_{p \in \mathcal{N}_S} \text{cpu}^V(i) \cdot \text{sec}^S(p) \cdot \text{cloud}^S(p) \cdot \text{wn}_{i,p} \\
&+ \sum_{i \in \mathcal{N}_V} \sum_{p \in \mathcal{N}_S} \text{cpu}^V(i) \cdot \text{sec}^S(p) \cdot \text{cloud}^S(p) \cdot \text{bn}_{i,p} \right] \\
&+ \beta_2 \left[ \sum_{(i,j) \in \mathcal{E}_V} \sum_{(p,q) \in \mathcal{E}_S} \alpha_{p,q} \cdot \text{sec}^S(p,q) \cdot \text{w}_{f_{i,j}^{p,q}} \\
&+ \sum_{(i,j) \in \mathcal{E}_V} \sum_{(p,q) \in \mathcal{E}_S} \alpha_{p,q} \cdot \text{sec}^S(p,q) \cdot \text{b}_{f_{i,j}^{p,q}} \right] \\
&+ \beta_3 \left[ \sum_{(i,j) \in \mathcal{E}_V} \sum_{(p,q) \in \mathcal{E}_S} \text{w}_{l_{i,j}^{p,q}} + \sum_{(i,j) \in \mathcal{E}_V} \sum_{(p,q) \in \mathcal{E}_S} \text{b}_{l_{i,j}^{p,q}} \right]
\end{align*}
$$

The first part of Eq. [1] covers the computing costs, including both the working and backup nodes (top 2 lines). The second part is the sum of all working and backup link bandwidth costs (lines 3-4). The last part of the objective function achieves the third goal presented above. The equation
considers the level of security of the substrate resources, where the selection of higher security incurs in increased costs. Likewise, the costs are proportional to the trust associated with the cloud where the resource is located. To address the possibility that substrate edges connecting two distinct clouds might have a different cost (monetary, delay, or other) than links inside a cloud, we have added a multiplicative parameter \( \alpha_{p,q} \). This parameter is a weight for each physical link that may assume a different value depending on whether \((p,q)\) is a inter-cloud edge (connection between two clouds) or an intra-domain link (connection inside a cloud).

Intuitively, this objective function attempts to economize the most “powerful” resources (e.g., those with higher security levels) for VNRs that explicitly require them. Therefore, for instance, virtual nodes with \( \text{sec}^{V} = 1 \) will be mapped onto substrate nodes with \( \text{sec}^{S} = 2 \) if and only if there are no other substrate nodes with \( \text{sec}^{S} = 1 \) available.

5.3. Security Constraints

Below are enumerated the constraints related to the security of nodes, edges, and clouds:

\[
\begin{align*}
wn_{i,p} &\quad \text{sec}^{V}(i) \leq \text{sec}^{S}(p), \quad \forall i \in \mathcal{N}^{V}, \quad p \in \mathcal{N}^{S} \quad (2) \\
bn_{i,p} &\quad \text{sec}^{V}(i) \leq \text{sec}^{S}(p), \quad \forall i \in \mathcal{N}^{V}, \quad p \in \mathcal{N}^{S} \quad (3) \\
w_{i,j}^{p,q} &\quad \text{sec}^{V}(i,j) \leq \text{sec}^{S}(p,q), \quad \forall (i,j) \in \mathcal{E}^{V}, \quad (p,q) \in \mathcal{E}^{S} \quad (4) \\
b_{i,j}^{p,q} &\quad \text{sec}^{V}(i,j) \leq \text{sec}^{S}(p,q), \quad \forall (i,j) \in \mathcal{E}^{V}, \quad (p,q) \in \mathcal{E}^{S} \quad (5) \\
wn_{i,p} &\quad \text{cloud}^{V}(i) \leq \text{cloud}^{S}(p), \quad \forall i \in \mathcal{N}^{V}, \quad p \in \mathcal{N}^{S} \quad (6) \\
bn_{i,p} &\quad \text{cloud}^{V}(i) \leq \text{cloud}^{S}(p), \quad \forall i \in \mathcal{N}^{V}, \quad p \in \mathcal{N}^{S} \quad (7)
\end{align*}
\]

