Towards Efficient Virtual Network Embedding Across Multiple Network Domains

Meng Shen*, Ke Xu*, Kun Yang†, and Hsiao-Hwa Chen‡
*Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science, Tsinghua University, P. R. China
†School of Computer Science & Electronic Engineering, University of Essex, UK
‡Department of Engineering Science, National Cheng Kung University, Taiwan
{shenmeng, ke.xu}@csnet1.cs.tsinghua.edu.cn; kunyang@essex.ac.uk; hshwchen@mail.ncku.edu.tw

***Abstract*—Network virtualization provides a promising way to run multiple virtual networks (VNs) simultaneously on a shared infrastructure. It is critical to efficiently map VNs onto substrate resources, which is known as the VN embedding problem. Most existing studies restrict this problem in a single substrate domain, whereas the VN embedding process across multiple domains (*i.e.*, inter-domain embedding) is more practical, because a single domain rarely controls an entire end-to-end path. Since infrastructure providers (InPs) are usually reluctant to expose their substrate information, the inter-domain embedding is more sophisticated than the intra-domain case.**

In this paper, we develop an efficient solution to facilitate the inter-domain embedding problem. We start with extending the current business roles by employing a broker-like role, virtual network provider (VNP), to make centralized embedding decisions. Accordingly, a reasonable information sharing scheme is proposed to provide VNP with partial substrate information meanwhile keeping InPs' confidential information. Then we formulate the embedding problem as an integer programming problem. By relaxing integer constraints, we devise an inter-domain embedding algorithm to handle online VN requests in polynomial time. Simulation results show that our solution outperforms other counterparts and achieves 80%-90% of the benchmarks in an *ideal* scenario where VNP has complete knowledge of all substrate information.

I. Introduction

Network virtualization has emerged as a powerful way to allow multiple virtual networks coexist in a shared infrastructure. This is done by decoupling infrastructure providers (InPs), who are responsible for managing the connectivity of the substrate networks, from service providers (SPs), who are dedicated to providing services to end users [1]. Recent advances in network equipment virtualization [19] and software-defined networking (SDN) [20] open possibilities for implementing such an Infrastructure as a Service (IaaS) model, where service providers deploy their protocols on customized virtual networks and pay for the resource usage.

A virtual network (VN) is a collection of virtual nodes (*e.g.*, routers) interconnected via a set of virtual links. Each VN, with requirement constraints on its virtual nodes and links (*e.g.*, CPU and link bandwidth [7]), should be mapped onto specific physical nodes and links in the substrate network, which is known as the VN embedding problem. Since multiple VNs share the same substrate resources, an efficient embedding algorithm is crucial to increase the utilization of the substrate resources and the revenue of InPs [8]. In this paper we use *embedding* and *mapping* interchangeably.

978-1-4799-4852-9/14/$31.00

Many algorithms and mechanisms have been proposed to make efficient embedding decisions, *e.g.*, [4, 5, 7–10]. These studies restrict the embedding problem within the same substrate domain, which is referred to as the ***intra-domain*** embedding hereafter. However, an individual InP rarely controls an entire end-to-end path [1], service providers thus desire virtual networks across multiple substrate domains so as to offer value-added services to end users. For example, Voice over IP (VoIP) needs to run a dedicated virtual network that provides services with guaranteed performance (*e.g.*, latency) to users in multiple network domains; live-streaming video providers require high-throughput virtual networks that deliver real-time video resources to their geographically distributed private servers. In this paper, we refer to the VN embedding across multiple substrate domains as the ***inter-domain*** embedding.

We argue that the inter-domain embedding is greatly different from the intra-domain one, and thus algorithms proposed for the latter cannot be directly applied to the former case. Within a single domain, both the detailed substrate network topology and VN requests are visible to an InP, who, in turn, makes optimal embedding decisions according to its operational goals (*e.g.*, minimizing the embedding cost of the current VN request). In the inter-domain scenario, however, InPs are typically reluctant to share substrate topology with each other. Therefore, no individual InP can make a global embedding decision for a cross-domain VN request.

A straight-forward solution to the inter-domain embedding can be described as follows: a service provider, who requires a cross-domain VN, should first coordinate with each of the potential InPs and then decide to require which InP to accommodate which component of the entire VN. It makes the service provider involved in multiple bilateral coordinations and negotiations, and thus sacrifices a major benefit of decoupling these two roles. In particular, it is even more costly for SPs who need to quickly set up short-term VNs.

In this paper, we develop an efficient solution to the inter-domain embedding, which frees service providers from sophisticated negotiations, while protecting infrastructure providers from revealing their confidential information. The major contribution of this paper, which comes along with addressing

the challenges of the inter-domain embedding, is two-fold and summarized as follows.

