Multi-Domain Virtual Network Embedding with Limited Information Disclosure

David Dietrich Amr Rizk Panagiotis Papadimitriou
Institute of Communications Technology, Leibniz Universität Hannover, Germany
{*firstname.lastname*}@*ikt.uni-hannover.de*

***Abstract*—The ever-increasing need to diversify the Internet has recently revived the interest in network virtualization. Network virtualization carries a significant benefit to Service Providers, as it enables the deployment of network services within customized virtual networks (VN) that offer performance and reliability guarantees. VN embedding across multiple substrate providers creates the need for a layer of indirection which is fulfilled by VN Providers. VN Providers are expected to have very limited knowledge of the physical infrastructure, since substrate providers do not disclose their network topology and resource information. This entails significant implications on resource discovery and assignment.**

In this paper, we study the challenging problem of multi-domain VN embedding with limited information disclosure (LID). In this context, we discuss the visibility of VN Providers on substrate network resources and question the suitability of topologies for VN request specifications. Our main contributions are as follows: (i) we present a traffic matrix based VN embedding framework that enables VN request partitioning under LID, and (ii) we conduct a feasibility study on VN embedding with LID compared to a "best-case" scenario where all information is available to VN Providers.

I. Introduction

Over the last years, the Internet has seen the emergence of many new network applications and services, such as video conferencing, IPTV, and online gaming. Unlike the evolution and the advent of new network services, the Internet architecture has remained almost unchanged, providing merely "best-effort" delivery. Existing technologies for Quality of Service, robust routing and security experience difficulties in transcending organization or enterprise boundaries. This has an adverse impact on the deployment of network services across wide areas, especially when network services pose stringent and varying requirements in terms of throughput, delay, packet loss or security.

The ever-increasing need to diversify the Internet has revived the interest in network virtualization [11]. Recent advances on server and router virtualization (e.g., [7], [5], [12], [22], [8]) satisfy the various requirements (e.g., high packet forwarding performance, isolation) for the concurrent deployment and operation of service-tailored network slices on top of shared physical infrastructures [19], [26]. In this context, both Service Providers (SPs) and Physical Infrastructure Providers (InPs) benefit from network virtualization. SPs can deploy network services within customized virtual networks (VNs) that offer performance and reliability guarantees. For InPs, network virtualization improves resource efficiency and reduces the operational (OPEX) and technology investment costs (CAPEX).

Wide-area VN deployment requires the ability to provision and operate VNs across multiple InPs. This creates the need for a layer of indirection, which is fulfilled by the so-called VN Providers (VNPs) that are responsible for the discovery, selection and allocation of virtual resources from multiple InPs. Essentially, VNPs assemble the resources allocated from the InPs into a functional network which can be subsequently configured and operated by the SP. The role and the required control interfaces for VNPs are exemplified in [19], while a similar layer of indirection also exists in Cabernet [26] and GENI [3].

VN embedding across multiple substrate networks entails significant challenges for VNPs. More precisely, the various InPs, as separate administrative entities, will be reluctant to disclose any resource and network topology information to third parties. As such, VNPs are required to discover and assign resources for VN embedding with limited knowledge of the substrate topology and resource availability. Most VN embedding algorithms require full knowledge of the substrate resources and network topology, and therefore cannot be used to map VN topologies onto multiple substrate networks [10], [24], [25], [17], [13]. An existing approach for VN request partitioning among substrate providers is based on a highly abstract view of the underlay (i.e., AS-level topology without any information on peering nodes) and as such, it can generate suboptimal embeddings [15].

In this paper, we investigate the feasibility of multi-domain VN embedding with limited information disclosure (LID). First, we discuss the visibility of VNPs on substrate network resources and define a minimum level of information disclosure based on the composition of resource and network topology information that is not treated as confidential by InPs. Subsequently, we present a framework that decomposes multi-domain embedding into a set of operations allowing VNPs and InPs to process incoming VN requests based on their visibility on substrate resources. In particular, the VNP matches requested to offered resources and partitions VN requests across multiple InPs. Subsequently, InPs select the resources relying on their detailed knowledge of substrate resources. To provide the ability to embed VN requests across multiple substrate networks, we present formulations for the VN request partitioning and intra-domain resource assignment problems.

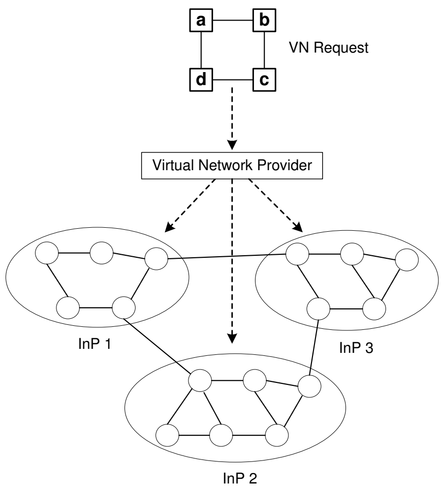


Fig. 1. VN embedding request across multiple domains.

A novel aspect of our embedding framework is the processing of VN requests, in which flow demands are represented by traffic matrices. In contrast to topology-based VN requests, VN specifications using traffic matrices enhance the flexibility in VN partitioning and mapping, while offering a higher level of abstraction to Service Providers. According to our knowledge, our work comprises the first study on the feasibility of multi-domain VN embedding with LID.

The remainder of the paper is organized as follows. In Section II, we discuss the multi-domain VN embedding problem and investigate some of its most critical aspects, such as the level of information disclosure and the specification of VN requests. Section III outlines our VN embedding framework which circumvents the difficulty of embedding VN requests across multiple InPs with LID. Section IV includes the VN partitioning and intra-domain resource assignment problem formulations. In Section V, we present our simulation results and discuss the feasibility of multi-domain VM embedding with LID. Section VI discusses related work. Finally, in Section VII, we highlight our conclusions.