These constraints guarantee that a virtual node is only mapped to a substrate node that has a security level equal or greater than its demand (Eq. 2). They also ensure the same for backup nodes (Eq. 3). The following two equations force each virtual edge to be mapped to (one or more) physical links that provide a larger or equivalent security level as the request. This is true for links connecting the primary nodes and the backups. The last constraints ensure that a virtual node \( i \) is mapped to a substrate node \( p \) only if the cloud where \( p \) is located has a trust level equal or greater than the cloud demanded by \( i \) (both for working and backups — Eq. 6 and 7).

5.4. Mapping Constraints

Node Embedding: We force each virtual node to be mapped to exactly one working substrate node, and if requested, to a single backup substrate node
We also have to guarantee that \( (i) \) a substrate node only receives at most a virtual node (Eq. 10 - 12); \( (ii) \) however, as explained in Section 5.1, if the virtual node requires no replicas then its working and backup must be mapped onto the same substrate node (Eq. 13 - 14).

\[
\sum_{p \in N^S} w_{ni,p} = 1, \quad \forall i \in N^V \tag{8}
\]

\[
\sum_{p \in N^S} b_{ni,p} = 1, \quad \forall i \in N^V \tag{9}
\]

\[
\sum_{i \in N^V} w_{ni,p} \leq 1, \quad \forall p \in N^S \tag{10}
\]

\[
\sum_{i \in N^V} w_{ni,p} + b_{nj,p} \leq 1, \quad \forall j \in N^V, \quad p \in N^S \tag{11}
\]

\[
\sum_{i \in N^V \setminus \{j\}} b_{ni,p} \leq w_{ni,p}, \quad \forall i \in N^V, \quad p \in N^S \tag{12}
\]

\[
\sum_{i \in N^V} b_{ni,p} \leq w_{ni,p}, \quad \forall i \in N^V, \quad p \in N^S \tag{13}
\]

\[
\sum_{i \in N^V} w_{ni,p} + b_{nj,p} \leq 2, \quad \forall j \in N^V, \quad p \in N^S \tag{14}
\]

The next constraints create relationships among the nodes and flows.

\[
w^{i,j}_{p,q} \geq w^i_{f^{i,j}_{p,q}}, \quad \forall (i, j) \in E^V, \quad (p, q) \in E^S \tag{15}
\]

\[
b^{i,j}_{p,q} \geq b^i_{f^{i,j}_{p,q}}, \quad \forall (i, j) \in E^V, \quad (p, q) \in E^S \tag{16}
\]

\[
w^{i,j}_{p,q} = w^{i,j}_{q,p}, \quad \forall (i, j) \in E^V, \quad p, q \in N^S \cup N^V \tag{17}
\]

\[
b^{i,j}_{p,q} = b^{i,j}_{q,p}, \quad \forall (i, j) \in E^V, \quad p, q \in N^S \cup N^V \tag{18}
\]

\[
\sum_{p \in N^S} (w_{ni,p \text{ doesItBelong}_{p,c}}) \geq w_{ci,c}, \quad \forall i \in N^V, \quad c \in C \tag{19}
\]

\[
\sum_{p \in N^S} (b_{ni,p \text{ doesItBelong}_{p,c}}) \geq b_{ci,c}, \quad \forall i \in N^V, \quad c \in C \tag{20}
\]

Eq. 15 ensures that if there is a flow between nodes \( p \) and \( q \) for a virtual edge \( (i, j) \), then this means that \( (i, j) \) is mapped to the substrate link whose end-points are \( p \) and \( q \). For example, if \( w^i_{f^{i,j}_{p,q}} \neq 0 \) then \( w^{i,j}_{p,q} = 1 \). The next equation achieves the same goal but for the backup. We also include two binary constraints to force the symmetric property for the binary variables.
related with links (Eq. 17 - 18). In a similar fashion, we also need to establish a relation between the virtual nodes and the clouds where they are embedded (both for working and backups) (Eq. 19 - 20). Namely, if virtual node $i$ is mapped onto a substrate node $p$ and $p$ belongs to cloud $c$, then $i$ is mapped on cloud $c$. Parameter $doesItBelong_{p,c}$ is 1 if substrate node $p$ belongs to cloud $c$, and 0 otherwise.

