Specificity vs. Flexibility: On the Embedding Cost of a Virtual Network

Arne Ludwig, Stefan Schmid, Anja Feldmann
Telekom Innovation Laboratories & TU Berlin, Germany
{aludwig,stefan,anja}@net.t-labs.tu-berlin.de

***Abstract*—The virtualization trend in today's Internet decouples services from the constraints of the underlying physical infrastructure. This decoupling facilitates more flexible and efficient resource allocations: the service can be realized at *any* place in the substrate network which fulfills the service specification requirements.**

This paper studies such flexibilities in the context of *virtual network (VNet)* embeddings. The network virtualization paradigm envisions an Internet where users can request arbitrary VNets from a substrate provider (e.g., an ISP). A VNet describes a set of virtual nodes which are connected by virtual links; both nodes and links provide certain QoS or resource guarantees. While some parts of the VNet may be fully specified (e.g., the node and link locations or technologies), other parts may be flexible or left open entirely.

We analyze how flexible specifications can be exploited to improve the embedding of virtual networks. We define a measure for specificity and introduce the notion of the *Price of Specificity (PoS)* which captures the resource cost of the embedding under a given specification. We identify parameters on which the Price of Specificity depends, and evaluate its magnitude in different scenarios.

For example, we find that the PoS can be large even in small settings, and depends both on the substrate size as well as—to a larger extent—the load of the substrate. Moreover, while skewed distributions of resources can yield better allocations, they entail the risks of a high PoS if the demand does not perfectly match. We also provide a formal analysis of the impact of migration, and show that the option to migrate can sometimes increase resource costs.

I. Introduction

Virtualization is a central design principle in the Internet today. While end-system virtualization has already been very successful in the context of datacenters and cloud computing, the virtualization trend now spills over to the *network*. A recent example is Google's *G-Scale network* [1], a *Software Defined Networking (SDN)* implementation which allows to flexibly and securely manage wide-area network traffic.

Even the typically more conservative *Internet Service Providers (ISPs)* have become interested in the opportunities of virtualization. Especially *network virtualization* [9] can be a way for ISPs to innovate their network as well as to offer novel services: many ISPs do not only have a large and fast network, but also many geographically distributed resources such as storage and computation (e.g., in their "micro-datacenters", the *Points-of-Presence* and the *street cabinets*). If these resources can be used (and/or leased) for new services, the infrastructure can be monetarized better and its efficiency is improved. Note that in contrast to other players in the Internet, e.g., *Content Distribution Network (CDN)* providers, ISPs have the advantage of having a detailed view of their infrastructure and the customers demand; this knowledge can be exploited to use the resources where and when they are most useful.

In this sense, we envision that in the future, ISPs will offer on-demand *virtual networks (VNets)*: networks which can be flexibly specified not only in terms of latency and bandwidth requirements along the virtual links, but also in terms of storage and computational requirements at the virtual nodes (e.g., at the Points-of-Presence where they are mapped to). The virtual network provides isolation from other VNets and appears as a "dedicated network". The resulting Quality-of-Service (QoS) guarantees can be interesting for a business customer who likes to set-up a high-quality multimedia conference call, or a startup company which cannot afford its own infrastructure yet.

In a fully virtualized resource infrastructure, the location where a VNet is realized (or embedded) *is only restricted by the VNet request specification*. For example, if a customer insists that his VNet nodes run on a 64-bit architecture, the choice of resources is restricted and the VNet may be more expensive to realize compared to a situation where also 32-bit architectures are allowed. Similarly, if a customer requires the VNet to be realized over storage resources in Switzerland only, the VNet embedding can be more expensive than if the requirements are less restrictive and allow, e.g., to exploit Europe-wide storage sites.

Our Contribution. This paper studies the impact of specificity in the context of network virtualization. We assume a two-player setting consisting of a *customer* (who requests specific VNets) and a *provider*. The customer could for example be a startup company or even a *Virtual Network Provider (VNP)*. The provider can be a *Physical Infrastructure Provider (PIP)*, e.g., an ISP, or again a VNP.

We assume that the customer specifies certain requirements of the VNet, and the provider will try to realize the VNet in a most resource-efficient manner subject to the customer's specifications. We investigate the tradeoff between VNet specificity and embedding costs. In order to avoid artifacts from heuristic or approximate VNet solutions, our methodology is based on *optimal embeddings*. Accordingly, we present a simple optimal algorithm to compute VNet embeddings, which also supports migration.

We present a formal model to measure the specificity σ of a given VNet request, and then introduce the notion of the *Price*

of Specificity (PoS). PoS(σ) captures the increased embedding cost of a given VNet request of specificity σ. We then identify different types of specifications (such as requirements on resource types and vendor, geographical embedding constraints, or whether migration is allowed after an initial placement), and analyze their influence on the VNet allocation cost. It turns out that the PoS depends both on the substrate size as well as the load of the substrate, while the load has a larger impact than a proportionally similar change of the VNet size; sometimes, the embedding cost can be larger than two (i.e., PoS(σ)>2), even in small settings. Our results also confirm the intuition that the relationship between the distribution of the requested and the supplied resource types is important: While skewed distributions of resources can yield better allocations, they entail the risks of a high PoS if the demand does not perfectly match the supply. Although migration is regarded as one of the advantages of network virtualization and we generally observe positive results in our experiments, we will also present a scenario where migration can also *increase* the resource costs (and hence the Price of Specificity). This is shown in a first formal analysis of the PoS.

We believe that our evaluation not only sheds light onto the resource costs of a VNet in different scenarios (and hence in some sense, the real "value" of a resource) but can also provide insights on how to structure a substrate network in order to increase the number of embeddable networks (and reduce the Price of Specificity) at minimal cost.

In this sense, we regard our work as a first step towards a better understanding of the economical dimension of the VNet embedding problem, and we provide a short discussion of its limitations and further directions.

Paper Organization. The upcoming section (Section II) provides some relevant background information on the envisioned network virtualization architecture. Section III presents our optimal embedding algorithm *FlexMIP*. The Price of Specificity is introduced formally in Section IV. Section V evaluates the Price of Specificity in different scenarios. We extend our discussion and analytically study the use of migration VI. After reviewing related work in Section VII, we conclude our paper in Section VIII.

II. The Age of Flexible Embeddings

Network virtualization has the potential to change Internet networking by allowing multiple, potentially heterogeneous and service-specific virtual networks (VNets) to cohabit a shared substrate network. This section first discusses some economical implications of the paradigm and the roles envisioned in our own network virtualization prototype architecture [25], and subsequently formally introduces the VNet embedding problem. Finally, we quickly discuss how VNet specifications can be described in a standardized manner.

