PolyViNE: Policy-based Virtual Network Embedding Across Multiple Domains

N. M. Mosharaf Kabir Chowdhury
Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
Email: nmmkchow@uwaterloo.ca

Fady Samuel
Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
Email: fsamuel@uwaterloo.ca

***Abstract*—Network Virtualization promises to eliminate the rigidity of the existing Internet architecture by allowing heterogeneous virtual networks (VNs) from multiple service providers (SPs) to coexist over a shared infrastructure. However, to enable network virtualization, end-to-end VNs must be provisioned by embedding VN requests onto substrate networks controlled by multiple infrastructure providers (InPs). Since the VN embedding problem is known to be $\mathcal{NP}$-hard even within a single InP, existing research is limited to proposing heuristics only for that scenario. This paper advocates local autonomy with global competition as the key to addressing the VN embedding problem in the multi-InP scenario. We present PolyViNE, a policy-based inter-domain VN embedding framework, that embeds end-to-end VNs in a decentralized manner. PolyViNE introduces a distributed protocol that coordinates the VN embedding process across participating InPs and ensures competitive prices for SPs. We also present a location aware VN request forwarding mechanism, based on a hierarchical addressing scheme (COST) and a location awareness protocol (LAP), to allow faster embedding and outline scalability and performance characteristics of PolyViNE through quantitative and qualitative evaluations.**

I. Introduction

Throughout the years, the Internet has proven itself to be capable of supporting varying distributed applications and networking technologies. However, it has traditionally been enabled by multiple stakeholders with conflicting goals and policies. The resulting tension has ossified the underlying architecture of the Internet and is restricting the rate of its evolution [1], [2]. Network virtualization holds the promise of eliminating the rigidity of the Internet architecture by allowing multiple heterogeneous network architectures to coexist on a shared substrate [2], [3]. In a network virtualization environment (NVE), multiple service providers (SPs) can create heterogeneous virtual networks (VNs) to offer customized *end-to-end* network services by leasing shared resources from one or more infrastructure providers (InPs) without significant investment in physical infrastructure [1]–[3]. As a result, network virtualization can be the perfect solution for rapid prototyping and quick deployment of experimental and industrial services and technologies.

The first step toward enabling network virtualization is to instantiate such VNs by embedding[1] VN requests onto a substrate network. But the VN embedding problem, with constraints on both virtual nodes and virtual links, is known to be $\mathcal{NP}$-hard [4], [5]. Several heuristics [4]–[7] have been proposed to address this problem in the *single InP* scenario. However, in realistic settings, end-to-end VNs must be provisioned across heterogeneous administrative domains belonging to multiple InPs to deploy and deliver distributed network services.

The biggest challenge in end-to-end VN embedding from the perspective of the InPs is to organize them under a framework without putting restrictions on their local autonomy. Each InP should be able to embed parts or the whole of a VN request by administering itself using its internal set of policies while maintaining global connectivity through mutual agreements with other InPs. SPs, on the other hand, are concerned about rapid instantiation and *fair value* for their VN requests without having to manually contact multiple geographically distributed InPs for leasing virtual resources.

This paper presents PolyViNE, a policy-based end-to-end VN embedding framework, that embeds VNs across multiple InPs in a globally distributed manner while allowing each concerned InP to enforce its local policies at the same time. PolyViNE introduces a distributed protocol that coordinates the participating InPs and ensure competitive pricing through repetitive bidding at every step of the embedding process.

The remainder of the paper is organized as follows. We provide a formal definition of the inter-domain VN embedding problem in Section II and derive the requirements for an inter-domain VN embedding framework (i.e., PolyViNE) in Section III. In Section IV we describe the design choices and a high-level overview of PolyViNE, followed by a discussion of its enabling technologies in Section V. Section VI and Section VII respectively provide quantitative and qualitative evaluations of PolyViNE. Section VIII summarizes related works, and we conclude the paper in Section IX by summarizing our contributions and by outlining future directions.

II. Problem Formulation

Unlike the existing Internet, the role of the traditional ISPs has been divided into two in a network virtualization environment (NVE): *infrastructure providers (InPs)*, in this

[1]The words 'embedding', 'mapping', and 'assignment' are used interchangeably throughout this paper.

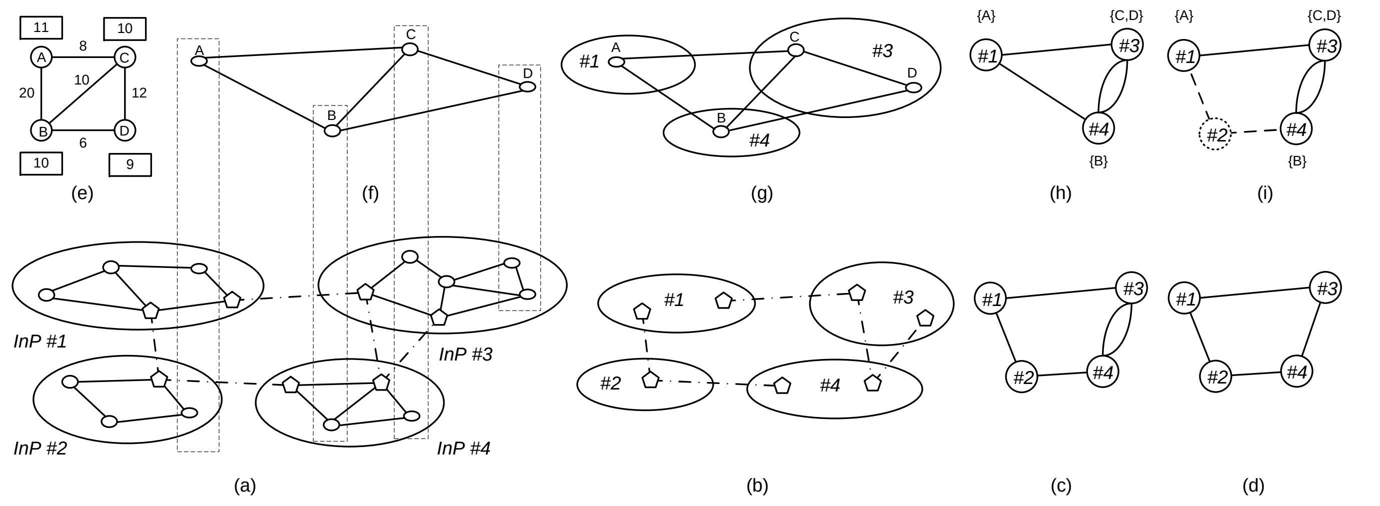


Fig. 1. Overview of inter-domain VN embedding: (a) substrate networks ($G_i^S = \left(N_i^S, L_i^S\right)$) from four InPs connected using inter-domain links; (b) the underlay graph ($G^U = \left(N^U, L^U\right)$) consisting of Adapters and inter-domain links; (c) the underlay multigraph ($G^W = \left(N^W, L^W\right)$) after topology abstraction; (d) Controller Network ($G^C = \left(N^C, L^C\right)$) obtained through simplification; (e) a single VN request ($G^V = \left(N^V, E^V\right)$) with CPU constraints in boxes and bandwidth constraints over links; (f) the same VN request (G^V) with location constraints on the virtual nodes shown in vertical boxes covering possible host physical nodes for them; (g) the embedded VN request with virtual nodes mapped into three different InPs; (h) the meta-VN request ($G_M^V = \left(N_M^V, L_M^V\right)$); (i) an InP-level view of the embedding (note that, InP #2 has not embedded any virtual node instead it is taking part in the embedding by taking part in an inter-domain virtual link).

