PolyViNE: Policy-based Virtual Network Embedding Across Multiple Domains

Mosharaf Chowdhury *
Computer Science Division
University of California, Berkeley, CA, USA
mosharaf@cs.berkeley.edu

Fady Samuel, Raouf Boutaba
Cheriton School of Computer Science
University of Waterloo, ON, Canada
{fsamuel, rboutaba}@cs.uwaterloo.ca

ABSTRACT

Intra-domain virtual network embedding is a well studied problem in the network virtualization literature. For most practical purposes, however, virtual networks (VNs) must be provisioned across heterogeneous administrative domains managed by multiple infrastructure providers (InPs).

In this paper 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 service providers (SPs), i.e., VN owners. 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 using quantitative and qualitative evaluations.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks

General Terms

Algorithms; Design

Keywords

Inter-domain Virtual Network Embedding; Policy-based Resource Allocation; Decentralized Embedding; Network Virtualization

1. INTRODUCTION

Network virtualization has gained significant attention in recent years as a means to support multiple coexisting virtual networks (VNs) on top of shared physical infrastructures [1, 3, 6, 16]. The first step toward enabling network virtualization is to instantiate such VNs by embedding[1] VN requests onto substrate networks. But the VN embedding problem, with constraints on virtual nodes and virtual links, is known to be $\mathcal{NP}$-hard [19, 21]. Several heuristics [5, 9, 11, 19, 21] have been proposed to address this problem in the single infrastructure provider (InP) scenario. However, in realistic settings, VNs must be provisioned across heterogeneous administrative domains belonging to multiple InPs to deploy and deliver services end to end.

One of the biggest challenges in end-to-end VN embedding is to organize the InPs 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 according to its internal administrative policies while maintaining global connectivity through mutual agreements with other InPs.

Moreover, InPs (i.e., network operators) are notoriously known for their secrecy of traffic matrices and topology information. As a result, existing embedding algorithms that assume complete knowledge of the substrate network are not applicable in this scenario. 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.

Finally, there will be constant tussles between SPs and InPs on multiple levels:

- 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.

- SPs are also interested in getting their requirements satisfied while minimizing their expenditure. Tussles might arise between SPs and InPs when each party try to optimize their utility functions.

Any inter-domain VN embedding mechanism must enforce proper incentives and mechanisms to address these tussles.

In this paper, we introduce 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

*This work was completed when the author was with the University of Waterloo.

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

VISA 2010, September 3, 2010, New Delhi, India.

the same time. PolyViNE introduces a distributed protocol that coordinates the participating InPs and ensures competitive pricing through repetitive bidding at every step of the embedding process.

We do not claim PolyViNE to be the best or the only way of performing end-to-end VN embedding. However, to the best of our knowledge, this is the first foray into this unexplored domain in the context of network virtualization, and we believe this problem to be absolutely critical in realizing network virtualization for most practical purposes.

The rest of the paper is organized as follows. Section 2 formally defines the inter-domain VN embedding problem. In Section 3 we describe the design choices and the distributed embedding protocol used by PolyViNE, followed by a discussion of its enabling technologies in Section 4. Section 5 and Section 6 respectively provide preliminary quantitative and qualitative evaluations of PolyViNE. We discuss related work in Section 7. Finally, Section 8 concludes the paper with a discussion on possible future work.

2. PROBLEM FORMULATION

The intra-domain VN embedding problem is well-defined in the literature [5, 9, 11, 19, 21]. In this section, we formally define the inter-domain VN embedding problem. For simplicity, we avoid intra-domain aspects (e.g., node and link attributes) wherever we see fit.

2.1 Substrate Networks and the Underlay

We consider the underlay to be comprised of D substrate networks (Figure 1(a)), and we model each substrate network controlled by the i-th InP[2] ($1 \leq i \leq D$) 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 network has a (centralized or distributed) logical Controller [4] that performs administrative/control functionalities for that InP. Finally, $A_i^S (\subset N_i^S)$ denotes the set of border nodes [4] 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.

We denote the underlay (shown in Figure 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 containing border nodes across all InPs ($i \leq i \leq D$) 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 [10]. All border nodes belonging to a single InP are collapsed to one single node corresponding to that InP (Figure 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 (Figure 1(d)) referring to the controller network [4], 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.

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

2.2 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 standard terms [5,19]. Figure 1(e) depicts a VN request with virtual node and 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 $loc\left(n^V\right)$ [5], which can be interpreted as the preferred geolocation of that virtual node. Figure 1(f) shows the substrate nodes within the preferred geolocation for each virtual node using dashed vertical boxes. Similar to InPs, SPs can also provide a set of policies/preferences $\mathcal{P}^V$ for a VN request to dictate certain characteristics. For example, the VN request in Figure 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 $InP\#4$.