Economic Roles. We envision a network virtualization environment where services are offered and realized by different *economic roles*, see [25] for a detailed discussion. In a nutshell, we assume that the physical network, or more generally: the *substrate network*, is owned and managed by one or several *Physical Infrastructure Providers* (PIP) while virtual network abstractions are offered by so-called *Virtual Network Providers* (VNP). VNPs can be regarded as resource *brokers*, buying and combining resources from different PIPs. The virtual network is operated by a so-called *Virtual Network Operator* (VNO). Finally, there is a *Service Provider* (SP) specifying and offering a flexible service.

In such an environment, a service is realized in multiple steps, and the application or service specifications are communicated from the SP down the hierarchy to the PIP. While the SP may specify the service on a high level (e.g., regarding maximally tolerable latencies experienced by users accessing the service), the specification is transformed by the VNP to a VNet topology describing which virtual node resources should be realized by which PIP and how to connect; in other words, the VNet is embedded by the VNP on a graph consisting of PIPs. The PIP then transforms the specification into a concrete VNet allocation / embedding on its substrate network.

Depending on the specificity of the request obtained from the preceding role in the hierarchy, a VNP or PIP provider can optimize the allocation for its needs. In this paper, we are mainly interested in the question of how and to what extent a PIP can exploit the specification flexibilities of a VNet to save on embedding resources.

VNet Embeddings. We represent the substrate network as a graph $G_S = (V_S, E_S)$ where V_S represents the substrate nodes and E_S represents the substrate links. Also a VNet request comes in the form of a graph, represented as $G_V = (V_V, E_V)$ (V_V are the virtual nodes, E_V are the virtual links). This VNet needs to be *embedded* on G_S: each virtual node of G_V is mapped to a substrate node, and each virtual link is mapped to a *path* (or a *set of paths*). Figure 1 illustrates an example.

For simplicity and to focus on the specificity, throughout this paper, we will study undirected and unweighted VNets only, i.e., we assume that each node $v \in V_V$ and each link $e \in E_V$ has a unit capacity. Our approach can easily be extended to weighted VNets.

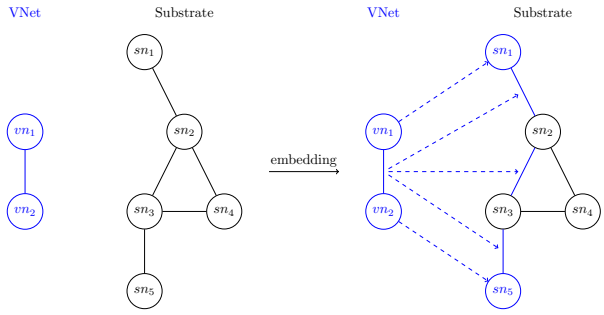

Fig. 1. Visualization of a VNet embedding: the 2-node VNet on the left is mapped to the 5-node substrate network. Each virtual node of the VNet maps to a substrate node, and for the realization of the virtual link resources are allocated along a path.

We will consider the following embedding cost model.

Definition II.1 (Embedding Costs)**.** *Let* $\Pi(e) = \{\pi_1, \pi_2, \ldots\}$ *for some* $e \in E_V$ *denote the set of substrate paths over which*

e is realized (i.e., embedded). Let $\omega(\pi)$ for some $\pi \in \Pi(e)$ denote the fraction of flow over path π, and let $\lambda(e)$ denote the length of e in terms of number of hops. The cost of embedding a VNet $G_V = (V_V, E_V)$ on a substrate G_S is defined as

$$\textit{Cost} = \sum_{e \in E_V} \sum_{\pi \in \Pi(e)} \omega(\pi) \cdot \lambda(e)$$

In other words, the allocation cost is simply the weighted distance of the different paths used by the virtual edges.

Specifying VNets. VNets can be specified in several ways, and the results in this paper do not depend on any specific language. However, for a concrete example, we refer the reader to the detailed description of the resource description language used in our prototype [24].

III. Optimal VNet Allocation

In order to compute optimal VNet embeddings and to exploit the specification flexibilities, we developed the *FlexMIP* algorithm. FlexMIP is a *Mixed Integer Program (MIP)*, and is described in Figure 3. It is a compact variation of the algorithm used in our network virtualization prototype architecture [25].

One difficulty of the FlexMIP is that, unlike in the formulation of classic multi-commodity flow problems, the endpoints of virtual links are variables as well and subject to optimization. Interestingly, the problem can still be formulated with linear constraints only, and even without using *big-M constraints* which can be harmful for the performance of a branch-and-bound algorithm (due to the inefficient relaxation). (See [14] for more technical background on these aspects.)

Concretely, the structure of FlexMIP is as follows. The constants describe the substrate and the VNets topologies including their capacities and requirements respectively. Only one full-duplex link can exist between two nodes. There is only one set of substrate vertices/edges (V_s, E_s) and exactly one set of virtual vertices/edges per VNet request $(V_v(r), E_v(r))$. The migration cost for virtual nodes is given by *Migration Cost*. To distinguish between VNnet node specifications, we use the constants *Possible Placements*. If a substrate node fulfills all VNet node specifications, it is a possible placement; otherwise the VNet node cannot be embedded there. *Node Mapping* and *Flow Allocation* are embedding variables. A VNet link can be split in flows and mapped on several links while a VNet node can only be mapped to exactly one substrate node. *Each Node Mapped* ensures that all VNets are embedded completely on a suitable substrate node and the *Feasible* constraint is guarantees the resources on these nodes. Communication in both directions is provided and the capacity boundary on each link is formulated by *Realize Flows*. *Guarantee Link Realization* guarantees that for each VNet link the needed resources are allocated on all substrate links. Outgoing traffic is positive and incoming traffic negative. For the substrate node where a VNet link starts, the outgoing traffic has to match exactly the *VEdge Demand*; at the substrate node where the link terminates, the incoming traffic has to match the negative *VEdge Demand*. The links in-between are forced to preserve the traffic and therefore must have the same amount of incoming and outgoing traffic concerning one VNet link. The objective function is to minimize the embedding cost which in our case is the *Flow allocation*.

IV. The Price of Specificity

To study the Price of Specificity, we consider the following model. We assume that each substrate node $v_S \in V_S$ of $G_S = (V_S, E_S)$ can be described by a set of k *properties* $P = \{p_1, \ldots, p_k\}$, e.g., the geographical *location* (e.g., `data center in Berlin, Germany`), the hardware *architecture* (e.g., `64-bit SPARC`), the *operating system* (e.g., `Mac OS X`), the virtualization *technology* (e.g., `Xen`), and so on. The specific property $p \in P$ of v_S can be realized as a specific *base type* $t_{v_S}(p)$. For example, the set $T(p)$ of base types for an operating system property $p \in P$ may be $T(p) = \{$`Mac OS X`, `RedHat 7.3`, `Windows XP`$\}$. (Note that if not every substrate node features each property, a dummy type *not available* can be used.)