II. Problem Description

In this section, we elaborate on multi-domain VN embedding with LID. In this respect, we investigate the level of substrate resource and network topology information that can be divulged without violating InP policies. Furthermore, we question the suitability of network topologies for VN request specifications and seek network abstractions that facilitate VN request partitioning and embedding.

Consider the example depicted in Fig. 1, where a SP requests the deployment of a VN consisting of virtual nodes and links associated with certain specifications. VN specifications are formulated by SPs based on the requirements of a particular service and are typically set at a high level of abstraction. The VN is mapped onto a number of InPs that offer the geographic footprint required by the SP. We rely on the multi-layer control plane architecture in [19] and particularly on the presence of VNPs, which act as brokers between SPs and InPs, carrying out resource discovery and InP selection for VN embedding. As such, VN requests are initially relayed to VNPs, which subsequently partition the VN among the participating InPs while satisfying one or multiple objectives (e.g., minimize the expenditure for the SP).

A. Information Disclosure

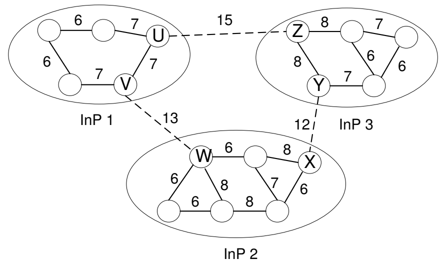
VNP's visibility on the substrate networks is critical for VN embedding. In this respect, it is important to enhance the ability of VNPs to access the resource and network topology information that is not regarded as confidential by InPs. Such level of information disclosure constitutes a prerequisite for multi-domain VN embedding, as discussed in Section III. To this end, we investigate information disclosure with respect to (i) (virtual) resource availability and (ii) substrate network topology.

Inline with cloud computing platforms such as Amazon EC2 [1], an InP classifies its resources into types and advertises them along with the associated cost. Each resource type essentially comprises all resources (i.e., nodes) with a common set of attributes (e.g., operating system, main memory, storage, virtualization technology). We anticipate, though, that InPs will not disclose the number of available instances for each offered resource type in order to conceal their resource utilization.

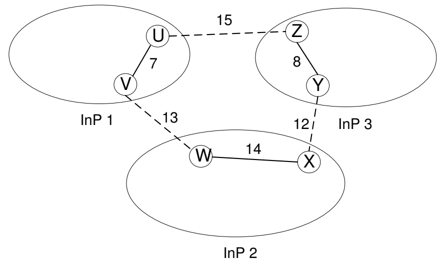
We further examine which aspects of the substrate network topology can be disclosed by InPs. For instance, a Point-of-Presence (PoP) level topology could augment VNP in estimating the link costs between any pair of virtual nodes. Fig. 2(a) illustrates POP-level topologies of three substrate providers, including the costs (i.e., per capacity unit) for purchasing bandwidth over the intra-domain and peering links. Taking into account the policies of Internet Service Providers (ISPs), the disclosure of router-level topologies is prohibitive. Instead, some ISPs publish topologies with PoPs. However, most of these topologies are oversimplified lacking not only router-level connectivity but also PoP structure [20].

Based on these observations, detailed topology information cannot be assumed to be accessible by VNPs. However, VNPs can enhance their limited substrate network view with certain aspects of the substrate topology which are not treated as confidential. Similar to ISPs, InPs may disclose their PoPs and information about their peerings. A VNP can also collect publicly available information from Internet Exchange Points (IXP) (e.g., DE-CIX [2] advertises topology information and traffic statistics) and databases on peering locations and participants (e.g., peeringDB [6]). Such information enables a VNP to construct a more comprehensive view of the underlay, including the location and connectivity of peering nodes. This is illustrated in Fig. 2(b), which shows the view of VNP on the substrate networks of Fig. 2(a), with $U, V, W, \ldots, Z$ representing the peering nodes.

The peering agreements between InPs (i.e., paid peering or settlement-free peering) will have an impact on the cost of virtual links that span multiple domains. Although the InP policies for inter-domain path selection will remain confidential, any incurring transit fees will be reflected in the advertised



(a) Substrate network topologies



(b) VNP's view on the substrate network topologies

Fig. 2. Substrate network visibility.

	a	b	c
a	-	5	6
b	5	-	8
c	6	8	-

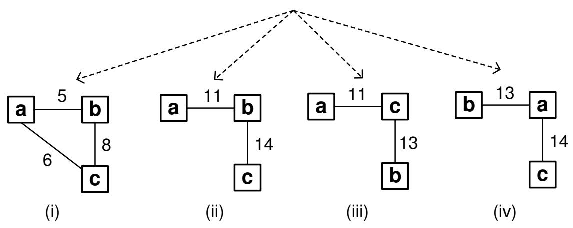
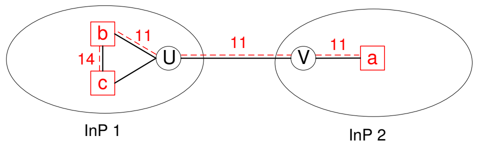
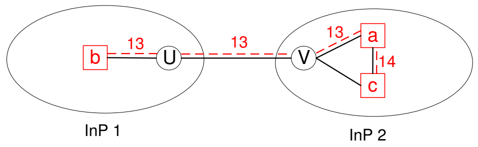


Fig. 3. VN specification with a traffic matrix.



(a) VN partition 1



(b) VN partition 2

Fig. 4. VN request partitions.

peering link costs. More precisely, the peering link costs will comprise the transit fees to the provider (in the case of paid peering) or the cost for the operation and maintenance of the peering link (in the case of settlement-free peering) plus the profit for the InP.

To optimize VN partitioning among InPs, the VNP will seek to minimize the VN embedding cost, which comprises all virtual node and link allocation costs. Since the VNP lacks detailed knowledge of the substrate topology, the total virtual link allocation cost will be estimated based on the advertised costs of the links between the peering nodes, as shown in Fig. 2(b). As such, the VNP will not be in position to account the intra-domain link costs. These costs will be known after the VN assignment by the InPs. Nevertheless, peering link cost is anticipated to be the dominant factor of the total link cost and thereby, the intra-domain link costs will not increase the SP's expenditure significantly.