case, are in charge of actual physical networks (a.k.a. *substrate networks*) and connect between themselves to create the complete underlying infrastructure (i.e., the *underlay*), while *service providers (SPs)* deploy customized network protocols on virtual networks (VNs) by aggregating resources from multiple InPs and offer end-to-end services to end users without significant investment in infrastructure. In this section, we formally define the inter-domain VN embedding problem.

A. Substrate Networks and the Underlay

We consider the underlay to be comprised of D substrate networks (Fig. 1(a)), and we model each substrate network controlled by the i-th InP[2] $(1 \le i \le D)$ as a weighted undirected graph denoted by $G_i^S = \left(N_i^S, L_i^S\right)$, where N_i^S is the set of substrate nodes and L_i^S is the set of *intra-domain* substrate links. Each substrate node $n^S \in N_i^S$ is associated with the CPU capacity weight value $c\left(n^S\right)$ and geographic location $loc\left(n^S\right)$. Each substrate link $l^S\left(n^S, m^S\right) \in L_i^S$ between two substrate nodes n^S and m^S is associated with the bandwidth capacity weight value $b\left(l^S\right)$ denoting the total amount of bandwidth. Each substrate network has a (centralized or distributed) logical Controller [8] that performs administrative/control functionalities for that InP. Finally, $A_i^S (\subset N_i^S)$ denote the set of Adapters [8] in the i-th InP that connect it to other InPs through *inter-domain* links based on Service Level Agreements (SLAs) to form the underlay. Each InP also has a set of policies $\mathcal{P}_i^S$ that is used to take and enforce administrative decisions.

[2]We will use the terms InP and substrate network interchangeably throughout the rest of this paper.

We denote the underlay (Fig. 1(b)) as a graph $G^U = \left(N^U, L^U\right)$, where $N^U \left(= \sum_i A_i^S\right)$ is the set of Adapters across all InPs $(i \le i \le D)$ and L^U is the set of physical inter-domain links connecting the border nodes between two InPs. However, the underlay does not have the full connectivity, which is achieved through simple topology abstraction method [9]. All border nodes belonging to a single InP are collapsed to one single node corresponding to that InP (Fig. 1(c)) in this representation resulting in a multigraph $G^W = \left(N^W, L^W\right)$, where N^W essentially is the set of InPs in the underlay and $L^W (= L^U)$ is a multiset of inter-domain links that connect the InPs. Finally, $G^C = \left(N^C, L^C\right)$ is a simple graph (Fig. 1(d)) referring to the Controller Network [8], where $N^C (= N^W)$ represents the set of Controllers in InPs and L^C is the set of links between Controllers obtained from the multiset L^W.

B. VN Request

Similar to substrate networks, we model VN requests as weighted undirected graphs and denote a VN request by $G^V = \left(N^V, E^V\right)$. We express the requirements on virtual nodes and virtual links in terms of the attributes of nodes and links in the substrate networks. Fig. 1(e) depicts a VN request with virtual node and virtual link requirements.

Each VN request has an associated non-negative value R^V expressing how far a virtual node $n^V \in N^V$ can be placed from the location specified by $loc\left(n^V\right)$ [7], which can be interpreted as the preferred geolocation of that virtual node. Fig. 1(f) shows the substrate nodes within the preferred geolocation for each virtual node using dashed vertical boxes.

Finally, similar to InPs, SPs can provide a set of policies/preferences $\mathcal{P}^V$ for a VN request to dictate certain characteristics. For example, the VN request in Fig. 1(e) could have a policy that would require embedding of the virtual node C in $InP\#3$ domain, ruling out the other possible embedding in $InP\#4$.

C. VN Assignment

The assignment of an end-to-end VN request V onto the underlay can be decomposed into three major components: (i) partitioning the VN request into K subgraphs to be embedded onto K substrate networks, (ii) establishing inter-connection between the K subgraphs using inter-domain paths, and (iii) embedding each subgraph in each InP substrate network using intra-domain algorithm. Since we want to allow each InP to implement its own embedding algorithms, intra-domain embedding is out-of-scope of this work. From PolyViNE's point of view, an end-to-end VN assignment is performed on the Controller Network.

The VN request $G^V = \left(N^V, L^V\right)$ is partitioned into K subgraphs $G_k^V = \left(N_k^V, L_k^V\right)$ such that $N^V = \cup_k N_k^V$ and $L^V = \left(\cup_k L_k^V\right) \bigcup L_M^V$, where L_M^V is the set of virtual links that will cross domain boundaries. In Fig. 1(g), $K = 3$: $G_1^V = (\{A\}, \{\})$, $G_2^V = (\{B\}, \{\})$, $G_3^V = (\{C, D\}, \{CD\})$, and $L_M^V = \{AB, AC, BC, BD\}$. Each subgraph G_k^V can be collapsed into a single node to form the meta-VN request $G_M^V = \left(N_M^V, L_M^V\right)$ using a transformation function $\mathcal{F} : G_k^V \rightarrow N_M^V$ (Fig. 1(h)) for simplicity.

Now we can formally express inter-domain VN embedding as two mappings, $\mathcal{M}_N : N_M^V \rightarrow N^C$ that embeds each subgraph to different InP and $\mathcal{M}_L : L_M^V \rightarrow L^C$ that embeds inter-domain links in the InP Controller Network. In Fig. 1(i), the node mapping is $\Big\{(\{A\}, \{\}) \rightarrow \#1, (\{B\}, \{\}) \rightarrow \#4, (\{C, D\}, \{CD\}) \rightarrow \#3\Big\}$ and the link mapping is $\Big\{(A, B) \rightarrow \{(\#1, \#2), (\#2, \#4)\}, (AC) \rightarrow \{(\#1, \#3)\}, (BC) \rightarrow \{(\#4, \#3)\}, (BD) \rightarrow \{(\#4, \#3)\}\Big\}$

III. Requirements Analysis

From the problem formulation, it can be seen that any inter-domain VN embedding must consider the following phenomena/requirements:

A. Framework for Resource Trading

In order to create geographically distributed end-to-end VNs, SPs must contact multiple InPs with available resources in those regions. One way is to establish individual agreements to put together a solution that will satisfy the constraints of all the concerned parties. But such a solution can be cumbersome as well as inefficient. In order to enable free trading and fair pricing, a framework is required that will enable SPs and InPs to communicate and partition VN requests among possible InPs. Such a framework can be distributed without the presence of any arbitrator/broker between the requester SP and the provider InPs, or it can be a centralized one.