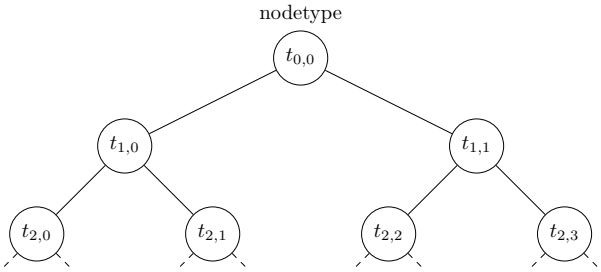

Fig. 2. Example of a binary hierarchical specification: the VNet node type of a property can be chosen from different specificities. A type of a certain layer allows an embedding on a substrate node with a type of a descendant node. The types for each property of the substrate nodes are always chosen from the leaves.

Similarly, the VNet $G_V = (V_V, E_V)$ comes with a certain *specification* of allowed types. While the substrate nodes V_S naturally are of specific base types, VNet specifications can be more vague. For example, the types $T(p)$ can often be described *hierarchically* as seen in Figure 2: the location `Berlin` can more generally be described by `Germany`, `Europe`, or `?` (*don't care*); or instead of specifying the operating system `Mac OS X`, a VNet may simply require a `Mac`.

Concretely, we assume that each virtual node v_V comes with a specification $\text{spec}(v_V) \subseteq T(p_1) \times \ldots \times T(p_k)$ of allowed type combinations for the different properties. The substrate node v_S to which v_V is embedded must fulfill at least one such type combination.

Definition IV.1 (Valid Embedding)**.** *Let $t(v_S) = \times_{p \in P} t_{v_S}(p)$ denote the vector of types of substrate node v_S. A VNet $G = (V_V, E_V)$ embedding is valid if for each virtual node $v_V \in V_V$, it holds that v_V is mapped to a node v_S with $t(v_S) \in \textit{spec}(v_V)$. In addition, node and link capacity constraints are respected.*

Of course, a user must not specify $t(v_S)$ explicitly by enumerating all allowed combinations: this set only serves for

Constants:

Requests : R

Substrate Vertices : V_s | Virtual Vertices : $V_\mathrm{v}(r), r \in R$

Substrate Edges : $E_\mathrm{s} : V_\mathrm{s} \times V_\mathrm{s}$ | Virtual Edges : $E_\mathrm{v}(r) :\to V_\mathrm{v}(r) \times V_\mathrm{v}(r), r \in R$

Unique : $uni\_check_\mathrm{s} : \forall (s_1, s_2) \in E_\mathrm{s} : (s_2, s_1) \notin E_\mathrm{s}$ | Unique : $uni\_check_\mathrm{v} : \forall r \in R, (v_1, v_2) \in E_\mathrm{v}(r) : (v_2, v_1) \notin E_\mathrm{v}(r)$

SNode Capacity : $snc(s) \to \mathbb{R}^+, s \in V_\mathrm{s}$ | VNode Demand : $vnd(r, v) \to \mathbb{R}^+, r \in R, v \in V_\mathrm{v}(r)$

SLink Capacity : $slc(e_\mathrm{s}) \to \mathbb{R}^+, e_\mathrm{s} \in E_S$ | VEdge Demand : $vld(r, e_\mathrm{v}) \to \mathbb{R}^+, r \in R, e_\mathrm{v} \in E_\mathrm{v}(r)$

Edges-Reverse : $ER_\mathrm{s} : \forall (s_1, s_2) \in E_\mathrm{s} \exists (s_2, s_1) \in ER_\mathrm{s} \wedge |E_\mathrm{s}| = |ER_\mathrm{s}|$ | Edges-Bidirectional : $EB_\mathrm{s} : E_\mathrm{s} \cup ER_\mathrm{s}$

Migration Cost : $mig\_cost(r, v, s) \to \mathbb{R}^{+\ |V_\mathrm{v}(r)| \times |V_\mathrm{s}|}, r \in R, v \in V_\mathrm{v}(r), s \in V_\mathrm{s}$

Possible Placements : $place(r, v, s) \to \{0, 1\}^{|V_\mathrm{v}(r)| \times |V_\mathrm{s}|}, r \in R, v \in V_\mathrm{v}(r), s \in V_\mathrm{s}$

Variables:

Node Mapping : $n\_map(r, v, s) \in \{0, 1\}, r \in R, v \in V_\mathrm{v}(r), s \in V_\mathrm{s}$

Flow Allocation : $f\_alloc(r, e, eb) \geq 0, r \in R, e \in E_\mathrm{v}(r), eb \in EB_\mathrm{s}$

Constraints:

Each Node Mapped : $\forall r \in R, v \in V_\mathrm{v}(r) : \sum_{s \in V_\mathrm{s}} n\_map(r, v, s) \cdot place(r, v, s) = 1$

Feasible : $\forall s \in V_\mathrm{s} : \sum_{r \in R, v \in V_\mathrm{v}(r)} n\_map(r, v, s) \cdot vnd(r, v) \leq snc(s)$

Guarantee Link Realization : $\forall r \in R, (v_1, v_2) \in E_\mathrm{v}(r), s \in V_\mathrm{s} \sum_{(s, s_2) \in V_\mathrm{s} \times V_\mathrm{s} \cap EB_\mathrm{s}} f\_alloc(r, v_1, v_2, s, s_2) -$
$\sum_{(s_1, s) \in V_\mathrm{s} \times V_\mathrm{s} \cap EB_\mathrm{s}} f\_alloc(r, v_1, v_2, s_1, s) = vld(r, v_1, v_2) \cdot (n\_map(r, v_1, s) - n\_map(r, v_2, s))$

Realize Flows : $\forall (s_1, s_2) \in E_\mathrm{s} \sum_{r \in R, (v_1, v_2)} f\_alloc(r, v_1, v_2, s_1, s_2) + f\_alloc(r, v_1, v_2, s_2, s_1) \leq slc(s_1, s_2)$

Objective function:

Minimize Embedding Cost : $min : \sum_{r \in R, (v_1, v_2) \in E_\mathrm{v}(r), (s_1, s_2) \in E_\mathrm{s}} f\_alloc(r, v_1, v_2, s_1, s_2) + f\_alloc(r, v_1, v_2, s_2, s_1)$

Fig. 3. Embedding constants, variables, constraints and the objective function of the *FlexMIP*. Explanations are given in Section III.

formal presentation. Rather, a user can specify the types of VNet nodes with an arbitrary resource description language, and use *white lists* (e.g., only `Mac`) or *black lists* (not on `Sparc`), or more complex logical formulas.