We introduce a new reasonable information sharing scheme to facilitate the embedding process. To avoid SPs negotiating with individual InPs, we extend the current business roles (*i.e.*, InPs and SPs) by employing a broker-like role named virtual network provider (VNP), who decomposes VN requests received from its customers (*i.e.*, SPs) and sends each component to a corresponding InP for an intra-domain embedding. To enable the VN decomposition conducted by VNP, a new information sharing scheme is designed, which requires InPs involved to provide VNP with partial information of their substrate resources. This scheme does not reveal the detailed topology of each domain and thus protect InPs' confidential information.

We develop an inter-domain embedding algorithm to handle online VN requests. The major difficulty in the inter-domain embedding is to enable VNP to make an efficient VN request decomposition with limited substrate information. To address this challenge, we employ an estimation-based approach to infer the unknown intra-domain topologies and create an augmented network to coordinate the node and the link mapping phases. The VN decomposition is formulated as a binary integer programming with an objective of minimizing the total decomposition cost. Since solving the integer programming is known to be computationally intractable for large-scale networks [12], we relax the integer constraints to derive a linear solution and then introduce heuristic approaches to approximate the original binary solution. An inter-domain embedding algorithm is devised accordingly to handle online VN requests in polynomial time.

We conduct extensive simulations to evaluate the performance of the proposed algorithm and compare it with other algorithms in terms of several metrics, such as the VN request acceptance ratio. Results show that for all the metrics, the proposed algorithm achieves 80%-90% of the benchmarks in an *ideal* scenario where VNP is assumed to obtain all substrate information of InPs. In addition, we experimentally explore the impact of different information sharing strategies on algorithm performance, which shows InPs have profit incentives to adopt the proposed information sharing scheme.

The rest of the paper is organized as follows. We describe the inter-domain embedding in Section II, and then present the network model and mathematical formulations in Section III and Section IV, respectively. The algorithm derived from the formulation is presented in Section V. Section VI exhibits the simulation results. We summarize related work in Section VII and conclude this paper in Section VIII.

II. Problem Description

In this section, we extend the current business roles and propose a new information sharing scheme accordingly. Then, the high-level inter-domain embedding process is described.

A. Business Roles

With the current business roles in the intra-domain embedding, an SP, who needs to establish a cross-domain VN, suffers from the exhausted bilateral negotiations with InPs. To make the embedding process more efficient, we extend the two-role business model by employing a broker-like role named VNP. The responsibility of each role in the inter-domain VN embedding is defined as follows.

- Service Provider, who sends VN requests with specific virtual resource requirements to VNP and then provides end-to-end services via these virtual networks.
- Virtual Network Provider, who decomposes each VN request from SPs into multiple *components* and sends them to corresponding InPs for intra-domain embedding.
- Infrastructure Provider, who manages the physical network and allocates virtual resources (*e.g.*, virtual router instances) to requests from VNP.

The role of VNP is first introduced in [3], but with a different responsibility. For the convenience of presentation, we assume that a unique VNP can communicate with all InPs. To take advantage of the layered business roles, we also assume a simple charging model, where 1) an SP makes a payment to VNP in proportion to the virtual resources it requires and thus does not care about the further processing by other roles (*i.e.*, VNP and InPs), and 2) VNP will be charged by an InP for a specific request component, which depends only on the desired virtual resources of the component.

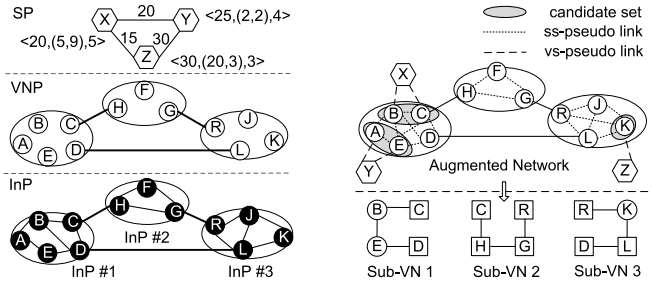
B. Information Sharing Scheme

In order to offer end-to-end services, an SP needs to design a VN that covers areas where their private servers or potential users are located. Accordingly, this VN request is sent to VNP with specific virtual topology attributes, including node requirements (*e.g.*, capacities and desired locations) and link requirements (*e.g.*, bandwidths). In addition, each virtual node is also associated with a distance constraint indicating how far it can be mapped from its desired location. In practice, this distance constraint can be used to approximately bound the transmission delay between the actual and desired locations.

A simple VN request with three virtual nodes represented by hexagons is shown in Fig. 1(a), where the numbers over the links represent their requirements. The triple beside each virtual node represents its required capacity, desired location and the distance constraint, respectively.

According to the above business model, the revenue VNP gains from accepting a VN request is fixed. Thus, VNP aims to get the VN request decomposed with the minimum expenditure. For instance, the price of an intra-domain link offered by InPs is usually much lower than that of an inter-domain one, so VNP prefers to embed as many virtual nodes as possible within the same domain. To this end, VNP is eager to get more information of the substrate resources in each domain, such as the available capacities of substrate resources. However, InPs are reluctant to expose their topology details.