Since we allow the tenant to choose between having no replication, or replication in one cloud or across different clouds, it is necessary to specify these constraints. First, we require each virtual node to be mapped to exactly one cloud (working or backup, Eq. 21 - 22). Parameter $wantBackup$ assumes value 1 if a backup is needed for at least one of the nodes of a VNR or value 0 otherwise. Second, we must restrict the location of the working and backup nodes to the same or distinct clouds, depending on the value of the availability attribute ($avail^V(i)$) (Eq. 23).

$$\sum_{c \in C} wc_{i,c} = 1, \forall i \in N^V$$  (21)$$\sum_{c \in C} bc_{i,c} = wantBackup, \forall i \in N^V$$  (22)$$|wc_{i,c} \cdot wantBackup - bc_{i,c}| = (avail^V(i) - 1) \times (wc_{i,c} \cdot wantBackup + bc_{i,c}), \forall i \in N^V, c \in C$$  (23)

**Link Embedding:** These constraints are related to the mapping of virtual links into the substrate. They take advantage of the meta link artifact (recall Figure 2), which connects a virtual node $i$ to the substrate node $p$ where it is mapped, to enforce a few restrictions.

$$wn_{i,p} \cdot bw^V(i, j) = w_{i,p}^{j,i}, \forall (i, j) \in E^V, p \in N^S$$  (24)$$wn_{j,q} \cdot bw^V(i, j) = w_{j,q}^{i,j}, \forall (i, j) \in E^V, p \in N^S$$  (25)$$bn_{i,p} \cdot bw^V(i, j) = b_{i,p}^{j,i} \cdot wantBackup, \forall (i, j) \in E^V, p \in N^S$$  (26)$$bn_{j,q} \cdot bw^V(i, j) = b_{j,q}^{i,j} \cdot wantBackup, \forall (i, j) \in E^V, q \in N^S$$  (27)$$\sum_{j,k \neq i} w_{i,p}^{j,k} + w_{j,q}^{i,k} + b_{i,p}^{j,k} + b_{j,q}^{i,k} = 0, \forall i \in N^V, p \in N^S$$  (28)

Notice that in the implementation, the modulus function had to be linearized because it is not allowed with variables as parameters.

---

4 Notice that in the implementation, the modulus function had to be linearized because it is not allowed with variables as parameters.
These constraints guarantee that the working flow of a virtual link \((i, j)\) always departs from \(i\) and arrives to \(j\), passing through the corresponding substrate nodes (\(p\) and \(q\) ) (Eq. 24 and 25). The next two equations compel the same requirement for the backup nodes. Notice that even though the backup path is only used if the working substrate path fails, we reserve the necessary resources during embedding to make sure they are available when needed. Eq. 28 forces meta-links to carry only 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.

The next equations specify flow conservation restrictions at the nodes.

\[
\sum_{p \in N^S} w_{f^i_{j,p}} - \sum_{p \in N^S} w_{f^j_{p,i}} = bw^V(i, j), \quad \forall (i, j) \in E^V 
\]

(29)

\[
\sum_{p \in N^S} w_{f^j_{i,p}} - \sum_{p \in N^S} w_{f^i_{p,j}} = -bw^V(i, j), \quad \forall (i, j) \in E^V 
\]

(30)

\[
\sum_{p \in N^S \cup N^V} w_{f^i_{j,p}} - \sum_{p \in N^S \cup N^V} w_{f^j_{p,i}} = 0, \quad \forall (i, j) \in E^V, \quad q \in N^S 
\]

(31)

Eq. 29, 30 and 31 refer to the working flow conservation conditions, which denote that the network flow to a node is zero, except for the source and the sink nodes, respectively. In an analogous way, the following three equations refer to the backup flow conservation conditions (Eq. 32, 33 and 34). The next constraints guarantee the same bandwidth in both directions. The first two equations for working and backups separately (Eq. 35 - 36) and the others in case there are relations between them (Eq. 37 - 38).