B. Tussles between Contrasting Utility Functions

All SPs and InPs are selfishly concerned about maximizing individual utility functions. Consequently, two major classes of contentions can arise:

1) *Between InPs:* Each InP will be interested in getting as much of the deployment as possible put on its equipment, and then optimizing allocation under given constraints. In addition, InPs will be more interested in getting requests for their high-margin equipment while offloading unprofitable work onto their competitors.
2) *Between SPs and InPs:* SPs are also interested in getting their requirements satisfied while minimizing their expenditure. Contentions might arise between SPs and InPs when each party try to optimize their utility functions.

Any inter-domain VN embedding framework must take these tussles into account and enforce proper incentives and mechanisms to address these contentions.

C. Concerns about Information Sharing between Multiple InPs

Intra-domain VN embedding algorithms [4]–[7] assume that they have a complete picture of the substrate network. However, in the inter-domain VN embedding problem this may not be a reasonable assumption. InPs, in this case, might not be interested in sharing their topologies and other information with their counterparts. Each InP will have to embed a particular segment of the VN request without any knowledge of how the rest of the VN request has already been mapped or will be mapped based only on its internal policies. Such individual, possibly selfish, behaviors can also give rise to unforeseen fallouts in intra-domain embeddings by each InP.

IV. PolyViNE Overview

In this section, we discuss the decisions we have made in designing PolyViNE, followed by an overview of its workflow. We also describe the distributed protocol that coordinates the PolyViNE embedding process.

A. Design Choices

Based on the requirements analysis, we have made design choices for PolyViNE aiming toward decentralization of the embedding process, promotion of policy-based decision making, and support for local agility within flexible global framework.

1) Decentralized embedding: PolyViNE argues for using a distributed (decentralized) VN embedding solution over a centralized broker-based one. In a centralized solution, the broker will have to know the internal details and mutual agreements between all the InPs to make an informed embedding. However, InPs are traditionally inclined to share as little information as possible with any party. A distributed solution will allow for embedding based only on mutual agreements. Moreover, in a distributed market there will be no single-point-of-failure or no opportunity for a monopolistic authority (e.g., the broker).

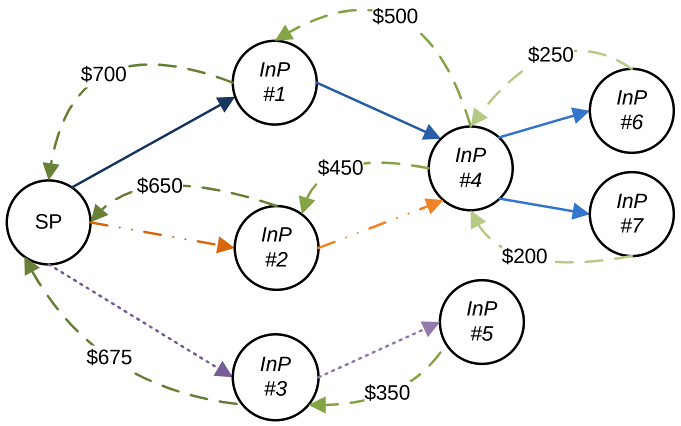


Fig. 2. Propagation of multiple instances of the same VN request in the Controller Network throughout the embedding process. Each InP performs a partial embedding of a request instance and outsources the rest to another InP. The dashed lines demonstrate the back-propagation of accumulated prices toward the SP.

2) Local autonomy with global competition: PolyViNE allows each InP to use its own policies and algorithms to take decisions without any external restrictions. However, it also creates a high level of competition among all the InPs by introducing competitive bidding at every level of distributed VN embedding. Even though each InP is free to make self-serving decisions, they have to provide competitive prices to take part and gain revenue in PolyViNE. To keep track of the behavior of InPs over time, a reputation management mechanism can also be introduced [10].

3) Location-assisted embedding: PolyViNE decision making and embedding process is deeply rooted into the location constraints that come with each VN request. After an InP embeds a part of a VN request, instead of blindly disseminating the rest of the request, it uses geographic constraints as beacons to route the request to other possible providers. PolyViNE aggregates and disseminates location information about how to reach a particular geographical region in the Controller Network and which InPs might be able to provide virtual resources in that region.

B. Workflow Summary

PolyViNE is an enabling framework for multi-step distributed embedding of VN requests across InP boundaries. In its simplest form, an SP forwards its VN request to multiple known/trusted InPs; once they reply back with embeddings and corresponding prices, the SP chooses the VN embedding with the lowest price similar to a bidding process.

However, a complete end-to-end VN request may not be mappable by any individual InP. Instead, an InP can embed a part of the request and *outsource* the rest to other InPs in a similar bidding process giving rise to a recursive multi-step bidding mechanism. Not only does such a mechanism keep a VN embedding simple for an SP (since the SP does not need to contact all of the eventual InPs), but it also ensures competitive prices due to bidding at every step. Fig. 2 provides a high level depiction of the inter-domain VN embedding process.

C. PolyViNE Embedding Protocol

In order to exchange information between the SP and the InPs, and to organize the distributed embedding process, a communication protocol must be established. We refer to this protocol as the PolyViNE Protocol, which is based on six types of messages: *EMBED*, *SUCCESS*, *FAILURE*, *CONNECT*, *RELAY* and *ACK*. These messages are sent and received asynchronously between concerned InPs and the SP to carry out the embedding process from beginning to end. The protocol messages are described in the following:

- **EMBED** $(Req\_id, G, InPSet)$**:** This message is sent from the SP to InPs to initiate the embedding process of the VN request G with an empty $InPSet$. InPs also use this message to outsource the unmapped part of the request after appending itself to $InPSet$.
- **SUCCESS** $(Req\_id, \mathcal{M}, InPSet)$**:** Once an embedding is successfully completed, InPs reply back with the embedding $\mathcal{M}$ and the set of InPs involved in that embedding, $InPSet$.
- **FAILURE** $(Req\_id, errorDesc)$**:** In case of a failure, InPs reply back with a description outlining the reason of failure using $errorDesc$.
- **CONNECT** $\left(Req\_id, G_M^V, InPSet\right)$**:** Once all the different parts of a VN request are embedded by different InPs, the meta-VN request G_M^V is formed. This message is sent to the InPs that will connect the nodes in G_M^V using inter-domain links.
- **RELAY** $(Req\_id, G, InPSet, InP\#)$**:** When an InP cannot embed any part of a request, it may relay the request to one or more InPs instead of announcing failure right away. $InP\#$ in this case indicates the requester InP to which the new InPs should directly reply back to.
- **ACK** (Req_id)**:** Once an SP decides on an embedding after receiving one or more *SUCCESS* messages, it will acknowledge the embedding by directly contacting to the InPs involved using this message.