2.3 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, G^C.

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 Figure 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 $\mathcal{F} : G_k^V \rightarrow N_M^V$ (Figure 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. Figure 1(i) shows a possible InP-level embedding for the VN request shown in Figure 1(e). Note that, $InP\#2$ has not embedded any virtual node but is still in the embedding by being in an inter-domain virtual link.

3. POLYVINE OVERVIEW

In this section, we discuss PolyViNE design decisions, explain its workflow, and describe the distributed protocol that coordinates the PolyViNE embedding process.

3.1 Design Choices

We have made the following 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.

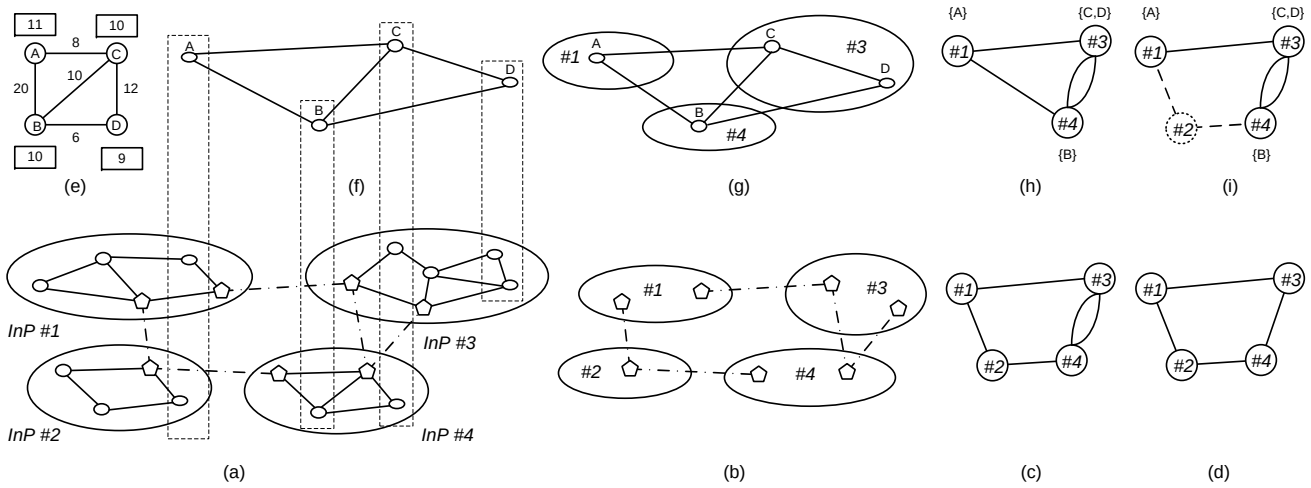

Figure 1: Overview of the inter-domain VN embedding process.

3.1.1 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).

3.1.2 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 [13, 18].

3.1.3 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.

3.2 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. Figure 2 provides a high level depiction of the inter-domain VN embedding process.

3.3 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.

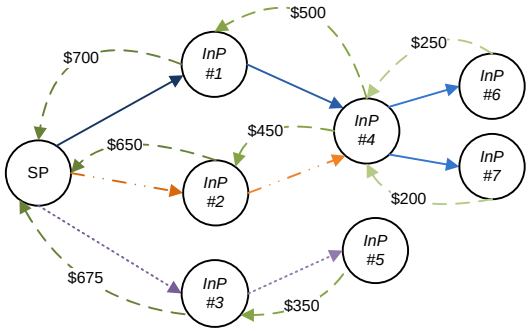

Figure 2: Propagation of multiple instances of the same VN request in the controller network throughout the embedding process. The dashed lines demonstrate the back-propagation of accumulated prices toward the SP.

- **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. Here, $InP\#$ 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.

3.4 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. Figure 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* or *FAILURE* messages. Once the SP selects an embedding, it proceeds toward instantiating its VN by sending *ACK* messages to the InPs involved in the selected embedding.

3.5 InP Workflow

While an SP's workflow is straightforward with a single decision at the end, it shifts much more work to the InPs. 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.

3.5.1 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.