\[
w_{f^i_{j,p,q}} = w_{f^j_{q,p,i}}, \quad \forall (i, j) \in E^V, \quad p, q \in N^S \cup N^V 
\]

(35)

\[
b_{f^i_{j,p,q}} = b_{f^j_{q,p,i}}, \quad \forall (i, j) \in E^V, \quad p, q \in N^S \cup N^V 
\]

(36)

\[
w_{f^i_{j,p,q}} = w_{f^j_{p,q,i}}, \quad \forall (i, j) \in E^V, \quad p, q \in N^S 
\]

(37)

\[
w_{f^j_{i,p,q}} = w_{f^i_{q,p,i}}, \quad \forall (i, j) \in E^V, \quad p, q \in N^S 
\]

(38)
Nodes and Links Disjointness: Since any substrate node or link of a working path can fail, we have to ensure that paths connecting the backups of the virtual nodes are disjoint from the substrate resources that are being used for the working part (otherwise a single failure could compromise both paths). The auxiliary binary variables $working_{p,q}$ and $backup_{p,q}$ define if a physical link $(p,q)$ belongs to the working or backup networks in the substrate.

\[ working_{p,q} \leq 1 - backup_{p,q}, \forall (p,q) \in E^S \]  \hspace{1cm} (39)

\[ wi_{i,j} \leq working_{p,q}, \forall (i,j) \in E^V, (p,q) \in E^S \]  \hspace{1cm} (40)

\[ bl_{i,j} \leq backup_{p,q}, \forall (i,j) \in E^V, (p,q) \in E^S \]  \hspace{1cm} (41)

First, we require disjointness between the working and backup parts (Eq. 39). Second, we guarantee that if the working path of a virtual edge $(i,j)$ is mapped onto a substrate link $(p,q)$, then $(p,q)$ needs to be in the working part. Similarly, we constraint the backups (Eq. 40 - 41).

5.5. Capacity Constraints

Node Capacity Constraints: Virtual nodes from different VNRs can be mapped to the same substrate node. For instance, a substrate node can receive both a working node of a virtual node $i$ from a VNR $x$ and a backup node of a virtual node $j$ from a VNR $y$. Let’s call $N^V$ the set of all virtual nodes belonging to every VNR that is at this moment mapped onto the substrate and $i \uparrow p$ to indicate that virtual node $i$ is hosted on the substrate node $p$. Then, the residual capacity of a substrate node, $R_N(p)$, is defined as the currently available CPU capacity of the substrate node $p \in N^S$.

\[ R_N(p) = cpu^S(p) - \sum_{i \in N^V} cpu^V(i), \forall p \in N^S \]

For a substrate node, it is necessary to ensure that we never allocate more than the residual capacity when carrying out a new embedding. This needs to take into consideration both the resources consumed by the working and backups (Eq. 42).

\[ \sum_{i \in N^V} wn_{i,p} \cdot cpu^V(i) + \sum_{j \in N^V} bn_{j,p} \cdot cpu^V(j) \leq R_N(p), \forall p \in N^S \]  \hspace{1cm} (42)

Link Capacity Constraints: Similarly, substrate links can also map virtual edges from different VNRs. Let’s define $E^V$ as the set of all virtual edges of every VNR currently mapped onto the substrate and $(i,j) \uparrow (p,q)$ denote that the flow of the virtual link $(i,j)$ traverses the substrate link $(p,q)$. The
residual capacity of a substrate link, $R_E(p,q)$, is defined as the total amount of bandwidth available on the substrate link $(p,q) \in E^S$.

$$R_E(p,q) = bw^S(p,q) - \sum_{(i,j) \uparrow (p,q)} bw^V(i,j), \ (i,j) \in E^V$$

The following constraint ensures that the allocated capacity of a substrate link should be less than the residual capacity of that physical link, taking into consideration both the working and backup parts.

$$\sum_{(i,j) \in E^V} w f_{p,q}^{i,j} + \sum_{(i,j) \in E^V} b f_{p,q}^{i,j} \leq R_E(p,q), \ \forall (p,q) \in E^S$$ (43)

6. Evaluation

This section presents performance results of our solution in random and Waxman network topologies and in diverse VNR settings. The simulations show promising results as our solution was able to show high acceptance rates and substrate resource utilizations across the various experiments.