D. SP Workflow

Since there is no centralized broker in PolyViNE, each SP must know at least one InP to send the VN request it wants to instantiate. However, sending the request to only one InP can encourage monopolistic behavior. To create a competitive environment, we argue that an SP should send its VN request to $k^{SP}(\geq 1)$ InPs based on direct contact. Fig. 3 depicts an SP sending embedding requests using the *EMBED* message to k^{SP} InPs. As soon as the receiving InPs have viable embeddings ($\mathcal{M}$) with corresponding prices ($Price\,(\mathcal{M})$) or they fail, the k^{SP} InPs reply back with *SUCCESS* and *FAILURE* messages, respectively. Once the SP selects an embedding, it proceeds toward instantiating its VN by sending *ACK* messages to the InPs involved in the selected embedding.

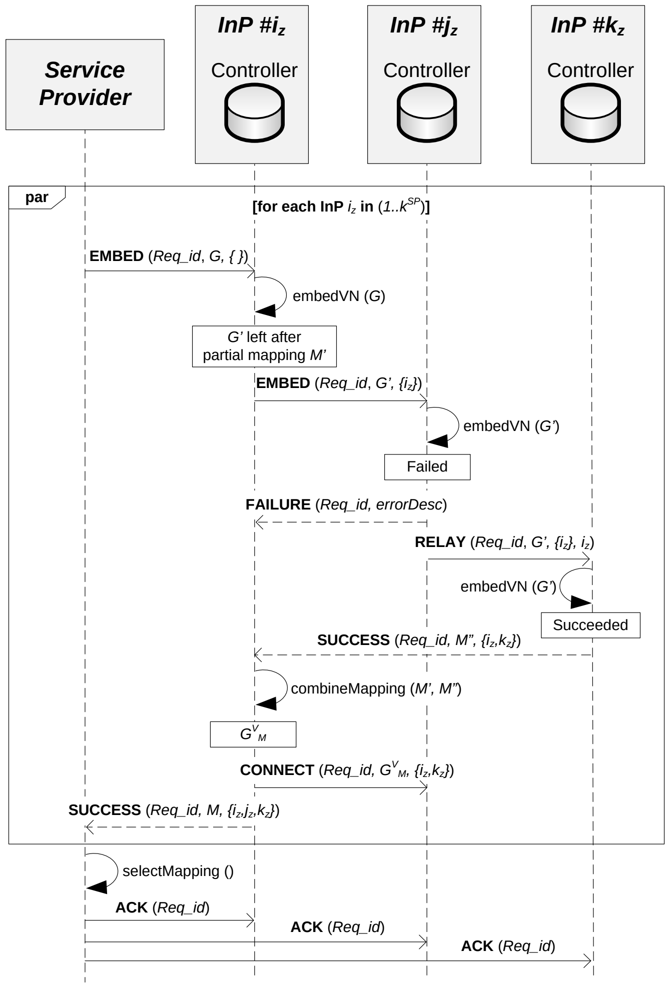

Service Provider
InP #i_z
Controller
InP #j_z
Controller
InP #k_z
Controller
par
[for each InP i_z in (1..k^{SP})]
EMBED (Req_id, G, { })
embedVN (G)
G' left after partial mapping M'
EMBED (Req_id, G', {i_z})
embedVN (G')
Failed
FAILURE (Req_id, errorDesc)
RELAY (Req_id, G', {i_z}, i_z)
embedVN (G')
Succeeded
SUCCESS (Req_id, M'', {i_z,k_z})
combineMapping (M', M'')
G^V_M
CONNECT (Req_id, G^V_M, {i_z,k_z})
SUCCESS (Req_id, M, {i_z,j_z,k_z})
selectMapping ()
ACK (Req_id)
ACK (Req_id)
ACK (Req_id)

Fig. 3. Sequence diagram showing PolyViNE embedding process in action.

E. InP Workflow

While an SP's workflow is straightforward with a single decision at the end, it shifts much more work to the InPs involved in the PolyViNE embedding process. An InP has to work through several steps of decision making, organizing, and coordinating between heterogeneous policies to complete the embedding process. From an InP's point of view, there are three major stages in embedding each end-to-end VN request.

1) Local embedding: Upon receiving a VN request, an InP must decide whether to reject or to accept the request. It can reject a VN request outright, in case of possible policy violations. Even if there are no discernible policy violations, it might still need to reject a VN request if it fails to profitably[3] embed any part of that request. In case of a failure, the InP will send back a *FAILURE* message (optionally, with reasons for the failure). However, sometimes it might know of other InPs that it believes will be able to embed a part or whole of the VN request. In that case, it will *RELAY* the VN request to that InP. In Fig. 3, $InP\#j_z$ is relaying the VN request G' to $InP\#k_z$.

[3]Each InP uses its own pricing mechanism by which it attaches a price to any embedding it provides.

2) Forwarding: If an InP can only partially embed a VN request, it will have to forward the rest of the request to other InPs in the Controller Network in order to complete the VN request. An InP should take care to not forward a VN request to another InP already in the $InPSet$ to avoid cycles. For example, $InP\#i_z$ in Fig. 3 is forwarding the unmapped VN request G' to $InP\#j_z$. Similar to SPs, InPs also forward the request to $k^{InP}(\geq 1)$ InPs for similar reasons (e.g., competitive prices). While forwarding a request, an InP can prefer to perform a transformation on the VN request in order to hide the details of its mapping (as in Fig. 1(h)). At this point, it can use one of the two possible methods for unmapped VN request forwarding:

- *Recursive forwarding:* In this case, when an InP forwards a VN request, the receiver InP embeds part of it based on its policies and forwards the rest further away to another InP.
- *Iterative forwarding:* In iterative forwarding, the receiver InP return the control back to the sender InP once it is finished with embedding.

In any case, the decision of which InP to send the request to next is a non-trivial one and requires careful consideration. We believe that instead of blindly forwarding based on completely random or deterministic heuristics, we can do informed forwarding by utilizing the location constraints attached to all the virtual nodes in a VN request. Details of this forwarding scheme is presented in the next section.