The question studied in this paper revolves around the tradeoff of the VNet specificity and the embedding cost.

Definition IV.2 (Price of Specificity (PoS) ρ)**.** *Given a VNet G_V, let Cost$_0$ denote the embedding cost (cf Definition II.1) of G_V in the absence of any specification constraints, and let Cost$_\sigma$ denote the embedding cost under a given specificity $\sigma(G_V)$. Then, the* Price of Specificity $\rho(G_V)$ *(or just ρ) is defined as $\rho =$ Cost$_\sigma$/Cost$_0$.*

Note that the Price of Specificity ρ depends on the specific embedding algorithm. In the following, we do not assume any specific embedding algorithm, but just use the placeholder ALG to denote an arbitrary state-of-the-art VNet embedding algorithm. (In the related work section, Section VII, we will review some candidates from the literature.) However, in the simulations, we will use an optimal algorithm *FlexMIP* that minimizes resources.

Although our definition of the Price of Specificity is generic and does not depend on a particular definition of specificity, for our evaluation, we will use the following metric.

Definition IV.3 (Specificity σ)**.** *The* specificity $\sigma(v_V)$ *of a virtual node v_V captures how many alternative type configurations are still allowed by a specification compared to a scenario where all configurations are allowed. Formally, we define $\sigma(v_V)$ as the percentage of lost alternatives: $\sigma(v_V) = 1 - (|t(v_S)| - 1)/(|T(p_1) \times \ldots \times T(p_k)| - 1)$. The* specificity $\sigma(G_V)$ *of a VNet $G_V = (V_V, E_V)$ is defined as the average specificity of its nodes $v_V \in V_V$: $\sigma(G_V) = \sum_{v_V \in V_V} \sigma(v_V)/|V_V|$.*

Note that $\sigma(G_V) \in [0, 1]$, where $\sigma(G_V) = 0$ and $\sigma(G_V) = 1$ denote the minimal and the maximal specificity, respectively. We will focus on scenarios where $|T(p_1) \times \ldots \times T(p_k)| > 1$.

V. Evaluation

This section studies the Price of Specificity (PoS) in different scenarios. In order to avoid artifacts resulting from approximate or heuristic embeddings, we consider optimal embedding solutions only.

A. Setup

In our evaluation, if not stated otherwise, we will focus on the following default scenario. We consider two different properties $P = \{p_1, p_2\}$ with four different types each ($T(p_1) = \{t_1^1, t_2^1, t_3^1, t_4^1\}$, $T(p_2) = \{t_1^2, t_2^2, t_3^2, t_4^2\}$). By default we do not allow to migrate already embedded VNets. The substrate node types are chosen independently at random from the base types such that each type occurs equally often (up to rounding), and the virtual node types are chosen independently uniformly at random according to the specificity level.

Concretely, we study five different degrees of specificity (in increasing order of specificity): (1) *all* types are allowed (no restrictions, i.e., specificity $\sigma = 0$); (2) only two types (either $\{t_1^1, t_2^1\}$ or $\{t_3^1, t_4^1\}$) are allowed for $T(p_1)$, but all types of $T(p_2)$ (specificity $\sigma \approx 0.533$); (3) only two types (either $\{t_1^1, t_2^1\}$ or $\{t_3^1, t_4^1\}$) are allowed for $T(p_1)$ and only

either $\{t_1^2, t_2^2\}$ or $\{t_3^2, t_4^2\}$ for $T(p_2)$ (specificity $\sigma = 0.8$); (4) only one type is allowed for $T(p_1)$ and only two types (either $\{t_1^2, t_2^2\}$ or $\{t_3^2, t_4^2\}$) for $T(p_2)$ (specificity $\sigma \approx 0.933$); (5) only one type is allowed for each property $T(p_1)$ and $T(p_2)$ (i.e., $\sigma(v_V) = \{t_1^1, t_1^2\}$ for all nodes $v_V \in V_V$, specificity $\sigma = 1$).

Furthermore, we assume that the nodes in the substrate all have a capacity of one unit, and that the links have an infinite capacity. The virtual nodes and links of the VNet have a demand of one unit (no colocation). Finally, we allow the embedding algorithm to split a virtual link into multiple paths.

Our substrate network is generated using the *Igen topology generator* [21]. Our default model uses one hundred nodes. Nodes are generated randomly and we use the clustering method `k-medoids:5` with five clusters (PoPs) based on distance. The nodes in these PoPs are access nodes which are all connected to the PoPs two backbone nodes. These backbone nodes are picked geographically as the most central ones among the access nodes within a cluster. The backbone topology is built by using a Delaunay triangulation connecting a backbone node with other backbone nodes next to it. Thereby the connectivity is preserved since the triangulation includes the minimal spanning tree and alternative paths are created to guarantee redundancy [15].

As for the VNets we will focus on master-slave (i.e., star) topologies. In the following, we will refer to a star with one center node and $x - 1$ leaves as an *x-star*. In most cases we study 4- or 5-stars.

B. Impact of Substrate Size and Load

We first study the impact of the substrate size and load and consider two different scenarios: (1) an empty substrate network, and (2) a scenario where the substrate nodes already host some virtual nodes. In both scenarios the arriving VNet is a 5-star.

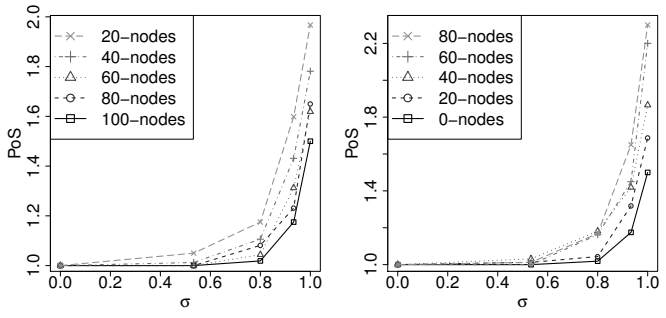

Fig. 4. *Left:* Impact of substrate size on the PoS. Each substrate was created with the Igen topology generator having five PoPs and two backbone nodes per PoP. *Right:* Impact of load on the PoS. Each scenario is based on the 100 nodes Igen substrate with different numbers of fully utilized nodes. The nodes are chosen uniformly at random.