B. *VN Request Specification*

Nearly all existing embedding algorithms (e.g., [10], [24], [25], [17], [13], [15]) process VN topology requests which are typically specified as undirected weighted graphs. VN topologies introduce unnecessary constraints in the VN embedding problem [21]. In particular, the efficiency of VN embedding depends on the VN topology specification, which in return may require detailed knowledge of the substrate network topology and resource availability. As discussed above, most of this information will remain confidential. On the other hand, SPs may prefer to specify VN requests at a higher level of abstraction, obviating the need for any VN topology specifications. In fact, a VN request comprising the virtual node specifications and the flow demands for each pair of nodes is deemed sufficient for most network service requirements.

As such, we investigate potential gains by using traffic matrices instead of topology-based VN requests. Fig. 3 illustrates a traffic matrix that specifies the required traffic flow units between all pairs of 3 virtual nodes. This traffic matrix essentially abstracts all four topology-based requests shown below, while it further represents the flow requirements of additional VN topologies that include intermediate nodes.

Using a traffic matrix for the VN request specification provides higher flexibility for VN partitioning, as shown in Fig. 4, where U, V represent peering nodes. We consider the partitioning of the VN request of Fig. 3 between two InPs given that virtual nodes a and b can be allocated only from InP 2 and InP 1, respectively, due to specific constraints such as location. As such, VN request partitioning depends only on the assignment of virtual node c. In this respect, Figs. 4(a) and 4(b) represent two different VN partitions. We observe that the topology (ii) is suitable for the VN partition in Fig. 4(a), since it directly gives the capacity requirement for the peering link. The other VN topology-based requests complicate the VN partitioning for different reasons: topologies (iii) and (iv) may result in resource overprovisioning along the peering link increasing the expenditure for the SP, while topology (i) requires a transformation to derive the overall capacity requirement over the peering link. Similarly, among the four topology-based VN requests, only topology (iv) is suitable for the VN partition in Fig. 4(b).

Consequently, topology-based VN requests unnecessarily restrict the VN embedding problem space excluding efficient solutions, in contrast to the traffic matrix counterpart. As such, we provide the ability to embed traffic matrix based VN requests across multiple substrate networks.

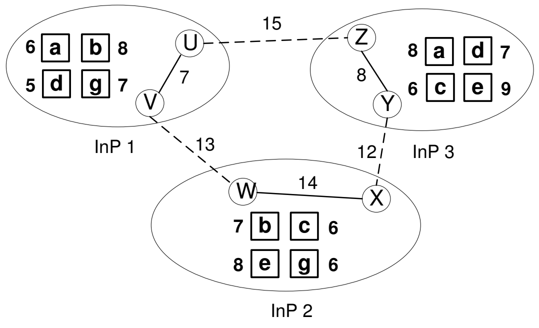


Fig. 5. Resource information accessed by VNP.

III. VN Embedding Framework

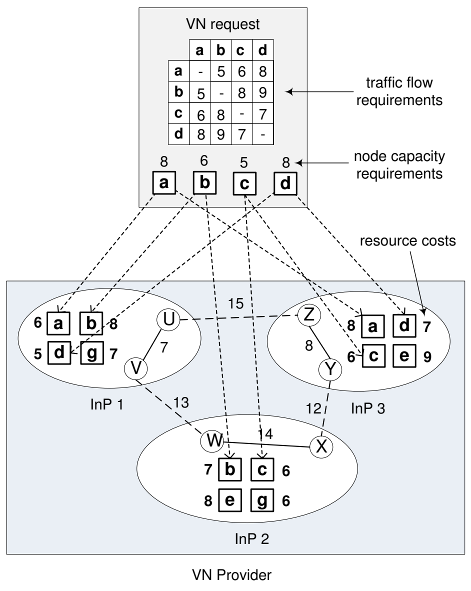
Hereby, we present our multi-domain VN embedding framework. Similar to [15], we decompose multi-domain VN embedding into a set of operations that allow the VNP and the participating InPs to process VN requests based on their knowledge of the substrate network. In the remainder of this section, we exemplify these VN embedding steps.

Resource Information Disclosure. The disclosure of resource information can facilitate resource discovery and VN request partitioning. As discussed in Section II-A, we expect that each InP will advertise the bandwidth cost of the links between peering nodes. Furthermore, we consider InPs advertising one instance of each offered virtual node type along with its unit cost. Virtual node types are explicitly associated with a set of (attribute, value) pairs that comprise the specification of this resource type. All disclosed information is collected and registered by the VNP into a local repository, which is subsequently used for resource matching and VN partitioning.

Fig. 5 illustrates the resource and substrate topology information available to the VNP. This comprises the set of offered virtual node types represented by $\{a,b,c,...,g\}$, the location of peering nodes represented by $\{U,V,W,...,Z\}$, and the connectivity of peering nodes along with the respective link costs. The disclosure of virtual node types promotes competition among InPs, since multiple InPs may offer resources with similar attributes.

Resource Matching. VNPs rely on the information disclosed by InPs to match requested to offered resources. To this end, the VNP identifies a set of candidate resources that fulfill the requirements of each requested virtual node. Fig. 6 depicts how four different virtual node types (i.e., a, b, c, d) of a VN request are matched against the resources advertised by the InPs. In this particular example, the requested and advertised resources have specifications of the same level (i.e., same set of attributes) which facilitates their matching. It is also possible that the attributes of the requested resources comprise a subset of the attributes of the advertised resources. In this case, the identification of matches across the set of disclosed resources requires similarity-search techniques, such as [18], [14].

VN Request Partitioning. VN requests are partitioned across multiple InPs, when none of the participating substrate providers can offer all the resources requested by the SP. Requests for wide-area VN deployments that exceed the



	a	b	c	d
a	-	5	6	8
b	5	-	8	9
c	6	8	-	7
d	8	9	7	-

Fig. 6. Resource matching by VNP.

geographic footprint of any InP comprise such an example.