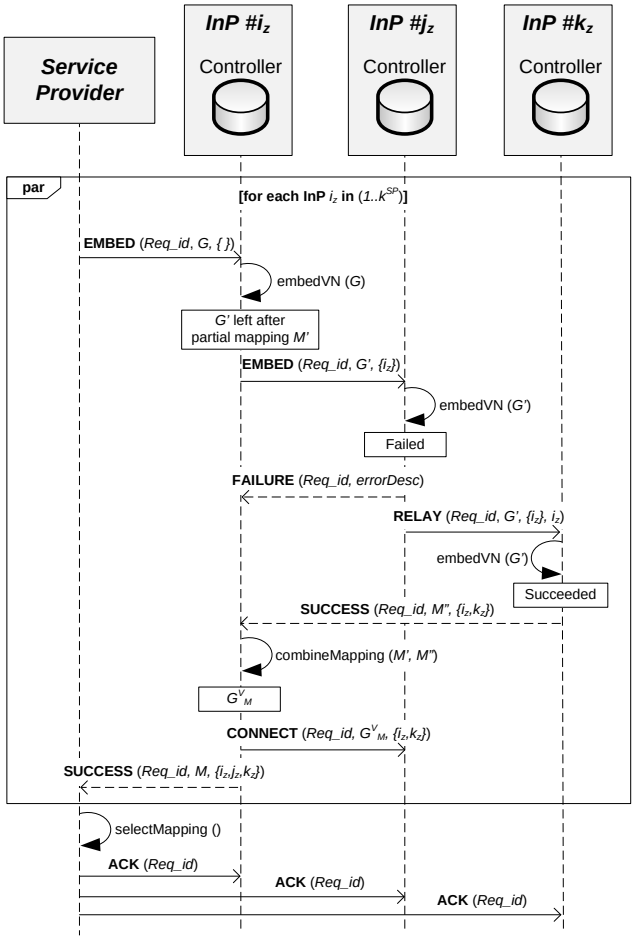

Figure 3: Sequence diagram showing PolyViNE embedding process in action.

In order to decide which part of a VN request to embed, if at all, the InP can use existing intra-domain VN embedding algorithms [5, 19] that can identify conflicting resource requirements in a VN request[4].

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 part or whole of the VN request. In that case, it will *RELAY* the VN request to that InP. In Figure 3, $InP\#j_z$ is relaying the VN request G' to $InP\#k_z$.

3.5.2 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 Figure 3 is forwarding the unmapped

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

[4]This can be done by looking into the output of the linear programs used in both [19] and [5] without modifying the actual algorithms presented in those work.

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 Figure 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 forwarding decision is a non-trivial one and requires careful consideration. We believe that instead of blindly forwarding based on some 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 are presented in the next section.

3.5.3 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 of this back-propagation of *SUCCESS* and *FAILURE* messages, the sender InP can 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 (Figure 2).

4. 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 together 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) to perform this task. Careful readers will notice in the following that COST and LAP are significantly influenced by BGP.

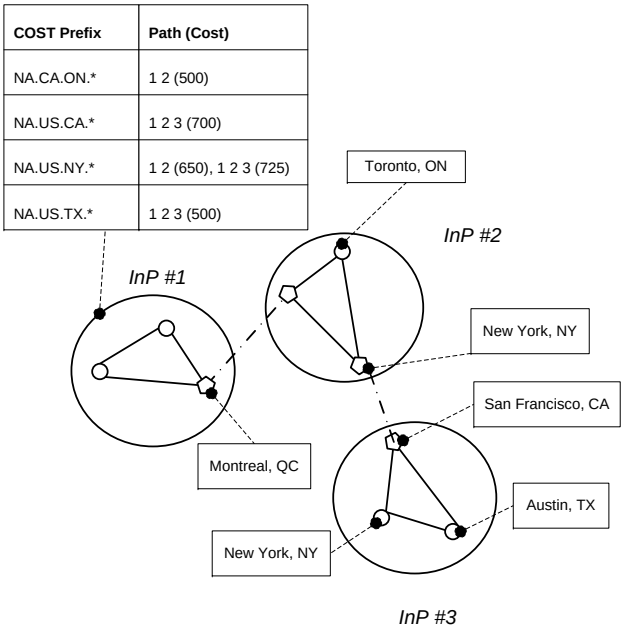

Figure 4: LAP database at InP #1. InP #1 has two choices to forward to an InP with a node in New York state.

4.1 COST Addressing Scheme

As outlined in the problem formulation (Section 2), 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 similar to [14] 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 affect its reputation among other InPs.

4.2 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 pro-

gressing 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 view of the resources in the underlay in each Controller's LAP database.

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. Note 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 them. 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. Figure 4 shows an example LAP database.