3) Back-propagation: The VN request proceeds from one InP to the next, until either there are no available InPs to send the request to (*FAILURE*) or the VN request has been satisfied completely (*SUCCESS*). In case of a successful embedding of a VN request, the *SUCCESS* message carries back the embedding details and corresponding price. At each step during this back-propagation of *SUCCESS* and *FAILURE* messages, the sender InP gets to select mappings based on internal policies or lower price or some other criteria. As VN embeddings follow paths back to the SP, the prices are accumulated and the SP ends up with multiple choices. The dashed arrows in Fig. 2 depict the back-propagation of embeddings and corresponding prices toward the SP.

V. Location Aware Forwarding

Naïvely an InP can forward a VN request to a set of InPs in the Controller Network at random. However, this decision is blind to the location requirements of the virtual nodes and the availability of virtual resources at the destination InP to satisfy the constraints for the VN request. This may result in high failure rate or prices well above the fair value. To avoid flooding a VN request or sending it to random InPs which might be unable to meet the constraints of the request, we propose using location constraints associated with unassigned virtual nodes to assist an InP in making this decision. Location constraints of the virtual nodes in conjunction with the location information of the underlay will allow informed VN request forwarding in the Controller Network.

To accommodate such location aware forwarding, we introduce a hierarchical geographic addressing scheme with support for aggregation, named COST. InPs in PolyViNE must associate COST addresses with all the substrate nodes and SPs must express location requirements in terms of COST. Controllers in different InPs publish/disseminate information about the geographic locations of their nodes along with the unit price of their resources. They can then aggregate and disseminate data collected from all neighboring Controllers to build their own knowledge bases of location to InP mappings, each accompanied by path vectors of InPs in the Controller Network and corresponding prices. We propose Location Awareness Protocol (LAP) that performs this very task in PolyViNE.

The following subsections will discuss the COST addressing scheme, the LAP protocol, and the policy decisions that need to be made throughout the process.

A. COST Addressing Scheme

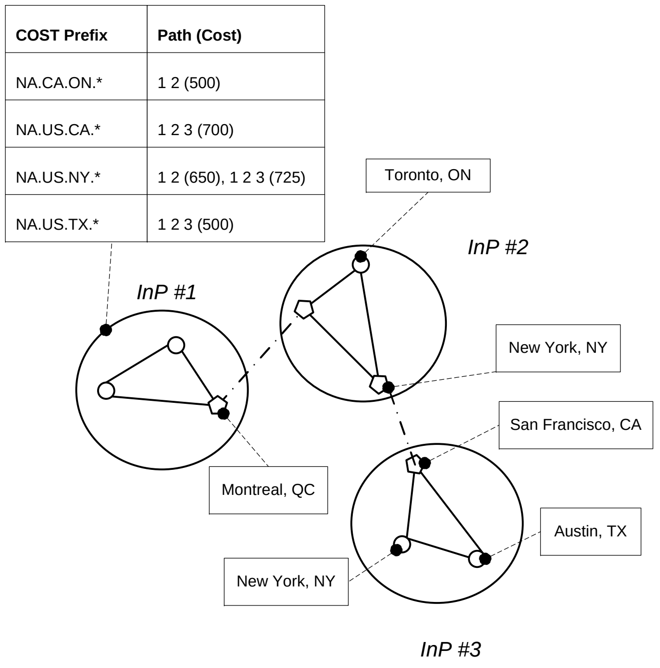
As outlined in the problem formulation (Section II), each virtual node in a VN request comes with a permissible geographic region in which it must be embedded. One design question at this point is how to represent and encode the geolocation. We have chosen a hierarchical geolocation representation scheme with the form *Continent.cOuntry.State.ciTy* (hence the name *COST*). Even though in this paper we are using a simple postal address like scheme for simplicity, any hierarchical geolocation representation system will work with PolyViNE.

A virtual node may restrict its location preference to any prefix in this addressing scheme. For example, to restrict a node within Canada, one may assign the address NA.CA.* to a virtual node. This indicates that beyond requiring that the node be mapped within Canada, the SP does not care where in the country it is ultimately mapped.

On the other hand, each substrate node has a complete COST address associated with it. This address indicates within which city lies the given substrate node. If an InP is not willing to share the exact location, it can always choose a higher level address. For example, instead of announcing nodes in Toronto using NA.CA.ON.Toronto, the InP can announce NA.CA.ON.*. However, such announcements can result in receiving of VN requests that it may never be able to satisfy, which will again affect the InP's reputation among other InPs.

B. Location Awareness Protocol (LAP)

Location Awareness Protocol (LAP) is a hybrid of Gossip and Publish/Subscribe protocols that assists an InP in making informed decisions about which InPs to forward a VN request to without making policy violations, and thus progressing toward completing the VN embedding. Controllers in different InPs keep track of the geolocations of their internal substrate nodes in COST format and announce availability and prices of available resources to their neighbors using LAP updates in the Controller Network. This information is aggregated and propagated throughout the Controller network to create global



COST Prefix	Path (Cost)
NA.CA.ON.*	1 2 (500)
NA.US.CA.*	1 2 3 (700)
NA.US.NY.*	1 2 (650), 1 2 3 (725)
NA.US.TX.*	1 2 3 (500)

Fig. 4. LAP database at InP #1. InP #1 has two choices to forward to an InP with a node in New York state.

view of the resources in the underlay in each Controller's LAP database. An InP can also provide mechanisms to allow InPs other than its neighbors to subscribe to its LAP updates to enable faster propagation of information.

Initially, LAP operates as a path vector based gossip protocol. Every InP in the Controller Network informs its neighbors of where its nodes are located along with estimated unit prices for its resources. Whenever a Controller receives a LAP update, it updates its LAP database and before announcing updates to its neighbors it adds itself to the path vector. It should be noted that keeping complete paths allows avoiding unnecessary forwarding toward and through InPs that might violate SP's policies or originating InP's policies. InPs can also tune this price to encourage or discourage VN request forwarding to themselves. In steady-state, each InP should know about all the InPs with nodes in a given geographic region along with price estimations of embedding on their substrate networks. Fig. 4 shows the LAP database of an InP.

However, in a rapidly changing environment with continuously fluctuating prices, gossip may not be sufficient to disseminate updated prices in a timely fashion. To reduce the number of failures stemming from staleness of pricing information, we propose extensions to LAP using a Publish/Subscribe mechanism along with its basic gossip protocol. By using this mechanism, any InP will be able to subscribe to announcements of Controllers that are not its direct neighbors. While we leave VN request routing decisions to the discretion of InPs, an InP may use the pricing information to prefer forwarding the VN request to a lower priced InP, all other things being equal.