Figure 4 (*left*) plots the PoS as a function of substrate sizes for the empty substrate scenario. As expected, since a larger network offers more embedding options, it is more likely that a low-cost embedding can be found, and the PoS is lower on larger substrates. At around one hundred nodes, the embedding is almost perfect for a VNet with specificity $\sigma = 0.8$ resulting in a PoS of nearly one whereas the PoS for the 20-nodes substrate is almost 1.2. At a specificity $\sigma = 1$ the PoS is nearly two in the 20 nodes substrate scenario implying that we need roughly twice as much link resources than actually stated in the VNet requirements. As to be expected, the larger substrates have a smaller PoS and the absolute difference is increasing with the specificity.

Let us now consider a scenario where there is already some load on the substrate network. We study a simplified model where x substrate nodes chosen uniformly at random are set to full load, i.e., no virtual nodes can be embedded. We compare the scenario where x out of 100 nodes are already in use to a scenario where the substrate consists of $100 - x$ nodes. The results shown in Figure 4 (*right*) look similar to those of the substrate size scenario. For the $\sigma = 1$ case the more loaded substrates (40-80 nodes in use) have a higher PoS than their representatives in the substrate size scenario. Given the larger substrate with many nodes in use simultaneously, the distances between the free nodes increase. This yields longer paths allocated for embeddings, and therefore a higher PoS. For lower specificities, this is negligible due to the smaller node type variance.

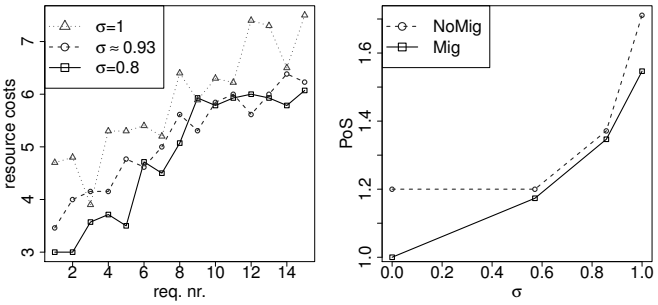

Fig. 5. *Left:* Amount of link resources needed per embedding as a function of request order. There are 15 incoming 4-star VNets with different σ on a 100-nodes Igen substrate with a substrate link capacity that allows the embedding of two VNet links. For this experiment, we disabled migration. *Right:* Impact of migration on the PoS. There are five 4-star VNets arriving over time on a 40-nodes Igen substrate with eight different substrate node types.

In order to get a better understanding of the impact of load on the PoS we studied a scenario where there are 15 VNets arriving over time. Each of them is embedded by *FlexMIP* without using migration, leading to more load over time. Figure 5 (*left*) shows the amount of link resources needed per embedding depending upon VNet arrival, and the substrate load respectively. While the first incoming VNet can always be embedded perfectly in the $\sigma = 0.8$ scenario, a higher specificity leads to 3.5 and 4.7 links on average. The link resource costs are increasing with the load and we notice the tendency of lower specificity impacts for the $\sigma = 0.8$ and $\sigma \approx 0.93$ scenarios: the curves of the resource costs are converging. Fully specified VNets are still causing more resource costs for each VNet. The impact of the load is shown in the resource costs for the 13th VNet or higher

which nearly takes twice as much resources as the first VNet. This especially occurs when the substrate is used close to its capacity. Since a substrate provider will typically try to fully utilize its infrastructure as well as trying to avoid costly embeddings, the PoS has to be understood in relation to the substrate load.

C. Impact of Migration

The load scenario from Section V-B was static in the sense that load was modeled on fixed nodes only. However, the possibility to migrate already embedded VNets to more suitable locations is one of the key advantages of the network virtualization paradigm, and hence we now attend to the use of such migrations. Migration can have very positive effects on the PoS: For instance, when a scarce type may have been blocked earlier in time by a VNet of low specificity, a migration may reduce the resource costs significantly. A better location to migrate to may also become available due to the expiration of a VNet.

We study a scenario where five 4-star VNets arrive over time on a 40 node Igen substrate with eight different substrate types. We only study runs where all five VNets have been embedded, resulting in a load of 50% on the substrate. This avoids heavily loaded substrate scenarios as well as scenarios of abundant capacity. Both scenarios naturally lead to no or only small effects of migration due to nearly optimal embeddings. Figure 5 (*right*) shows the aggregated PoS over all five VNets. We show averaged values as the embedding costs for an already embedded VNet can change over time in the migration scenario. Interestingly, migration is already effective even without specificity on the VNets (compare PoS Mig:1 - NoMig:1.2). This is due to embeddings which are initially optimal regarding resource costs but use resources that might be more effective in later embeddings, i.e., nodes with a higher degree. While the impact of migration is rather low for the following specificities, it is again recognizable for fully specified VNets.

Generally migration lowers the resource costs and hence the PoS in all our scenarios.

D. Impact of Type Distribution

The diversity of resources and especially the distribution of requested and supplied types is crucial for the PoS. We expect that in a scenario where the requested types follow the same distribution as the available substrate types, the embedding cost and hence the PoS is lower. In the following, we therefore study different probability distributions for the node types in the substrate as well as the VNets. In addition to the uniform distribution studied so far, we consider a heavy-tailed distribution: a Zipf distribution with exponent 1.2.[1]

Figure 6 (*left*) studies five different scenarios: the standard scenario where both (substrate and VNet) types are uniformly distributed, a scenario where both are heavy-tailed distributed and the mixed cases. Additionally we study a scenario where

[1]E.g., the type distribution in a 100-node substrate with 16 node types is $[28, 12, 12, 8, 8, 5, 5, 5, 3, 3, 2, 2, 2, 2, 2, 1]$.

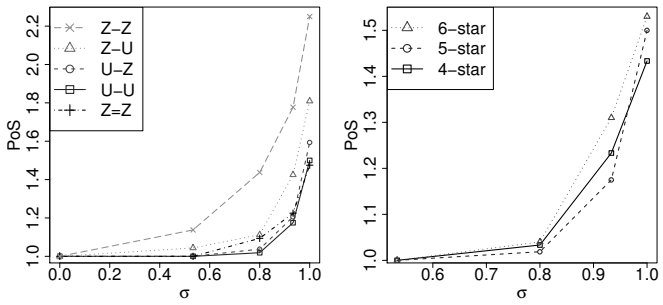

Fig. 6. *Left:* Impact of different type distributions on the PoS. We compare all combinations between uniformly and heavy-tailed distributed types as well as a scenario where two heavy-tailed distributions have inverse type frequencies (i.e., the most frequent type becomes the least frequent type). As a heavy-tailed distribution Zipf was chosen with exponent 1.2. (legend: Z=Zipf, U=uniform, Substrate-VNet)*Right:* Impact of different VNet sizes on the PoS via 4-, 5- and 6-star VNets embedded on the 100-node Igen substrate.