We consider VN partitioning as a cost minimization problem, taking into account all disclosed resource costs as well as the inter-dependencies between virtual node and link costs. In Section IV-B, we present a formulation for the VN request partitioning problem. Our solution provides a set of VN segments, where each virtual node is mapped onto a particular peering node.

The exact placement of each virtual node is determined by InPs exploiting the detailed knowledge of the substrate resources. To this end, each VN segment is relayed to its corresponding InP, where resource assignment and allocation is carried out. Similar to the initial VN request, each VN segment consists of virtual node specifications and a traffic matrix that represents the flow demands spanning all pairs of virtual and peering nodes (Fig. 7).

Resource Assignment. Upon VN request partitioning, InPs map their assigned VN segments onto their substrate networks while satisfying certain virtual node capacity and flow requirements. The mapping complies with the virtual node to peering node bindings, as given in the VN segment specification. Resource assignment corresponds to the intra-domain VN embedding problem which has been investigated in [10], [24], [25], [17], [13], [15]. However, these methods have limited applicability to our embedding framework, since they cannot process VN requests specified using traffic matrices. In Section IV-C, we present an adapted problem formulation along with a developed solver for the resource assignment problem.

IV. VN Embedding Problem Formulation

In this section, we provide formulations for the VN request partitioning and intra-domain resource assignment problems.

A. Network Model

Substrate Network Model. The substrate network is represented as a weighted directed graph $G_s = (N_s, L_s)$, where N_s

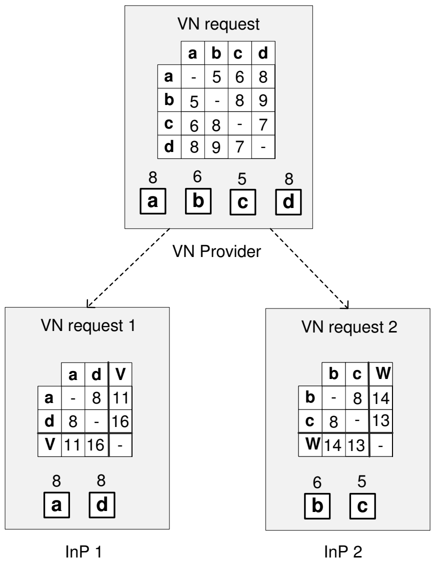


Fig. 7. VN request partitioning. The virtual nodes a,d are mapped onto the peering node V, and the virtual nodes b,c are mapped onto the peering node W. Flow demand specifications for each VN segment are expressed with a traffic matrix.

is the set of substrate nodes and L_s is the set of substrate links between the nodes within the set N_s. Each substrate node $u \in N_s$ is associated with a set of attributes, such as the location and the residual capacity denoted by r_u. Each substrate link $(u,v) \in L_s$ between two substrate nodes u and v is associated with the residual capacity denoted by r_{uv}.

VN Request Model. A VN request consists of the set of virtual nodes N_v and the flow demands d^{ij} between any pair of virtual nodes $i,j \in N_v$. Each virtual node i is associated with a set of attributes (e.g., location) and the required capacity denoted by g^i. For a VN request partitioned into k segments, N_v^k denotes the set of nodes that comprise the VN segment specification, including the virtual nodes plus the assigned peering nodes (e.g., VN request 1 and 2 in Fig. 7). For any pair of nodes $i,j \in N_v^k$, d^{ij} represents the respective flow demands for the kth VN segment.

B. VN Partitioning Formulation

We consider a VNP having access to limited resource and substrate topology information, as discussed in Section II-A. Given the set of disclosed peering nodes across the InPs, denoted by P, we seek the assignment of requested virtual nodes to peering nodes, i.e., $N_v \to P$, such that the cost for the SP is minimized.

We define a binary integer programming (BIP) formulation that takes as inputs the VN request, the node and link allocation costs advertised by InPs, and the virtual node to peering node mapping feasibility obtained from the preceding resource matching phase. In this respect, let c_p^i denote the cost for allocating the virtual node i from peering node p. We further introduce a weight $w_p^i \in \{1,\infty\}$ to specify the feasibility of mapping virtual node i to p. Hence, $w_p^i = \infty$ if the mapping is not feasible (i.e., none of the advertised node specifications matches the requested virtual node specification). We use a binary variable x_p^i to indicate the mapping between virtual node i and p. Similarly, a binary variable $y_{pp'}^{ij}$ indicates whether a pair of virtual nodes i,j are mapped to a pair of peering nodes p,p', respectively. Essentially, the variable $y_{pp'}^{ij}$ allows the VNP to account for the link costs (denoted by $c_{pp'}^{ij}$) between all assigned virtual nodes. In the case that the peering nodes p,p' belong to different InPs, $c_{pp'}^{ij}$ accumulates all the link costs along the inter-domain path from p to p'. In the presence of multiple paths, the path with the lowest cost is selected.

The BIP formulation is expressed as follows:

$$\text{Minimize} \sum_{i\in N_v}\sum_{p\in P} w_p^i c_p^i x_p^i + \sum_{\substack{i,j\in N_v \\ (i\neq j)}} \sum_{p,p'\in P} w_p^i w_p^j c_{pp'}^{ij} y_{pp'}^{ij} \tag{1}$$

subject to:

$$\sum_{p\in P} x_p^i = 1 \quad \forall i \in N_v \tag{2}$$

$$\sum_{p'\in P}\sum_{j\in N_v} y_{pp'}^{ij} + y_{p'p}^{ji} = 2(|N_v|-1)\cdot x_p^i$$
$$i \neq j, \forall i \in N_v, \forall p \in P \tag{3}$$

$$x_p^i, y_{pp'}^{ij} \in \{0,1\} \quad \forall i,j \in N_v, \forall p,p' \in P \tag{4}$$

The first term of the objective function (1) represents the sum of the costs for the mapping of virtual nodes to the peering nodes. By multiplying each node cost c_p^i by w_p^i, we exclude all the infeasible virtual node to peering node mappings (since the cost becomes infinite). The second term of (1) accumulates the link costs between feasibly mapped nodes.