6.1. Experimental Setup

We have extended a simulator [10] to evaluate the embedding when processing the dynamic arrival of VNRs to a system. To create the substrate networks we resorted to the GT-ITM tool [11]. Two kinds of networks were utilized: one based on random topologies, where every pair of nodes is randomly connected with a probability between 25% and 30%; and the other employing the Waxman model to link the nodes with a probability 50% [12].

Substrate networks have a total of 25 nodes. CPU and bandwidth ($cpu^S$ and $bw^S$) of nodes and links is uniformly distributed between 50 and 100. These resources are also uniformly associated with one of three levels of security ($sec^S \in \{1.0, 1.2, 5.0\}$). The rationale for these values is to achieve a good balance between the diversity of security levels and their monetary cost. We performed an analysis of the pricing schemes of Amazon EC2 and Microsoft Azure for plain and secure VMs. It was possible to observe a wide range of values depending on the included defenses. For example, while an EC2 instance that has container protection is around 20% more expensive than a normal instance (hence our choice of 1.2 for the intermediate level of security), the cost of instances that offer threat prevention or encryption is at least 5 times greater (our choice for the highest level of security).

The substrate nodes are also uniformly divided among three clouds, each one with a different security level ($cloud^S \in \{1.0, 1.2, 5.0\}$), which are justified along the same line of reasoning. The goal is to represent a setup that
| Notation | Algorithm description |
|----------|-----------------------|
| NS+NA    | SecVNE with no security or availability requirements for VNs |
| 10S+NA   | SecVNE with VNRs having 10% of their resources (nodes and links) with security requirements (excluding availability) |
| 20S+NA   | Similar to 10S+NA, but with security requirements (excluding availability) for 20% of the resources |
| NS+10A   | SecVNE with no security requirements, but with 10% of the nodes requesting replication for increased availability |
| NS+20A   | Similar to NS+10A, but with 20% of the nodes asking for replication |
| 20S+20A  | SecVNE with 20% of the resources (nodes and links) with security requirements and 20% of the nodes with replication |
| D-ViNE   | VNE MILP model presented in [8] |

Table 4: VNR configurations that were evaluated in the experiments.

includes a public cloud (lowest level), a trusted public cloud, and a private datacenter (assumed to offer the highest security).

VNRs have a number of virtual nodes uniformly distributed between 2 and 4\(^5\). Pairs of virtual nodes are connected with a Waxman topology with probability 50%. The CPU and bandwidth of the virtual nodes and links are uniformly distributed between 10 and 20. Several alternative security and availability requirements are evaluated, as shown in Table 4. We assume that VNRs arrivals \((Time^V)\) are modeled as a Poisson process with an average rate of 4 VNRs per 100 time units. Each VNR has an exponentially distributed lifetime \((Dur^V)\) with an average of 1000 time units.

The MILPs are solved using the open source library GLPK [13]. In the objective function, we set \(\beta_1 = \beta_2 = \beta_3 = 1\) to balance evenly the cost components (Eq. 1). Parameter \(\alpha\) was also set to 1 because our pricing analysis showed negligible differences in cost between intra- and inter-cloud links in most of the relevant scenarios. We setup 20 experiments, each with a different substrate topology (10 random and 10 Waxman). Every experiment ran for 50 000 time units, during which embedding is attempted for a group VNRs (10 sets of 2 000 VNRs were tested). The order of arrival and the capacity requirements of each VNR are kept the same in each of the configurations of Table 4 ensuring that they solve equivalent problems.