the substrate types are Zipf distributed and the VNet types are *inversely* Zipf distributed in the sense that the least frequent type is the most frequent one in the other distribution. As to be expected, we see that the highest PoS is obtained in the scenario with contrary Zipf distributions. While this scenario is very different from the one with next highest PoS (S-zipf V-uni), the case where both types follow the same Zipf distribution only marginally differs from the scenario where both types are uniformly distributed. Another interesting observation is that the PoS which is obtained in the scenario where the substrate node types are Zipf distributed and the VNet types are uniformly distributed is higher than the PoS of the opposite combination. This is due to the fact that in a Zipf distributed substrate, there are many node types from which only three or less nodes exist whereas the types are requested at equal proportions. Therefore, the possibility that a scarce and hence relatively far away type is requested is high. The opposite distribution has a lower PoS because even the possibility to have at least two nodes from the most common node type in the request is below 50% and around 20% for three or more nodes. Since there are approximately six nodes per node type in the uniformly distributed substrate, the distances are not that large, resulting in a lower PoS for this scenario. The same argument holds for the only marginal difference between the scenarios where both distributions are Zipf and where both distributions are uniform.

One takeaway from these results is that a specialization of the substrate entails the risk of a high PoS if the demand does not perfectly fit.

E. Impact of VNet Topology

Section V-B has shown how the size of the substrate impacts the PoS, and we expect a similar impact of changing the VNet size. Figure 6 (*right*) shows the PoS for different sizes of the VNets regarding the number of nodes. For small specificities ($\sigma \leq 0.8$) the PoS is almost constant. This is due to the relatively (regarding the amount of types and the size of the

VNets) large substrate which is robust against these small limitations on the embeddings. The flexibility can still be maintained and an optimal allocation can be found. For higher specificities, we see similar PoSs for the different star sizes. There is a tendency that for increasing specificity the larger stars have a higher PoS than the smaller ones but those differences are rather small. Therefore we investigate also the effect of changing the topology of a 4-star by adding links.

Figure 7 (*left*) shows a scenario where additional leaves of the 4-star are connected one by one. The figure reveals the interesting part of the higher specified VNets only. All scenarios with additional links have a higher or at least equal PoS than the standard star scenario, since the complexity to embed these topologies is increasing. Nevertheless the difference between the most complex topology (4-star+3) and the 4-star is relatively low, with an absolute difference always around 0.2.

We find that the modification of the VNet topologies by adding nodes or links does not have a big impact on the PoS. However, note that this conclusion might be different if the load on the substrate is increased.

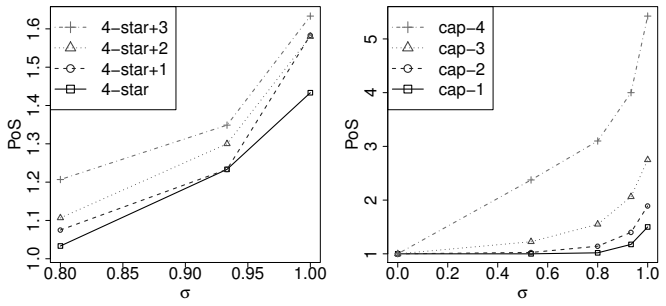

Fig. 7. *Left:* Impact of additional links on the PoS: we connect the leaf nodes with each other, and 4-star+1 stands for the scenario with a 4-star VNet and two of its leaves connected. *Right:* Impact of substrate node capacities on the PoS. The figure shows the result from embedding a 5-star VNet while changing the node capacity of the underlying substrate from one VNet node per substrate node up to four nodes per substrate node.

F. Impact of Capacity

Network virtualization allows the substrate provider to embed VNets according to its needs, e.g., to minimize link resource costs in the case of our algorithm *FlexMIP*. The capacities in the infrastructure will not fit exactly the specifications of the VNets in general. Therefore the embedding of several VNet nodes on one substrate node promises lower link resource costs while the restriction on link capacities may yield higher costs.

Figure 7 (*right*) shows a scenario where the node capacities are increased up to four units. The curves show a similar trend and there is a higher PoS noticeable with increasing node capacity. This is a consequence of using the substrate nodes at full capacity without VNet node specificities whereas a higher specificity prevents this due to the different VNet node types. A PoS is also observed on specificities where there was no additional costs before. At a specificity of $\sigma \approx 0.533$, the PoS for the scenario with a node capacity of four is already higher than the PoS of a fully specified VNet in a scenario with node capacity one.

G. An Out-Sourcing Scenario

While our previous simulations may also describe geographical types (as a property), in this section we examine a concrete geographic use-case more closely. Generally we now change the experiments in a way that some VNet nodes assume a fixed position in the substrate. This use-case can for example represent a company which owns certain servers and runs its own network, but out-sources computation or storage to the cloud. While the locations of the infrastructure in the corporate network is given, the out-sourced resources may be specified at different granularity.

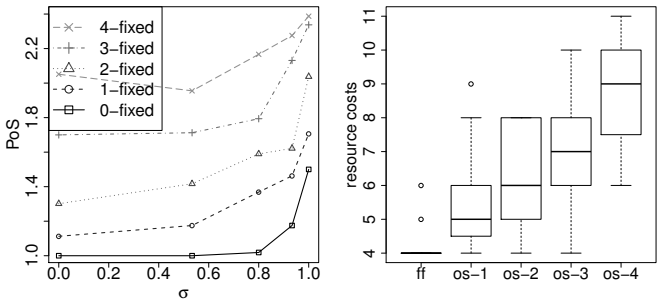

Fig. 8. *Left:* Impact of different degree of out-sourcing on the PoS. Embedding of a 5-star VNet while changing the number of fixed corporate nodes in it, e.g. three fixed nodes lead to only two free nodes affected by the specificity. Note that the specificity in those scenarios is referred only to the free nodes while the PoS is calculated regarding the 0-fixed scenario without specificity (causing higher PoSs than 1 even for $\sigma = 0$). *Right:* Boxplot showing the variation for the fully flexible scenario (ff - zero nodes fixed) and four different out-sourcing scenarios (os1 - one fixed node, etc.) for a specificity of $\sigma \approx 0.53$.