The question that remains open to more investigation is why would an InP wish to be honest when announcing pricing estimates? Why would it not want to simply advertise an estimate of 0 so that other InPs will prefer to forward VN requests to it? We believe that a reputation metric is necessary to remedy this situation. A reputation metric will be an indication of how accurate an InP's pricing estimate is to the actual cost of establishing a VN request. If an InP is known to be wildly off in its pricing estimates, InPs dealing with it will report poor reputation indicators which, in turn, will dissuade other InPs from forwarding VN requests to it in the future. We would like to integrate such reputation metric within LAP to allow dissemination of path vectors attributed with corresponding prices as well as overall reputation score of the InPs on the paths. While taking a forwarding decision, an InP will then be able to use pricing and reputation scores to rank multiple paths to a common destination.

VI. Experimental Evaluation

In this section, we first describe the simulator that we have developed to evaluate PolyViNE followed by the experimental settings for the simulations. Later, we present the experimental results from four simulation scenarios that evaluate different characteristics of PolyViNE.

A. Simulator Design

To evaluate PolyViNE, we have developed a multi-threaded C++ simulator that allows independent responses from various entities in the Controller Network. At the time of writing this paper, the simulator sits at nearly 3000 lines of code. While the embedding process is complete for the most part, the back-propagation of price information, and the selection of the lowest-priced mapping at each step is yet to be implemented.

Each InP object in the simulator runs in a separate thread that receives and processes messages from its neighbors. Each SP object runs in two threads. The update thread, similar to the InP objects, receives and processes (LAP) messages from neighbors. The other thread generates and disseminates random VN requests of various sizes at fixed intervals. Each entity's update thread begins by disseminating its location information to its neighbors. It then sleeps on a semaphore until it is woken up when a message is sitting on its *pending* queue. It processes the message and sleeps again until the next message arrives.

In our current experiments, each InP performs a naive greedy node and link mapping. An InP first attempts to map as many nodes as possible. Then, it tries to map as many links between the mapped nodes as possible. If a link cannot be mapped to a substrate path, then one of the nodes is dropped, which ultimately results in the largest possible connected component mapped in each InP. The InP then picks out a random node yet to be mapped, and uses its LAP table to look-up InPs that can satisfy the node's location constraints. It forwards the partial request to the top three InPs provided by LAP. After 12 hops between InPs, we drop the VN request and report failure.

B. Simulation Settings

We have performed four preliminary experiments on an Intel Core 2 Quad Q6600 processor with 4GB RAM. We leave each experiment running for 10 minutes per data point, log messages, and aggregate data to disk. Finally we use the collected information from disk to produce the result presented.

Unless otherwise specified, we have used the following settings for the experiments presented in this section. For each experiment, we randomly create a Controller Network with 100 InPs. Each InP network consists of 80 to 100 nodes and 540 to 600 links on the average. Each node has a maximum available CPU capacity uniformly chosen from 1 to 100 CPU units, and each link has a maximum available bandwidth capacity of 100 bandwidth units. VN requests vary between experiments in terms of the average numbers of virtual nodes and virtual links.

C. Node Mapping and Hop Count

In the first experiment, we investigate the number of virtual nodes mapped by each InP located increasing number of hops away from the SP. Intuitively, as a VN request is passed forward from one InP to another, each InP will map some virtual nodes and leave fewer and fewer nodes *to be* mapped by the other participating InPs. Fig. 5 confirms this intuition. In this particular result, we have used VN requests with 50 nodes and 200 links on the average.

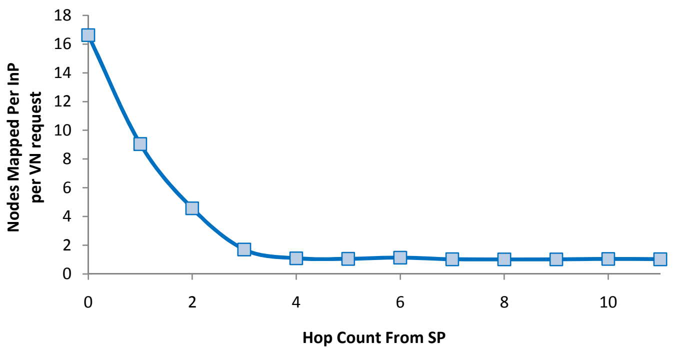


Fig. 5. Nodes mapped per InP per VN request vs. hop count: As a VN request is passed forward, less work remains and so fewer nodes are mapped per InP. We do not consider intermediate InPs that are participating in the mapping, but do not themselves map any nodes. These *relay* InPs merely reserve bandwidth to carry traffic between participating InPs.

We observe that there appears to be an exponential decay in the number of nodes mapped as the hop count increases. At this time, we are uncertain why this pattern has emerged. This pattern may be a function of the random graphs we generated or the simple greedy algorithm we use for mapping or both, but we are yet to pin down the actual cause. If this exponential decay is found to be a reproducible property of VN mappings in random graphs, we may be able to use that information to compute a reasonable number of hops that must be allowed for the completion of a successful VN mapping.

D. Node Mapping and VN Request Size

In the second experiment, we look at the number of nodes mapped by the first set of InPs neighboring the request-generating SP. We vary the VN request size, where each VN request is a sparse random graph with an expected n nodes and $4n$ links (n is set between 10 and 70).

Fig. 6 demonstrates that the number of nodes mapped by the first-hop InPs grows linearly with the size of the VN request. Since each VN request is significantly sparser and smaller than the resources available to the InPs, this result is unsurprising. However, it is possible that we may be observe significantly different behavior as the size of the VN request approaches total available resources of the first-hop InPs. We leave that experiment for future work.

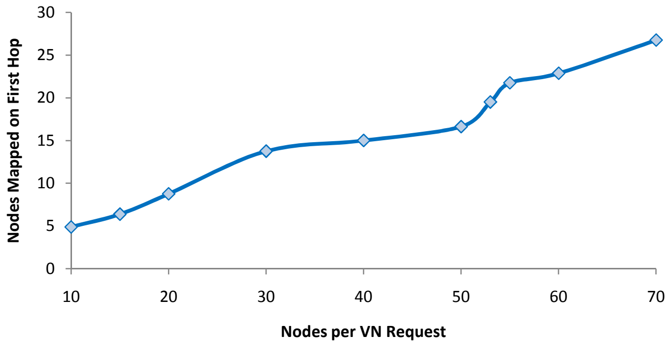


Fig. 6. Nodes mapped on first hop vs. VN request size: The number of nodes mapped by first-hop InPs increases linearly with the size of sparse VN requests (n nodes, $4n$ links).