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 be honest when announcing pricing estimates? We believe that a reputation metric – indicating long-term accuracy of an InP's pricing estimate to the actual cost of establishing a VN request – is necessary to remedy this situation. We would like to integrate such a reputation metric within LAP to allow dissemination of path vectors attributed with corresponding prices and overall reputation score of the InPs on the paths. An InP will then be able to use pricing and reputation scores to rank multiple paths to a common destination to make a forwarding decision.

5. NUMERICAL EVALUATION

We have written a 3000 line multi-threaded C++ simulator that allows independent responses from various entities in the controller network. 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 are yet to be implemented.

We have performed three preliminary experiments. In our current experiments, each InP performs a naive greedy node and link mapping to embed the largest possible connected component. 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. To restrict the search space, we drop the VN request and report failure after 12 hops between InPs.

Unless otherwise specified, we have used the following settings. 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 CPU capacity uniformly chosen from 1 to 100 CPU units, and each link has a maximum bandwidth capacity of 100 bandwidth units. VN requests vary between experiments in terms of the average numbers of virtual nodes and virtual links.

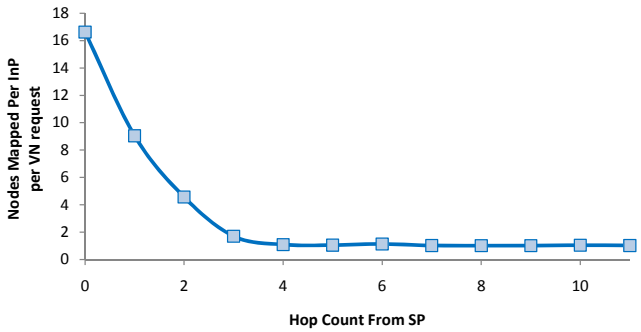

Figure 5: Nodes mapped by each InP per VN request vs. hop count.

5.1 Node Mapping and Hop Count

In the first experiment, we investigate the number of virtual nodes mapped by each InP located at 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. Figure 5 confirms this intuition. For this particular experiment, we have used VN requests with 50 nodes and 200 links on the average.

We observe an exponential decay in the number of nodes mapped as the hop count increases. At this time, we are uncertain whether this pattern is a function of the random graphs we generated or the simple greedy algorithm we use for mapping or both. 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.

5.2 Node Mapping and VN Request Size

Next, 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).

Figure 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 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.

5.3 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 con-

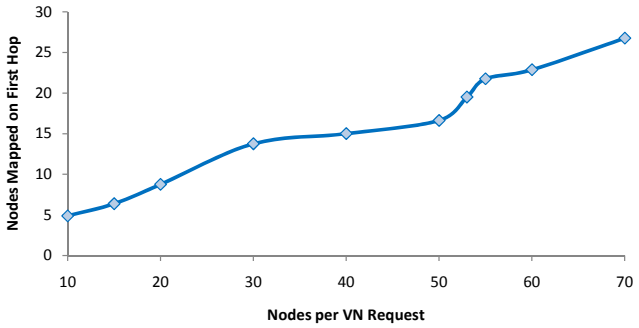

Figure 6: Nodes mapped on first hop vs. VN request size.

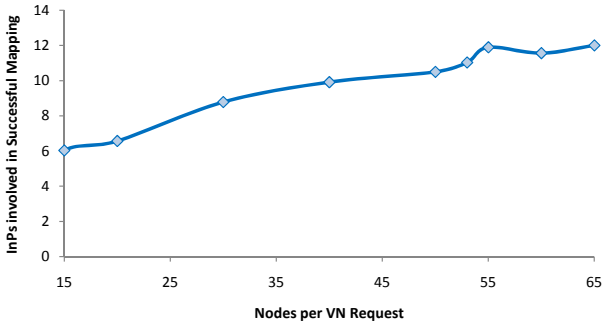

Figure 7: Number of InPs involved in a successful mapping with increasing VN request size.

tribute 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. As evident in Figure 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.

6. DISCUSSION

6.1 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 the 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. 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 and the total number of InPs in the controller network.

6.2 Performance

6.2.1 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.

6.2.2 Embedding quality

The quality of a VN embedding in a distributed multi-InP environment is highly dependent on the individual policies enforced by the participating InPs. 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.

6.2.3 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.

6.3 Trust and reputation

Since each InP will 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 to use mechanism design or game theory to thwart such behaviors in a large scale distributed system [12]. Our solution against such behavior is the use of competitive bidding at each step of embedding to expose the market price of any leased resource.