We briefly explain the BIP constraints. Constraint (2) ensures that each virtual node i is mapped to exactly one of the peering nodes. Constraint (3) binds the mapping of each virtual node to the mapping of the respective virtual links. The full traffic matrix implies that a mapped virtual node $i \in N_v$ is connected to $(|N_v|-1)$ virtual nodes $j \in N_v$ using bidirectional links. Finally, condition (4) denotes the binary domain constraints for the variables x_p^i and $y_{pp'}^{ij}$.

C. Resource Assignment Formulation

Next, we formulate the resource assignment problem within InPs as a mixed integer multi-commodity flow problem. In this context, each traffic flow requirement corresponds to a commodity $Com^{ij} = \{i,j,d^{ij}\}$, where $i,j \in N_v^k$ represent the source and destination nodes, respectively, while d^{ij} is the flow demand. The flow variable f_{uv}^{ij} denotes the total amount of flow units on the substrate link (u,v) for the flow demand between the virtual nodes i and j, with $u,v \in N_s$. We also use the binary variable x_u^i to indicate whether the virtual node i is mapped to the substrate node u (i.e., a value of 1 denotes the assignment). Since not all substrate nodes may fulfill the requirements of a virtual node, we use the weight $w_u^i \in \{1,\infty\}$ to denote the feasibility of mapping virtual node i to substrate node u (i.e., $w_u^i = \infty$ denotes that the mapping is not feasible).

Furthermore, we define α_u and β_{uv} as the substrate node and link cost per capacity unit, respectively.

We seek the assignment of all virtual nodes within N_v^k and the computation of flows f_{uv}^{ij} with $i, j \in N_v^k$ and $u, v \in N_s$, such that the VN embedding cost is minimized. The resource assignment problem formulation is as follows:

$$\text{Minimize} \sum_{u \in N_s} \alpha_u \sum_{i \in N_v^k} w_u^i g^i x_u^i + \sum_{\substack{(u,v) \in L_s \\ (u \neq v)}} \beta_{uv} \sum_{\substack{i,j \in N_v^k \\ (i \neq j)}} w_u^i w_v^j f_{uv}^{ij} \quad (5)$$

subject to:

$$\sum_{u \in N_s} x_u^i = 1 \quad \forall i \in N_v^k \quad (6)$$

$$\sum_{v \in N_s} f_{uv}^{ij} - \sum_{v \in N_s} f_{vu}^{ij} = d^{ij}(x_u^i - x_u^j)$$
$$i \neq j, \forall i, j \in N_v^k, u \neq v, \forall u \in N_s \quad (7)$$

$$\sum_{i \in N_v^k} g^i x_u^i \leq r_u \quad \forall u \in N_s \quad (8)$$

$$\sum_{i,j \in N_v^k} f_{uv}^{ij} \leq r_{uv} \quad \forall u, v \in N_s \quad (9)$$

$$x_u^i \in \{0, 1\} \quad \forall u \in N_s, \forall i \in N_v^k \quad (10)$$

$$f_{uv}^{ij} \geq 0 \quad \forall u, v \in N_s, \forall i, j \in N_v^k \quad (11)$$

Conditions (6)–(11) imply the following constraints. First, constraint (6) ensures that each node $i \in N_v^k$ is mapped to exactly one of the substrate nodes. Note that N_v^k includes the virtual nodes as well as their associated peering nodes. To satisfy the flow demands at the peering nodes (as specified in the VN request), we map each peering node $i \in N_v^k$ to its respective substrate (peering) node $u \in N_s$, by setting the w_u vector to infinity, except $w_u^i = 1$. Furthermore, each peering node is associated with zero required capacity, i.e., $g^i = 0$. The constraint (7) enforces flow conservation, i.e., the summation of flows entering or leaving a substrate node, which is not a source or a sink, must be zero. Given flow demand specifications for all pairs of virtual nodes in both traffic directions, the summation of flows must be zero in any substrate node that does not host a virtual node. Constraint (8) enforces the residual capacity limit for each substrate node, while constraint (9) ensures that the summation of flow units on a traffic direction does not exceed the substrate link capacity. Condition (10) denotes the binary domain constraints for the variable x_u^i. Finally, constraint (11) ensures that the flows f_{uv}^{ij} are always positive.

The complexity of the VN partitioning and intra-domain resource assignment problems can be reduced by employing relaxation and rounding techniques (e.g., similar to [10]). Since our main goal is to investigate the VN embedding feasibility with LID, we leave this for future work.

V. Evaluation

In this section, we evaluate the efficiency and discuss the feasibility of VN embedding with limited information disclosure. In Section V-A we present our evaluation environment, in Section V-B we describe the comparison method and metrics, and in Section V-C we discuss our simulation results.

A. Evaluation Environment

We have implemented a simulation environment for multi-domain VN embedding. The implementation consists of a set of modules for resource advertisement, resource matching, VN request partitioning across multiple InPs, and resource assignment within InPs, as discussed in Section III. We implemented all modules in C/C++ and we further relied on the GNU Linear Programming Kit 4.47 (GLPK) [4] to solve the linear programs of Section IV using the branch-and-cut method. Our tests are carried out on a server with one Intel Xeon six-core CPU at 2.66 GHz and 6 GB of main memory. Below, we give more details on the substrate network and VN request specifications, as used in our evaluation.

Substrate Network. For substrate networks, we use synthetic topologies which are generated using the IGen network topology generator [16]. We exemplary fix the number of substrate nodes for each InP to 25 and generate links using the Waxman method [23]. We use a fixed number of substrate nodes for each InP in order to compare results with a diverse number of InPs. The mean number of intra-domain and peering links per InP is set to 70 and 4, respectively. Each node is associated with a location identifier, while the capacity value for each node and link is randomly selected from a uniform distribution. The residual capacities for substrate nodes and links are updated after a new VN request has been admitted or an existing VN allocation has expired. We also update the resource advertisements from an InP to the VNP, when a previously advertised resource is no longer available or in the case that new resources become available due to VN releases.