E. Number of InPs Involved in a Successful Mapping

In the third experiment, we observe the number of InPs that are involved in a successfully satisfied VN request. As before, for this experiment, we only consider InPs that contribute substrate node resources to the VN mapping, and not InPs that simply reserved bandwidth as *relays*. We consider sparse VN requests, with an expected n nodes, and $4n$ links.

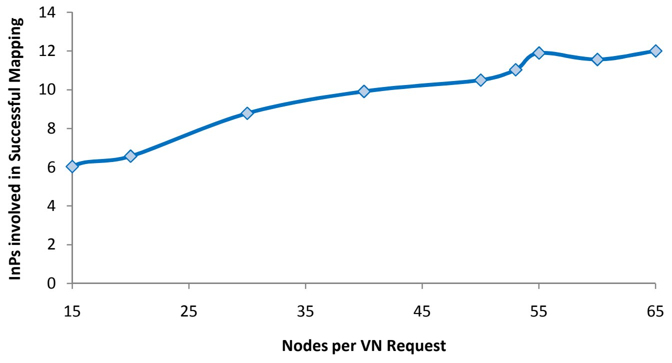


Fig. 7. InPs involved in a successful mapping vs. VN request size: We consider only InPs that have mapped nodes for this experiment and not *relay* InPs. The number of nodes mapped by first-hop InPs grows linearly with the size of the VN request up to 70 nodes. At 70 nodes and up, VN requests cannot be mapped with a search horizon limited to 12 hops.

As evident in Fig. 7, the number of InPs involved grows linearly with the size of the VN request. However, the 12-hop restriction prevents mapping VN requests with 70 nodes or more.

F. Success Rate and VN Request Size

We look at the success rate of VN request instances relative to increasing VN request sizes in the final experiment, where success rate is defined as $\frac{count(SUCCESS)}{count(SUCCESS)+count(FAIL)} * 100\%$. Currently in our experiments, failures are only observed once a VN request passes 12 hops. We believe this is an interesting measure to allow us to predict a reasonable quantity for the maximum hop count.

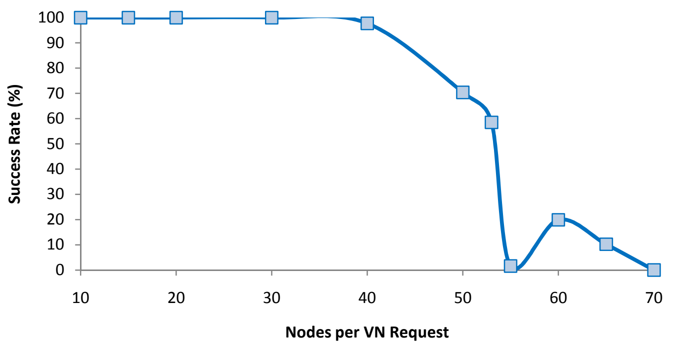


Fig. 8. Mapping success rate vs. VN request size: Small request sizes relative to InP resources are always mappable within 12 hops. As request size grows, the success rate decreases in a linear fashion. At about 70 nodes, no requests are successful. The anomaly with VN requests of roughly 55 nodes cannot be explained at this time.

In Fig. 8, it can be seen that below 40 nodes, all VN requests are successful in under 12 hops. As the size of the VN request increases from 40 to 70 nodes, the success rate gradually drops. At this time, we are unable to explain the significant dip at and around 55 nodes. We have found that we are able to reliably reproduce this result. At this time, we suspect it might be an issue with the greedy VN mapping algorithm.

VII. Discussion

Since PolyViNE is still in an early development stage, only a brief quantitative evaluation could be provided. In this section, we qualitatively discuss several open questions regarding the characteristics of PolyViNE.

A. Scalability

Scalability concerns in PolyViNE come from several fronts: size of the search space, dissemination time of location information, and storage of location and price information among others. As the number of InPs increases in the Controller Network, the amount of control traffic will increase even with tweaks proposed in this paper. Moreover, the size of stored location and path information will grow very quickly with more and more InPs joining the Controller Network. In that case, we can limit the number of stored paths to a certain destination based on some heuristics (e.g., keep only the top M paths and flush the rest after each update), but such loss can result in degraded embedding. Finally, the freshness of the location information is dependent upon the update frequency. If there are a large number of InPs, it will take longer to get updates from the farthest point of the Controller Network than it would take for smaller number of InPs.

B. Performance

Primary performance issues regarding PolyViNE include response time, embedding quality (e.g., whether or not embedded, price etc.), and overhead of InPs due to VN request relaying without actually taking part in the embedding itself.

1) Response time: Recursive processes, by definition, can go on for a long time in the absence of proper terminating conditions resulting in unsuitable response times. Combining iterative mechanism wherever possible and limiting the level of recursion at the expense of search completeness can improve the response time of PolyViNE. However, the question regarding suitable response time depends on the arrival rate and the average life expectancy of VN requests.

2) Embedding quality: The quality of a VN embedding in a distributed multi-InP environment is highly dependent on the individual policies enforced by the InPs participating in that embedding. Since PolyViNE uses a multi-InP bidding mechanism at every step, the price of an embedded VN request is likely to be very competitive within the search horizon. However, PolyViNE avoids flooding in the Controller Network by restricting the number of bidder InPs at each step, which can result in inadvertently rejected requests or higher embedding prices.

3) Overheads: InPs participating in a PolyViNE embedding will face major computation overheads while trying to map the VN request and minor communication overheads due to relaying of the rest of the request. Since for each VN embedding every InP in each step except for the winning bidder will fail to take part in the embedding, the overheads can be discouraging. We are working toward finding incentives for the InPs to partake in the embedding process.

C. Trust and reputation

Trust and reputation are well studied areas in computing literature in different contexts, unfortunately still without any robust scheme against breaches. Since each InP will always try to selfishly improve its own performance and will not expose its internal information, InPs can lie to or hide information from each other. From previous studies it is known that it is hard if not impossible to use mechanism design or game theoretic models to thwart such behaviors in a large scale distributed system [11]. Our solution against such behavior is the use of competitive bidding at each step of embedding so that the market price of any leased resource is exposed.