\(^5\)Notice that a node corresponds to a switch, which can connect many hundreds of containers in a large VM (recall Figure 1). Therefore, a VNR with 4 nodes can easily link together in the order of a thousand of containers.
In the evaluation, we compared our approach with the algorithm D-ViNE [8]. D-ViNE was chosen because it has been considered as the baseline for many VNE works and due to the availability of its implementation as open-source software. D-ViNE requirements are only based on CPU and bandwidth capacities, while our algorithm adds to these requirements also security demands, including availability needs, and cloud preferences.

6.2. Metrics
We used several performance metrics for the evaluation:
- **VNR acceptance ratio**: the percentage of accepted requests (i.e., the number of accepted VNRs divided by the total number of VNRs);
- **Node stress ratio**: average load on the substrate nodes (i.e., average over all nodes of the percentage of CPU that is in use);
- **Link stress ratio**: average load on the substrate links (i.e., average over all edges of the percentage of bandwidth that is in use);
- **Average revenue by accepting VNRs**: One of the main goals of VNE is to maximize the profit of the virtualization provider. For this purpose, and similar to [8, 14], the revenue generated by accepting a VNR is proportional to the value of the acquired resources. As such, in our case, we take into consideration that stronger security defenses will be charged at a higher (monetary) value. Therefore, the revenue associated with a VNR is:

\[
R(VNR) = \lambda_1 \sum_{i \in N^V} [1 + \varphi_1(i)] \text{cpu}^V(i) \ sec^V(i) \ cloud^V(i) + \\
\lambda_2 \sum_{(i,j) \in E^V} [1 + \varphi_2(i,j)] \text{bw}^V(i,j) \ sec^V(i,j),
\]

where \( \lambda_1 \) and \( \lambda_2 \) are scaling coefficients that denote the relative proportion of each revenue component to the total revenue. These parameters offer providers the flexibility required to price differently the different resources. Variables \( \varphi \) account for the need to have backups, either in the nodes \( \varphi_1(i) \) or in the edges \( \varphi_2(i,j) \) (\( \varphi_1(i) = 1 \) if a backup is required or 0 otherwise; \( \varphi_1(i,j) = 1 \), in case of at least one node needs a backup or 0 otherwise).

This metric accounts for the average revenue obtained by embedding a VNR (i.e., the total revenue generated by accepting the VNRs divided by the number of accepted VNRs). In the experiments, we set \( \lambda_1 = \lambda_2 = 1 \).
- **Average cost of accepting a VNR**: 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 certain virtual edges
may end up being embedded in more than one physical link (as in the substrate edge between nodes $b$, $d$ and $c$, in Figure 2). The cost may also increase if the VNR requires higher security for its virtual nodes and links. Thus, we define the cost of embedding a VNR as:

$$
C(VNR) = \lambda_1 \sum_{i \in N^V} \sum_{p \in N^S} cpu^i_p \ sec^S(p) \ cloud^S(p) + \\
\lambda_2 \sum_{(i,j) \in E^V} \sum_{(p,q) \in E^S} f_{p,j}^{i,j} \ sec^S(p,q),
$$

where $cpu^i_p$ corresponds to the total amount of CPU allocated on the substrate node $p$ for virtual node $i$ (either working or backup). Similarly, $f_{p,j}^{i,j}$ denotes the total amount of bandwidth allocated on the substrate link $(p,q)$ for virtual link $(i,j)$. $\lambda_1$ and $\lambda_2$ are the same weights introduced in the revenue formula to denote the relative proportion of each cost component to the total cost.

6.3. Evaluation Results

Figure 3a displays the acceptance ratio over time for one particular experiment with a random topology substrate. We can observe that after the first few thousand time units, the acceptance ratio tends to stabilize. A similar trend also occurs with the other experiments, and for this reason the rest of the results are taken at the end of each simulation. Due to space constraints, the results in the following graphs are from the Waxman topologies only (Figure 3b - 3f). We note however that the conclusions to be drawn are exactly the same as for the random topologies. The main conclusions are:

**SecVNE** exhibits a higher average acceptance ratio when compared to D-ViNE, not only for the baseline case, but also when including security requirements: Figure 3b indicates that SecVNE can make better use of the available substrate resources to embed the arriving VNRs when compared to the most commonly employed VNE algorithm. It is interesting to note that SecVNE is better than D-Vine even when 20% of the VNRs include security requirements, which are harder to fulfill. This does not mean D-Vine is a poor solution – it merely shows that its model is not the best fit for our particular problem. In particular, D-Vine uses geographical distance of substrate nodes as one of the variables to consider in node assignment. This parameter is less relevant in our virtualized environment but constrains D-Vine options. In any case, notice that the results for D-Vine represent its best configuration with respect to geographical location – we have tested D-Vine with the entire range of options for this parameter.
A richer set of demands decreases the acceptance ratio, but only slightly: VNRs with stronger requirements have a greater number of constraints that need to be satisfied, and therefore it becomes more difficult to find the necessary substrate resources to embed them. However, a surprising result is the small penalty in terms of acceptance rate in the presence of security demands (see Figure 3b again). For instance, an increase of 20 percentage points (pp) in the resources with security needs results in a penalty of only around 1 pp in the acceptance ratio. Also interesting is the fact that the reduction in acceptance ratio is more pronounced when VNRs have availability requirements, when compared to security. In this case, an increase of 20 pp in the number of nodes with replication results in a penalty of around 10 pp. This is because of the higher use of substrate resources due to the reservation of backup nodes/links.

Security demands only cause a small decrease on substrate resources utilization. Figures 3c and 3d show the substrate node and link stress ratio, respectively. We observe that the utilization of node resources is very high in all cases (over 80%), meaning the mapping to be effective. It is also possible to see that slightly more resources are allocated in the substrate network with SecVNE than with D-ViNE, which justifies the higher acceptance ratio achieved. If the existing resources are used more extensively to
be able to serve more virtual network requests, then the assignment of virtual requests is being more effective. As the link stress ratio is lower (again, for all cases), this means the bottleneck is the node CPU. Finally, the link stress ratio of D-Vine is lower than in our solution. This is due to D-vine incorporating load balancing into the formulation.

**Security and availability requirements increases costs and revenues.** Figures 3e and 3f display the average cost and revenue for each VNR embedding, respectively. The results show that reasonable increases in the security requirements (10% and 20%) only cause a slight impact on the costs. However, higher costs are incurred to fulfill availability needs due to the extra reservation of resources (nodes and links). Since D-vine does not consider security and availability aspects, it ends up choosing embeddings that are more expensive (e.g., with respect to “NS+NA”). In terms of revenue, it can be observed that by charging higher prices for security services, virtualization providers can significantly enhance their income — average revenue almost doubles for a 20% increase in security needs.

7. Related Work

There is already a wide literature on this problem [7]. Yu et al. [14] where the first of to 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 [8], 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 survivability in VNE has gained attention recently. H. Yu et al. [15] 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. [16] 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. [17] 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. [18] 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 user-centric cloud setting with different trust domains.

Another problem in the same area is to ensure virtual network connectivity in presence of multiple substrate link failures. It is explored by Shahriar et al. [19] where two solutions are proposed, one completely heuristic (augment the virtual network and computing the virtual links to be embedded disjointly and the embedding itself considering the disjointly requirements) and other with the second part not heuristic.

The majority of the works in VNE field only consider a single InP. This may not be the most realistic scenario nowadays, since many VNs need to be provisioned across heterogeneous administrative domains belonging to multiple InPs to deploy and deliver services end to end. M. Chowdhury et al. [20] address the conflicts of interest between SPs (that are interested in satisfying their demands while minimizing their expenditure) and InPs (that strives to optimize the allocation in its equipment by getting requests with higher revenue while offloading unprofitable work onto their competitors). They present PolyVINE a policy-based end-to-end VNE framework - that, in short, partitions a VN request into $k$ subgraphs to be embedded onto $k$ SNs, establishes inter-connections between the $k$ subgraphs using inter-domain paths, and embeds each subgraph in each InP SN using an intra-domain algorithm.