VN Request. A VN request consists of virtual node specifications and the traffic flow specification between each pair of virtual nodes using a traffic matrix. The number of virtual nodes for each VN request is randomly sampled from a uniform distribution. The node capacity and flow demands also follow a uniform distribution. In each virtual node specification, we use a location attribute, which consists of latitude and longitude values with a maximum tolerance between 250 and 500 distance units. We use the location to purposely enforce constraints on the assignment of virtual nodes onto InPs. Location constraints essentially prohibit the assignment of VNs onto a single InP, without significantly restricting the search space, since different InPs have overlapping geographic presence and consequently any virtual node can be still mapped onto multiple InPs.

In our simulations, we process a fixed number of VN requests with limited lifetime which is randomly given by a uniform distribution. We model the arrival of VN requests through a Poisson process, which is an established assumption in the literature (e.g., [10], [15], [17], [24], [25]). We set

TABLE I
EVALUATION PARAMETERS

Substrate Network	
Nodes per InP	25
Intra-domain links per InP	70 (mean)
Peering links per InP	4 (mean)
Link distribution	Waxman method
Node capacity	uniform distrib. [200, 300]
Link capacity	uniform distrib. [4000, 6000]

Virtual Network Request	
Virtual nodes	uniform distrib. [5, 15]
Node capacity	uniform distrib. [1, 8]
Flow demand	uniform distrib. [1, 10]
Mean arrival rate	1 request per 100 time units
VN lifetime	uniform distrib. [500, 5000]

the mean arrival rate to 1 request for every 100 time units. The evaluation parameters for the substrate network and VN requests are summarized in Table I.

We point out that VNP constitutes an emerging business model that has not yet materialized in the Internet. Given the lack of real-data sources for VN request workloads, our simulation parameters are adjusted after broad inspection of resource pricing by cloud providers (e.g., Amazon EC2) and the input parameters used in other VN embedding evaluation environments [10], [15], [17], [24], [25].

B. Comparison Method

Our main goal is to investigate whether (and to what extent) LID composes a limiting factor for multi-domain VN embedding. We further aim to evaluate the efficiency of our VN embedding method. Due to the lack of previous work on multi-domain VN embedding with a well-defined level of resource and topology information disclosure, we compare our VN embedding solution to a "best-case" scenario where all information is available to VNPs, i.e., full information disclosure (FID). In this case, the complete knowledge of the substrate networks allows their composition into a single substrate where VN requests can be directly embedded using any VN mapping algorithm. For VN embedding with FID, we rely on our resource assignment method (Section IV-C) with the required modifications in terms of node feasibility mapping.

We compare VN embedding with LID versus FID in terms of embedding cost and VN request acceptance rate. We define the *embedding cost* of *one VN request* as:

$$\sum_{k\in K}\left(\sum_{u\in N_s^*}\alpha_u\sum_{i\in N_v^k}g^i+\sum_{\substack{(u,v)\in L_s^*\\(u\neq v)}}\beta_{uv}\sum_{\substack{i,j\in N_v^k\\(i\neq j)}}f_{uv}^{ij}\right)+\sum_{\substack{i,j\in N_v\\(i\neq j)}}\sum_{p,p'\in P^*}c_{pp'}^{ij} \qquad (12)$$

where the superscript $(\cdot)^*$ at the sets P^*, N_s^*, and L_s^* denotes the set of assigned peering points, substrate nodes and substrate links, respectively. The set K comprises all VN segments of a given VN request. The explanation of the remaining subscripts and superscripts is given in Section IV. The embedding cost for multiple VN requests is the sum of the individual VN request costs, each given by (12). The first term in (12) represents the costs associated with assigned substrate nodes and links for all VN segments of one VN request. The second term in (12) comprises the peering costs incurring between the VN segments.

In the following, we use the metric *extra cost* to represent the proportion of additional cost incurred by embedding with LID in comparison to FID. We further explore the origins of this extra cost through regression models that empirically show the correlation with specific model variables. Additionally, we use the VN request acceptance rate to represent the proportion of accepted VN requests.

C. Evaluation Results

Initially, we measure the extra cost with LID across 60 simulations runs. To this end, we consider a large-scale VN embedding scenario with 250 VN requests across a diverse number (5-10) of InPs. Each VN request includes a random number of virtual nodes within $[5,15]$, as discussed in Section V-A. Fig. 8 shows that in the considered scenario LID incurs 15%-20% extra cost compared to the ideal case where all information is available to the VNP. Considering that most resource and substrate topology information is concealed from the VNP, such cost increase is deemed reasonable. This result also corroborates the efficiency of our multi-domain VN embedding solution. Fig. 9 depicts the extra cost for VN embedding with LID versus the number of InPs and the number of virtual nodes per request. This indicates an increase in the extra cost for VNs of larger size.

In addition, we investigate the number of InPs selected for VN embedding with LID and FID. We find no statistical evidence of significant difference of the medians between the two information disclosure levels as given in Fig. 10.

Next, we investigate the origins of the extra cost with LID. To this end, we examine the correlation between the extra cost and the extra link cost. Fig. 11 illustrates a scatter plot of extra cost vs. extra link cost, showing a strong correlation between both quantities. This is validated by the coefficients provided by the regression model, i.e., slope $\beta = 0.82$ with $R^2 = 0.96$ and p-value < 0.001. With respect to nodes, our results (not shown here) do not reveal any perceptible increase in the node cost with LID. This occurs because the costs of virtual nodes with certain specifications are advertised to the VNP and are therefore, taken into account during VN partitioning. In fact, we observe a few cases where the embedding cost minimization with FID results in the selection of more expensive virtual nodes (compared to the nodes assigned with LID) when the associated link cost is lower.