Figure 8 (*left*) shows the impact of the amount of fixed corporate nodes on the PoS. The 5-star is the incoming VNet basis while we vary the amount of fixed nodes from one to four. In all cases the PoS is calculated regarding the 0-fixed scenario while σ is only referring to the specificity of the remaining free nodes. Noticeable on the first sight is that even for a specificity of $\sigma = 0$ for the remaining nodes, the PoS is in all cases higher than one. Especially with three or four nodes fixed (meaning only two and one additional nodes to be embedded) the PoS reaches ~ 1.7 and ~ 2.05 respectively. With an increasing specificity the PoSs of the different scenarios are converging. This is due to the declining effect of the positioning of the outsourced nodes and the increasing effect of the specificity. Keep in mind that this effect might be different in different use cases, e.g., if the fixed nodes are positioned next to each other and not picked uniformly at random of the substrate.

The impact of the fixed node locations can also be understood as a function of the link resource costs of the out-sourcing scenarios. Figure 8 (*right*) shows a boxplot compar-

ing these costs with each other for a specificity of $\sigma \approx 0.53$. In the fully flexible scenario the VNet is almost always embedded with the smallest possible amount of link resources while an increasing number of fixed nodes also increases the variance as well as the used link resources. Nonetheless, an optimal embedding regarding the link resource costs can be achieved even in the out-sourcing scenarios provided an appropriate random placement of the fixed nodes, which can be observed in the scenarios with one to three fixed nodes.

VI. Excursion: Use of Migration

An intriguing question regards the impact of migration on the Price of Specificity. At first sight it may seem that migration can only be beneficial (see also our simulation results in Section V-C). However, we will show that this is only true for scenarios where links have unbounded capacities. Otherwise, there exist situations where migration can be harmful. We will refer to this phenomenon as the *Migration Paradox*.

In order to study the impact of migration on the PoS we extend the Definition IV.2 towards several embedded VNets.

Definition VI.1 (Price of Specificity ρ for multiple VNets)**.** *Given a sequence of VNets $\mathcal{G}_k = (G_V^1, ..., G_V^k)$, the PoS $\rho(\mathcal{G}_k)$ is defined as the average PoS over all VNets $\rho(\mathcal{G}_k) = \sum_{i=1}^{k} \rho(G_V^i)/k$.*

Given the PoS definition for a sequence of VNets, we will make the following assumptions: (1) We focus on embedding algorithms ALG which greedily accept all incoming VNets if possible, while trying to minimize the corresponding embedding costs. (2) Migration itself is only causing no/negligible costs regarding the PoS. (3) The substrate links have unbounded link capacities.

Certainly, all scenarios satisfying these assumptions can only benefit from migration.

Theorem VI.2. *In scenarios satisfying Assumptions (1)-(3), migration can only decrease the overall PoS of the embedded VNets, meaning $\rho_1(\mathcal{G}_k) \leq \rho_0(\mathcal{G}_k)$ for any sequence of VNets $\mathcal{G}_k$, with ρ_1 representing the PoS for the migration scenario and ρ_0 representing the one without.*

Proof: Let $\mathcal{G}_k$ be a sequence of VNets (with $\mathcal{G}_k = (G_V^1, ..., G_V^k), k \in \mathbb{N}$) requested at the substrate one after another. Note that due to Assumption 1 and 3, an identical order of incoming VNet requests results in exactly the same VNets that will be accepted in both scenarios, meaning that there are valid embeddings for the same requests (cf Definition IV.1). Since only unavailable node types can cause a VNet to be rejected, it is sufficient to show that for an arbitrary sequence of embedded VNets the link resource usage is not exceeding those of the scenario without migration. The notation $\mathcal{G}_{k'},\ k' \leq k$ describes such a k'-tuple of embedded VNets from an incoming sequence $\mathcal{G}_k$.

The proof is by induction over the number of requests. Take an arbitrary sequence $\mathcal{G}_{k'} = (G_V^1, ..., G_V^{k'})$ of embeddable VNets. We will show that $\rho_1(\mathcal{G}_{k'}) \leq \rho_0(\mathcal{G}_{k'})$ for each $k \in \mathbb{N}$. For $k = 1$ the claim holds: The only incoming VNet G_V^1 of $\mathcal{G}_1$ will always be embedded equally in both scenarios. There are no other VNets that can be migrated, hence ALG embeds this VNet in both scenarios identically. Therefore $\rho_1(\mathcal{G}_1) \leq \rho_0(\mathcal{G}_1)$ is satisfied.

For the *induction step* ($k > 1, k \in \mathbb{N}$), assume that $\rho_1(\mathcal{G}_k) \leq \rho_0(\mathcal{G}_k)$ holds for an arbitrary $k \in \mathbb{N}$. We show that the embedding of an arbitrary additional VNet G_V^{k+1} still satisfies the claim $\rho_1(\mathcal{G}_{k+1}) \leq \rho_0(\mathcal{G}_{k+1})$. We know that $\rho_1(\mathcal{G}_k) \leq \rho_0(\mathcal{G}_k)$, and an additional VNet request cannot yield higher costs without migration, as the embedding configuration can always be migrated to any possible cheaper configuration. The migration will not increase the PoS due to Assumption 2 and therefore a lower PoS may be achieved. ■

However, note that if link capacities are limited, Theorem VI.2 no longer holds. Figure 9 shows an example where migration can lead to larger resource costs. It shows a simple substrate with a line-topology and link capacities of one. There are only two different types of nodes in the substrate, namely *A* and *B* and a small VNet with two connected nodes of type *B* embedded. This VNet has to be migrated in order to embed a VNet with two connected nodes of type *A*. Therefore the new embedding is using all of the remaining link capacity and prevents the embedding of additional VNets.

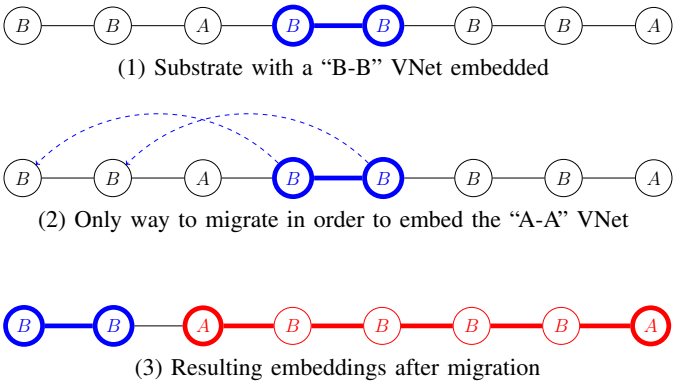

Fig. 9. Blocked resources due to enabled migration. After migrating the "B-B" VNet and embedding the "A-A" VNet no link capacities are available for further VNet embeddings.

Thus, we have the following result.

Theorem VI.3. *Generally, there are scenarios where migration can* increase *the PoS.*

Therefore it is necessary to have a proper access control even though the PIP is able to migrate certain VNets.