In this context, Nonde et al. [21] and Houidi et al. [22] proposed VNE considering cloud environments. The first one, researched an energy efficient VNE where power savings are introduced by consolidating resources in the networks/datacenters. A naive greed heuristic is proposed based on the CPU usage and security characteristics are not considered. And, the second paper, considered VNE in a cloud environment to develop a multi-objective algorithm taking into account some objectives, constraints, including survivability. However, both of the works only handled single clouds/datacenters issues. That is, they do not explore properly user-centric cloud environments.
8. Conclusions

The paper proposes a multi-cloud VNE solution that addresses a diverse set of security requirements, applied both to communication links and virtual nodes. These requirements enable several trade-offs to be explored with regard to the selected defenses. In addition, multiple clouds are considered with distinct levels of trust. By not relying on a single cloud provider we avoid internet-scale single points of failures (with the support of backups), and privacy issues can be accommodated by constraining the mapping of certain virtual nodes to specific classes of clouds (e.g., private). The experimental results show that our solution leads to a high request acceptance rate and an efficient use of substrate resources. The inclusion of requests with stronger security demands can cause an increase on the average revenue for the provider if appropriate pricing schemes are employed.

Acknowledgments: This work was partially supported by the EC through project SUPERCLOUD (H2020-643964), and by national funds of FCT with reference UID/CEC/00408/2013 (LaSIGE).

References

References

[1] T. Koponen, et al., Network virtualization in multi-tenant datacenters, in: USENIX NSDI, 2014.

[2] A. Al-Shabibi, et al., OpenVirteX: Make your virtual SDNs programmable, in: HotSDN, 2014.

[3] M. Alaluna, et al., Secure and Dependable Multi-Cloud Network Virtualization, in: XDOM0, 2017.

[4] M. Lacoste, et al., User-Centric Security and Dependability in the Clouds-of-Clouds, IEEE Cloud Computing 3 (5) (2016) 64–75.

[5] A. Froehlich, 9 spectacular cloud computing fails, information Week (Jul. 2015).

[6] J. Tsidulko, The 10 biggest cloud outages of 2016 (so far), the Channel Company (Jul. 2016).
[7] A. Fischer, et al., Virtual Network Embedding: A Survey, IEEE Communications Surveys Tutorials 15 (4) (2013) 1888–1906.

[8] C. M., et al., ViNEYard: Virtual Network Embedding Algorithms with Coordinated Node and Link Mapping, IEEE/ACM Transactions on Networking 20 (1) (2012) 206–219.

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

[10] ViNE-Yard, http://www.mosharaf.com/ViNE-Yard.tar.gz.

[11] E. W. Zegura, et al., How to Model an Internetwork, in: IEEE INFOCOM, 1996, pp. 594–602.

[12] M. Naldi, Connectivity of Waxman Topology Models, Computer Communications 29 (1) (2005) 24–31.

[13] GLPK, GNU Linear Programming Kit, http://www.gnu.org/software/glpk/ (2008).

[14] M. Yu, et al., Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration, ACM SIGCOMM Computer Communication Review 38 (2) (2008) 17–29.

[15] H. Yu, et al., Cost Efficient Design of Survivable Virtual Infrastructure to Recover from Facility Node Failures, in: IEEE ICC, 2011.

[16] M. Rahman, et al., Survivable Virtual Network Embedding, in: NETWORKING, 2010, pp. 40–52.

[17] A. Fischer, et al., Position paper: Secure virtual network embedding, Praxis der Informationsverarbeitung und Kommunikation.

[18] S. Liu, et al., Security-Aware Virtual Network Embedding, in: IEEE ICC, 2014.

[19] N. Shahriar, et al., Connectivity-Aware Virtual Network Embedding, in: IFIP Networking, 2016.

[20] M. Chowdhury, et al., PolyViNE: Policy-based Virtual Network Embedding Across Multiple Domains, in: ACM SIGCOMM VISA, 2010.

[21] L. Nonde, et al., Energy Efficient Virtual Network Embedding for Cloud Networks, Journal of Lightwave Technology 33 (9) (2015) 1828–1849.
[22] I. Houidi, et al., Exact Multi-Objective Virtual Network Embedding in Cloud Environments, The Computer Journal 58 (3) (2015) 403–415.