VIII. Related Work

The VN embedding problem, with constraints on both virtual nodes and virtual links, is known to be $\mathcal{NP}$-hard [4], [5]. As a result, a number of heuristic-based algorithms have appeared in the literature based on complete separation of the node mapping and the link mapping phases [4]–[6]. Moreover, in order to make the problem tractable, existing research has been restricting the problem space in different dimensions: [4], [6] consider the offline version of the problem; [6] ignores node requirements; [4], [6] assume infinite capacity in substrate nodes and links to obviate admission control; and [6] focuses on specific VN topologies. Recently, Chowdhury et al. have proposed a pair of algorithms that consider all these constraints and provide improved performance through increased correlation between the two phases of VN embedding [7]. However, all previous work addresses VN embedding as an intra-domain problem involving only one InP and take advantage of having a centralized controller to embed the complete VN request. Even though the authors in [12] have proposed a multi-agent based distributed algorithm for VN embedding by allowing each substrate node to take independent decisions, they restrict themselves within a single InP. To the best of our knowledge, PolyViNE is the first of its kind to address VN embedding in a distributed multi-InP scenario.

While inter-domain VN embedding is more or less of an uncharted territory, inter-domain lightpath provisioning [9], [13] as well as cross-domain QoS-aware path composition [10], [14] are well studied areas. UCLP [13] allows users to dynamically compose, modify, and tear down lightpaths across domain boundaries and over heterogeneous networking technologies (e.g., SONET/SDH, GMPLS etc.). Xiao et al. have shown in [14] that QoS-assured end-to-end path provisioning can be solved by reducing it to the classic k-MCOP (k-Multi Constrained Optimal Path) problem. iREX architecture [10], on the other hand, uses economic market-based mechanisms to automate inter-domain QoS policy enforcement through negotiation between participating domains. PolyViNE is similar to iREX in its allowance of intra-domain policy-enforcement and in using market-based mechanisms, but iREX is concerned about mapping simple paths whereas PolyViNE embeds more complicated VN requests. PeerMart [15], [16] is another auction-based marketplace for resource trading in an NVE, but it basically deals only with virtual links.

The geographic location representation and related information dissemination protocol proposed in PolyViNE is inspired by the previous proposals of geographic addressing and routing in IPv6 networks [17], [18] as well as the predominant global routing protocol in the Internet, BGP [19]. However, unlike these works, PolyViNE does not use the information for addressing or routing purposes; rather it uses the location information to find candidate InPs that will be able to embed part or whole of the remaining unmapped VN request. Moreover, such location information is disseminated between and stored in Controllers instead of border routers as in BGP or GIRO [18]. The concepts of Controllers in InPs and Controller Network connecting multiple InPs' Controllers are discussed in the iMark framework [8].

IX. Conclusion

Embedding of end-to-end VNs across multiple InP domains is the first step toward enabling network virtualization for the next-generation networking paradigm. However, this crucial problem has remained virtually untouched in the literature over the years due to its complex nature. In this paper, we have provided a formal definition of the inter-domain VN embedding problem and performed a requirements analysis for the first time in the literature.

Our findings have led us to PolyViNE, a novel policy-based inter-domain VN embedding framework for NVE. PolyViNE allows embedding of end-to-end VNs in a distributed and decentralized manner by promoting global competition in the presence of local autonomy. It addresses the complexities of distributed VN embedding process by providing a loose framework that is guided by the policies of individual InPs and SPs, and coordinated by a distributed protocol. We have laid down the workflows of InPs and SPs throughout the PolyViNE embedding process and identified the most crucial stage in the InP workflow, VN request forwarding. In this respect, we have proposed a hierarchical addressing system, COST and a location dissemination protocol, LAP that jointly allow InPs to make informed forwarding decisions. Finally, we have evaluated PolyViNE performance characteristics through simulation.

However, several issues remain unresolved in this work. In the future we would like to address issues such as pricing models, reputation management, and incentives for InP truthfulness. Relative advantages and disadvantages of contrasting choices (e.g., recursive vs iterative forwarding, values of k^{SP} and k^{InP} etc.) in different stages of InP workflow should also be scrutinized. Most importantly, the scalability and performance characteristics of PolyViNE must be further evaluated through larger simulations and distributed experiments with heterogeneous mix of intra-domain VN embedding algorithms and policies to establish PolyViNE as a viable inter-domain VN embedding solution.

References

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, 2005.

[2] J. Turner and D. Taylor, "Diversifying the Internet," in *IEEE GLOBECOM*, vol. 2, 2005.

[3] N. Feamster, L. Gao, and J. Rexford, "How to lease the Internet in your spare time," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 61–64, 2007.

[4] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *IEEE INFOCOM*, 2006.

[5] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM CCR*, vol. 38, no. 2, pp. 17–29, April 2008.

[6] J. Lu and J. Turner, "Efficient mapping of virtual networks onto a shared substrate," Washington University, Tech. Rep. WUCSE-2006-35, 2006.

[7] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *IEEE INFOCOM*, 2009.

[8] N. M. M. K. Chowdhury, F.-E. Zaheer, and R. Boutaba, "iMark: An identity management framework for network virtualization environment," in *IEEE IM*, 2009.

[9] Q. Liu, N. Ghani, N. S. V. Rao, A. Gumaste, and M. L. Garcia, "Distributed inter-domain lightpath provisioning in the presence of wavelength conversion," *Computer Communications*, vol. 30, no. 18, pp. 3662–3675, 2007.

[10] A. Yahaya, T. Harks, and T. Suda, "iREX: Efficient automation architecture for the deployment of inter-domain QoS policy," *IEEE TNSM*, vol. 5, no. 1, pp. 50–64, March 2008.

[11] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Experiences applying game theory to system design," in *ACM SIGCOMM workshop on Practice and Theory of Incentives in Networked Systems (PINS '04)*, 2004, pp. 183–190.

[12] I. Houidi, W. Louati, and D. Zeghlache, "A distributed virtual network mapping algorithm," in *IEEE ICC*, 2008, pp. 5634–5640.

[13] E. Grasa, G. Junyent, S. Figuerola, A. Lopez, and M. Savoie, "UCLPv2: A network virtualization framework built on web services," *IEEE Communications Magazine*, vol. 46, no. 6, pp. 126–134, March 2008.

[14] J. Xiao and R. Boutaba, "QoS-aware service composition and adaptation in autonomic communication," *IEEE JSAC*, vol. 23, no. 12, pp. 2344–2360, December 2005.

[15] D. Hausheer and B. Stiller, "Auctions for virtual network environments," in *Workshop on Management of Network Virtualisation*, 2007.

[16] ——, "PeerMart: Decentralized auctions for bandwidth trading on demand," *ERCIM News*, no. 68, pp. 42–43, January 2007.

[17] T. Hain., "Application and use of the IPv6 provider independent global unicast address format," Internet Draft (http://tools.ietf.org/html/draft-hain-ipv6-pi-addr-use-10.txt), August 2006.

[18] R. Oliveira, M. Lad, B. Zhang, and L. Zhang, "Geographically informed inter-domain routing," in *IEEE ICNP*, October 2007, pp. 103–112.

[19] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271 (Draft Standard), January 2006.