7. RELATED WORK

The VN embedding problem, with constraints on both virtual nodes and virtual links, is known to be $\mathcal{NP}$-hard [19,21]. A number of heuristics have appeared in the literature based on the complete separation of the node mapping and the link mapping phases [11, 19, 21]. Existing research has also been restricting the problem space in different dimensions: [11,21] consider the offline version of the problem; [11] ignores node requirements; [11, 21] assume infinite capacity in substrate nodes and links to obviate admission control; and [11] focuses on specific VN topologies. Chowdhury et al. [5] proposed a pair of algorithms that provide improved performance through increased correlation between the two phases of VN embedding, while [9] proposed a graph isomorphism-based integrated solution that can take exponential time in the worst case. All these algorithms address VN embedding as an intra-domain problem and take advantage of a centralized embedding entity.

Recently proposed V-Mart [20] framework approaches the inter-domain VN embedding problem using an auction-based model, where the SP performs the partitioning task using heuristics for simplification. As a result, V-Mart cannot enable local and inter-InP policy enforcement and fine-grained resource management.

Unlike inter-domain VN embedding, inter-domain lightpath provisioning [2, 10] as well as cross-domain QoS-aware path composition [17, 18] are well studied areas. UCLP [2] 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 [17] 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 [18], 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 [8] is another auction-based marketplace for resource trading in a network virtualization environment, 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 [7, 14] as well as the predominant global routing protocol in the Internet, BGP [15]. 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 [14]. The concepts of Controllers in InPs and controller network connecting multiple InPs' Controllers are discussed in the iMark framework [4].

8. CONCLUSIONS AND FUTURE WORK

In this paper we have formally defined the inter-domain VN embedding problem and presented PolyViNE – a novel policy-based inter-domain VN embedding framework – to address it. PolyViNE allows embedding of end-to-end VNs in a distributed and decentralized manner by promoting global competition in the presence of local autonomy. 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. We have also presented preliminary performance characteristics of PolyViNE through simulation.

In the future we would like to address issues such as pricing models, InP interactions, 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}) in different stages of InP workflow should also be scrutinized. Finally, the scalability, stability, and performance characteristics of PolyViNE require further studies through larger simulations and distributed experiments with a heterogeneous mix of intra-domain VN embedding algorithms and policies.

Another interesting direction of research for this problem would be to model it as a distributed constrained optimization problem (DCOP) and to try to solve that with minimal information exchange between InPs.

Acknowledgments

We would like to thank the anonymous reviewers for their comments and suggestions. This work was jointly supported by the Natural Science and Engineering Council of Canada (NSERC) under its Discovery program, Cisco Systems, and WCU (World Class University) program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0).

9. REFERENCES

[1] T. Anderson et al. Overcoming the Internet impasse through virtualization. *Computer*, 38(4):34–41, 2005.

[2] R. Boutaba et al. Lightpaths on demand: A web-services-based management system. *IEEE Communications Magazine*, 42(7):101–107, July 2004.

[3] N. M. M. K. Chowdhury and R. Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862–876, 2010.

[4] N. M. M. K. Chowdhury et al. iMark: An identity management framework for network virtualization environment. In *IEEE IM*, 2009.

[5] N. M. M. K. Chowdhury et al. Virtual network embedding with coordinated node and link mapping. In *IEEE INFOCOM*, 2009.

[6] N. Feamster et al. How to lease the Internet in your spare time. *ACM SIGCOMM Computer Communication Review*, 37(1):61–64, 2007.

[7] 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.

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

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

[10] Q. Liu et al. Distributed inter-domain lightpath provisioning in the presence of wavelength conversion. *Computer Communications*, 30(18):3662–3675, 2007.

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

[12] R. Mahajan et al. Experiences applying game theory to system design. In *SIGCOMM PINS*, pages 183–190, 2004.

[13] L. Mekouar et al. Incorporating trust in network virtualization. In *IEEE CIT*, 2010.

[14] R. Oliveira et al. Geographically informed inter-domain routing. In *IEEE ICNP*, 2007.

[15] Y. Rekhter et al. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006.

[16] J. Turner and D. Taylor. Diversifying the Internet. In *IEEE GLOBECOM*, volume 2, 2005.

[17] J. Xiao and R. Boutaba. QoS-aware service composition and adaptation in autonomic communication. *IEEE JSAC*, 23(12):2344–2360, December 2005.

[18] A. Yahaya et al. iREX: Efficient automation architecture for the deployment of inter-domain QoS policy. *IEEE TNSM*, 5(1):50–64, March 2008.

[19] M. Yu et al. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM CCR*, 38(2):17–29, 2008.

[20] F. Zaheer et al. Multi-provider service negotiation and contracting in network virtualization. In *IEEE/IFIP NOMS*, 2010.

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