VII. Related Work

At the heart of network virtualization lies the idea of making more efficient use of a given infrastructure and its resources by sharing it among multiple VNets. The important problem of finding good VNet embeddings [20] has been studied intensively, both from offline [17] and online perspectives [4], [12], by focusing on bandwidth constraints only [13], by pursuing heuristic approaches without admission control [27], or by employing simulated annealing techniques [22]. The

survey by Belbekkouche [5] provides a nice overview of allocation and embedding algorithms. Lischka and Karl [16] present an embedding heuristic that uses backtracking and aims at embedding nodes and links concurrently for improved resource utilization. A mixed integer program for the embedding of certain types of VNets is formulated in [7]; the formulation has been extended to support migration and reconfiguration in [23]. Since the general embedding problem is computationally hard, most of the literature is on heuristical or approximative algorithms. An interesting perspective is taken by Yu et al. [19] who advocate to rethink the design of the substrate network to simplify the embedding which makes it computationally tractable; for instance, they allow to split a virtual link over multiple paths and perform periodic path migrations. The general embedding problem is also related to network design [26], virtual circuit planning [3], or minimal linear arrangement [6].

Our contribution is orthogonal to this line of research. In fact, the Price of Specificity could be studied for each embedding algorithm reviewed above. In order to focus on the main properties of the Price of Specificity, we use a an optimal embedding approach for our evaluation, and ask the question how the VNet specification effects the cost. A short paper of this work appeared at UCC [18].

In the context of the relatively new concept of network virtualization itself, not much work on economical aspects exists yet. Courcoubetis et al. [10] identify incentive issues arising in the management of virtual infrastructures and show that well-designed policies are mandatory to prevent agents from contributing less resources than is desirable. PolyViNE [8] is a decentralized, policy-based inter-domain embedding protocol ensuring competitive prices for service providers. In more general context, Antoniadis et al. [2] employ coalitional game theory to study how participants should share the value of federation in virtualized infrastructures (in the context of ISP interconnections, peer-to-peer systems, the Grid, or cloud computing).

VIII. Conclusion

While today, we have a fairly good understanding of how to realize the vision of virtual networks (cf, e.g. [11]), surprisingly little is known about economical implications. We consider our paper as a first step to shed light on the impact of virtual network specification flexibility on the embedding cost. We hope that our approach provides a means to reason about policies for VNet pricing and embedding, especially in competitive markets where infrastructure providers operate within small budget margins.

Acknowledgments. This work was supported in part by the EU FP7 projects BigFoot (FP7-ICT-317858), CHANGE (FP7-ICT-257422) and OFELIA. We would also like to thank Gregor Schaffrath and Carlo Fürst for many discussions.

References

[1] Going with the flow: Google's secret switch to the next wave of networking (wired web site), April 2012. http://www.wired.com/wiredenterprise/2012/04/going-with-the-flow-google/all/1.

[2] P. Antoniadis, S. Fdida, T. Friedman, and V. Misra. Federation of virtualized infrastructures: sharing the value of diversity. In *Proc. 6th CoNEXT*, 2010.

[3] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *Proc. IEEE FOCS*, 1993.

[4] N. Bansal, K.-W. Lee, V. Nagarajan, and M. Zafer. Minimum congestion mapping in a cloud. In *Proc. ACM PODC*, pages 267–276, 2011.

[5] A. Belbekkouche, M. Hasan, and A. Karmouch. Resource discovery and allocation in network virtualization. *IEEE Communications Surveys Tutorials*, (99):1–15, 2012.

[6] M. Charikar, K. Makarychev, and Y. Makarychev. A divide and conquer algorithm for d-dimensional arrangement. In *Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 541–546, 2007.

[7] K. Chowdhury, M. R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. In *Proc. INFOCOM*, 2009.

[8] M. Chowdhury, F. Samuel, and R. Boutaba. PolyViNE: Policy-based virtual network embedding across multiple domains. In *Proc. 2nd ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architecture (VISA)*, 2010.

[9] M. K. Chowdhury and R. Boutaba. A survey of network virtualization. *Elsevier Computer Networks*, 54(5), 2010.

[10] C. Courcoubetis and R. R. Weber. Economic issues in shared infrastructures. In *Proc. ACM Workshop on Virtualized Infrastructure Systems and Architectures (VISA)*, pages 89–96, 2009.

[11] D. Drutskoy, E. Keller, and J. Rexford. Scalable network virtualization in software-defined networks. *Internet Computing, IEEE*, PP(99):1, 2012.

[12] G. Even, M. Medina, G. Schaffrath, and S. Schmid. Competitive and deterministic embeddings of virtual networks. In *Proc. 13th ICDCN*, 2012.

[13] J. Fan and M. H. Ammar. Dynamic topology configuration in service overlay networks: A study of reconfiguration policies. In *Proc. INFOCOM*, 2006.

[14] I. Griva, S. G. Nash, and A. Sofer. *Linear and Nonlinear Optimization (2nd ed.)*. Society for Industrial Mathematics, 2009.

[15] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot. Feasibility of ip restoration in a tier 1 backbone, 2004.

[16] J. Lischka and H. Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proc. VISA*, pages 81–88, 2009.

[17] J. Lu and J. Turner. Efficient mapping of virtual networks onto a shared substrate. In *Technical Report, WUCSE-2006-35, Washington University*, 2006.

[18] A. Ludwig, S. Schmid, and A. Feldmann. The price of specificity in the age of network virtualization. In *Proc. 5th IEEE/ACM UCC*, 2012.

[19] J. R. M. Yu, Y. Yi and M. Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review*, 38(2):17–29, Apr 2008.

[20] B. Monien and H. Sudborough. Embedding one interconnection network in another. In *Computational Graph Theory*, 1990.

[21] B. Quoitin, V. V. den Schrieck, P. Franois, and O. Bonaventure. Igen: Generation of router-level internet topologies through network design heuristics. In *Proc. 21st International Teletraffic Congress (ITC)*, 2009.

[22] R. Ricci, C. Alfeld, and J. Lepreau. A solver for the network testbed mapping problem. *SIGCOMM CCR*, 33(2):65–81, 2003.

[23] G. Schaffrath, S. Schmid, and A. Feldmann. Optimizing long-lived cloudnets with migrations. In *Proc. 5th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2012.

[24] G. Schaffrath, S. Schmid, I. Vaishnavi, A. Khan, and A. Feldmann. A resource description language with vagueness support for multi-provider cloud networks. In *Proc. ICCCN*, 2012.

[25] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy. Network virtualization architecture: Proposal and initial prototype. In *Proc. VISA*, 2009.

[26] D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15:708–717, 1995.

[27] Y. Zhu and M. H. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *Proc. INFOCOM*, 2006.