To further investigate the origin of the extra link cost, we depict in Fig. 12 a scatter plot of the extra hop count versus the extra link cost. We perform a linear regression analysis on this data to obtain the following coefficients: slope $\beta = 1.18$, $R^2 = 0.80$, and p-value < 0.001. Thereby, we find that the extra link

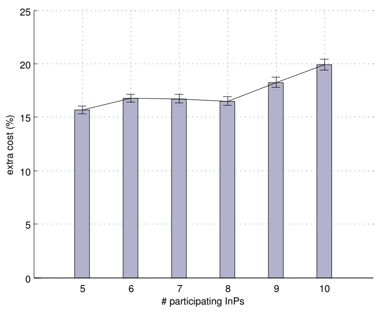


Fig. 8. Extra cost for VN embedding with limited information vs. number of participating InPs.

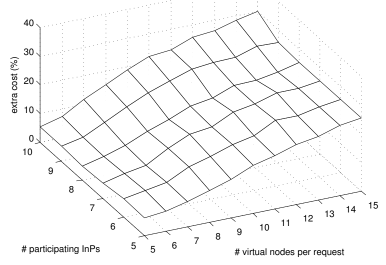


Fig. 9. Extra costs for VN embedding with LID vs. number of participating InPs and vs. VN size.

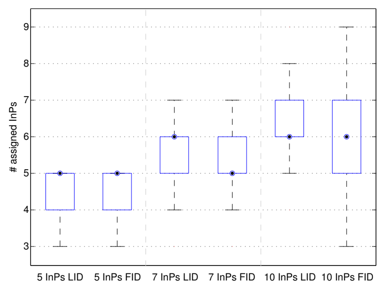


Fig. 10. Assigned InPs vs. participating InPs.

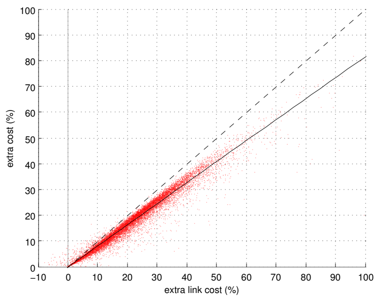


Fig. 11. Scatter plot of overall extra cost vs. extra link cost.

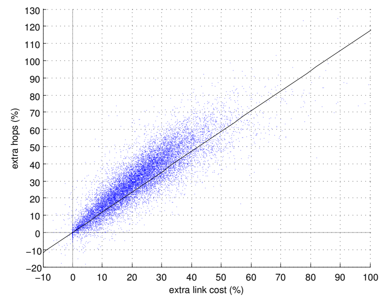


Fig. 12. Scatter plot of extra hops vs. extra link cost.

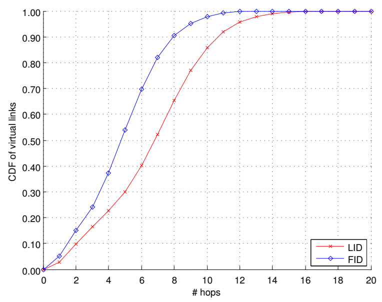


Fig. 13. CDF of hop count of virtual links.

cost is correlated with longer paths (i.e., additional hops). This finding is also confirmed by Fig. 13, which depicts the CDF of the hop count of virtual links both with LID and FID. It is clearly shown that embedding with LID increases the number of hops onto which virtual links are mapped. According to Fig. 13, the maximum increase in the number of hops is 3 for 95% of the virtual links, which is considered acceptable given the restricted access to resource information in this example.

Finally, we compare the acceptance rate of VN requests with LID and FID across 20 simulations runs, each one containing 1000 VN requests. Fig. 14(a) depicts the acceptance rate with non-expiring VN requests. Embedding with LID leads to VN request rejection prior to the full information counterpart. The higher acceptance rate for FID implies better resource utilization and essentially higher revenue for the substrate providers. Fig. 14(b) provides a comparison in the acceptance rate of expiring VN requests with diverse VN request arrival rates under LID. It is shown that the acceptance rates for all arrival rates reach a steady state. This indicates that a significant fraction of the VN requests can be embedded for long periods. As depicted in Fig. 14(b), the steady-state acceptance rate decreases for higher arrival rates.

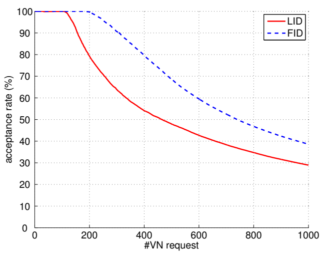
Our simulation results indicate the feasibility of multi-domain VN embedding with LID. Although most of resource and topology information is concealed from the VNP, the embedding cost in most cases is moderately higher, while the VN request acceptance rate converges to a steady-state which ensures predictable embedding. The mapping of virtual links onto longer substrate paths mainly accounts for the extra embedding cost. Furthermore, the higher revenue under FID, as the outcome of the higher acceptance rate, can incentivize InPs to disclose more resource information to VNPs.

VI. Related Work

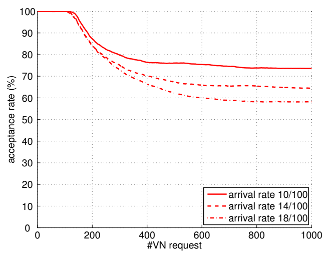
In the following, we discuss related work with respect to VN embedding and network virtualization architectures.

Multi-domain VN Embedding. Multi-domain VN embedding has not been studied in depth. The work in [15] is one of the few studies that provide algorithms for VN embedding across multiple substrate networks. The main goal of [15] is the comparison of exact and heuristic methods in terms of VN embedding efficiency. Without any inspection of resource information disclosure, VN partitioning is carried out based on a highly abstract view of the substrate network (i.e., AS-level topology), which does not include information that is publicly available (e.g., location of PoPs and peering nodes). This abstract underlay view combined with the restrictions of topology-based VN requests can lead to inefficient VN embeddings and increased expenditure for SPs. Based on our simulation results, the low level of information disclosure considered by [15] can increase the embedding cost and reduce the VN acceptance rate and revenue for InPs.

Intra-domain VN Embedding. Most VN embedding algorithms require full knowledge of the available physical resources and substrate topology [10], [24], [17], [25],



(a) non-expiring VN requests.



(b) expiring VN requests for different arrival rates under LID.

Fig. 14. Acceptance rates of VN requests under different scenarios.

[13]. As such, they are not applicable to multi-domain VN embedding with LID. These algorithms are comparable only to our intra-domain resource assignment problem formulation (Section IV-C). In contrast to these algorithms which process and embed topology-based VN requests, we assign VNs in which flow demands are specified using traffic matrices. In this way, we avoid the constraints induced by topology-based VN requests and their restrictions in the problem space.

Network Virtualization Architectures. Previous work [19], [26], [9] presents architectures for VN provisioning across multiple substrate networks. Our multi-domain VN embedding methods are directly applicable to the network virtualization architectures [19], [26] that rely on a layer of indirection between SPs and InPs. The architecture presented in [9], where VN embedding requests are relayed across InPs until the completion of the VN embedding, also benefits from our VN embedding algorithms. In particular, our VN partitioning method (Section IV-B) can be used to determine the InP that should be contacted first, improving the convergence of this distributed VN embedding approach.

VII. Conclusions

In this paper, we conducted a feasibility study on multi-domain VN embedding with LID based on a well-defined level of information disclosure and formulations for the the VN partitioning and intra-domain resource assignment problems. We showed that LID incurs moderately increased embedding costs, in comparison to a "best-case" scenario where all substrate network information is available to VNPs. We also found that the additional embedding cost under LID stems from the increased hop count for virtual links. The lower embedding cost in conjunction with the higher VN request acceptance ratio under FID can incentivize InPs to divulge more information to VNPs, since this can improve the resource utilization and increase the revenue for InPs. Beyond the scope of VN embedding, we believe that our results are encouraging for other emerging business models that require separation between the network operations and the physical infrastructure.

References

[1] Amazon EC2 Instance Types, http://aws.amazon.com/ec2/instance-types/.
[2] DE-CIX, https://www.de-cix.net/.
[3] GENI: Global Environment for Network Innovations, http://www.geni.net/.
[4] GNU Linear Programming Kit, http://www.gnu.org/software/glpk/.
[5] OpenVZ Project, http://www.openvz.org.
[6] PeeringDB, http://www.peeringdb.com/.
[7] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R.Neugebauer, I.Pratt, and A. Warfield, Xen and the Art of Virtualization, Proc. 19th ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA, October 2003.
[8] S. Bhatia, M. Motiwala, W. Muhlbauer, Y. Mundada, V. Valancius, A. Bavier, N. Feamster, L. Peterson, and J. Rexford, Trellis: A Platform for Building Flexible, Fast Virtual Networks on Commodity Hardware, Proc. 3rd ACM ROADS, Madrid, Spain, December 2008.
[9] M. Chowdhury, F. Samuel, and R. Boutaba, PolyViNE: Policy-based Virtual Network Embedding Across Multiple Domains, Proc. ACM SIGCOMM VISA, New Delhi, India, September 2010.
[10] N. Chowdhury, M. Rahman, and R. Boutaba, Virtual Network Embedding with Coordinated Node and Link Mapping, Proc. IEEE Infocom 2009, Rio de Janeiro, Brazil, April 2009.
[11] T. Anderson, L. Peterson, S. Shenker, and J. Turner, Overcoming the Internet Impasse through Virtualization, IEEE Computer, 38(4), 2005, pp. 34–41.
[12] E. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici, and L. Mathy, Towards High Performance Virtual Routers on Commodity Hardware, Proc. ACM CoNEXT 2008, Madrid, Spain, December 2008.
[13] I. Houidi, W. Louati, and D. Zeghlache, A Distributed Virtual Network Mapping Algorithm, Proc. IEEE ICC 2008, Beijing, China, May 2008.
[14] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, and L. Mathy, Adaptive Virtual Network Provisioning, Proc. ACM SIGCOMM VISA, New Delhi, India, September 2010.
[15] I. Houidi, W. Louati, W. Bean-Ameur, and D. Zeghlache, Virtual Network Provisioning Across Multiple Substrate Networks, Computer Networks, 55(4), March 2011.
[16] IGen Network Topology Generator, http://informatique.umons.ac.be/networks/igen.
[17] J. Lischka and H. Karl, A Virtual Network Mapping Algorithm based on Subgraph Isomorphism Detection, Proc. ACM SIGCOMM VISA, Barcelona, Spain, August 2009.
[18] H. Medhioub, I. Houidi, W. Louati, and D. Zeghlache, Design, Implementation and Evaluation of Virtual Resource Description and Clustering Framework, Proc. IEEE AINA 2011, Biopolis, Singapore, March 2011.
[19] 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, Proc. ACM SIGCOMM VISA, Barcelona, Spain, August 2009.
[20] N. Spring, R. Mahajan, and D. Wetherall, Measuring ISP Topologies with RocketFuel, Proc. ACM SIGCOMM, Pittsburgh, USA, August 2002.
[21] C. Wang and T. Wolf, Virtual Network Mapping with Traffic Matrices, Proc. ACM/IEEE ANCS 2011, New York, USA, October 2011.
[22] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, Virtual Routers on the Move: Live Router Migration as a Network Management Primitive, Proc. ACM SIGCOMM 2008, Seattle, USA, August 2008.
[23] B. M. Waxman, Routing of Multipoint Connections, IEEE Journal of Selected Areas in Communications, 6(9), 1988.
[24] M. Yu, Y. Yi, J. Rexford, and M. Chiang: Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration, ACM SIGCOMM Computer Communications Review, 38(2), April 2008, pp. 17–29.
[25] Y. Zhu and M. Ammar: Algorithms for Assigning Substrate Network Resources to Virtual Network Components, Proc. IEEE Infocom 2006, Barcelona, Spain, April 2006.
[26] Y. Zu, R. Zhang-Shen, S.Rangarajan, and J. Rexford, Cabernet: Connectivity Architecture for Better Network Services, Proc. ACM ReArch '08, Madrid, Spain, December 2008.