To address this tussle among VNP and InPs, we define a information sharing scheme, which considers the concerns of VNP and InPs simultaneously. Three types of resource information in each substrate domain are provided by the InP (*i.e.*, operator of this domain) to VNP, including:



(a) Information obtained by each role (b) VN decomposition

Fig. 1. An example illustrating the inter-domain VN embedding process.

- Node: its location (*e.g.*, geographic coordinate), available capacity (*e.g.*, the number of available virtual router instances) and unit price (*e.g.*, $\$/(\text{instance}\cdot\text{hour})$)
- Inter-domain link: its vertices, available capacity (*e.g.*, bandwidth) and unit price (*e.g.*, $\$/(\text{bps}\cdot\text{hour})$).
- Intra-domain link: a *length*-based price for connecting any two nodes in its domain (*e.g.*, $\$/(\text{bps}\cdot\text{hour}\cdot\text{length})$).

The information disclosed by InPs is determined based on three considerations. First, the intra-domain links remain under cover so as to protect the detailed topology of each InP from VNP and its competitors. As shown in Fig. 1(a), VNP just obtains a partial view of the overall substrate topology (*i.e.*, isolated substrate nodes and the inter-domain links). Second, providing prices of all resources enables VNP to calculate the overall cost of decomposing a VN request. Meanwhile, VNP is able to ensure that the decomposed components of a VN request can be connected via the inter-domain links, though not all of these components might be successfully embedded by their corresponding InPs. Finally, since multiple VN requests compete for limited substrate resources, VNP and InPs can benefit from disclosing available capacities of physical resources (*i.e.*, substrate nodes and inter-domain links). An experimental validation is shown in Section VI, which is regarded as an explicit incentive for InPs to provide such information.

C. Inter-domain VN Embedding Process

Based on the predefined business model and the information sharing scheme, the high-level inter-domain VN embedding process is described by the following steps.

Step 1: VN request decomposition. Upon receiving a VN request from SP, VNP should determine how to decompose it into multiple components so as to minimize the total cost. To this end, VNP first conducts the node pre-mapping that aims to associate each virtual node with a candidate set consisting of substrate nodes that meet its requirements. Take the VN request in Fig. 1(a) for example, the candidate set of virtual node X consists of two substrate nodes (*i.e.*, B and C) as shown in Fig. 1(b). Since the placement of virtual nodes will affect the path selection for virtual links, the costs of the node and the link mapping are closely inter-dependent. In addition, the limited substrate information obtained by VNP is inadequate to determine an entire path for each virtual link, because the intra-domain topologies are unavailable.

To address these challenges, we introduce an augmented network in Fig. 1(b), which provides VNP with estimation-based intra-domain topologies and coordinates the node mapping and the link mapping together when making decomposing decisions. We assume a full-mesh topology for each domain and thus connect each pair of the substrate nodes within the same domain via the *ss-pseudo* links. With the given length-based unit prices of intra-domain links, VNP is able to calculate the price of each ss-pseudo link (*e.g.*, $\$/(\text{bps}\cdot\text{hour})$). Then the virtual nodes are connected with their respective candidate nodes via the *vs-pseudo* links. Both the ss-pseudo and vs-pseudo links are associated with infinite capacities.

With the augmented network, mapping of each virtual link with bandwidth constraints is equivalent to finding a path with sufficient residual bandwidth. By introducing binary constraints, we can ensure that each virtual node is mapped onto one unique substrate node. Due to the coexisting node and link constraints, the VN embedding problem is NP-hard [7]. Even when all virtual nodes are pre-located, embedding of virtual links is still NP-hard [15]. We leave mathematical formulations in Section IV.

At the end of this step, a VN request is decomposed into several sub-VN requests, each of which is sent to a corresponding InP. Assume that the minimum-cost mapping of the VN request in Fig. 1(a) corresponds to the node mapping $\{X \mapsto B,\ Y \mapsto E,\ Z \mapsto K\}$ and the link mapping $\{(X,Z) \mapsto (B,C,H,G,R,K),\ (Y,Z) \mapsto (E,D,L,K),\ (X,Y) \mapsto (B,E)\}$. Then the entire request is decomposed into 3 sub-VN requests, where the nodes in squares are used to identify links and thus have no capacity requirements.

Step 2: Sub-VN request processing. Each InP involved independently gets its portion (*i.e.*, a sub-VN request) embedded according to its own strategies. Thus, the problem is reduced to a conventional intra-domain embedding. Note that the mapping from the nodes in a sub-VN to the substrate nodes is already determined by VNP, so an InP only needs to map the links in the sub-VN request. Each InP responds to VNP with a positive (*e.g.*, successful) or negative (*e.g.*, failed) feedback indicating the embedding result of its portion.

Step 3: Substrate resource allocation. After receiving feedbacks from all InPs, VNP will reply them with a confirmation message to carry on the actual resource allocation if feedbacks from InPs involved are *all* positive, otherwise terminate the processing of the current VN request.

III. Network Model

A. Substrate and Virtual Network

`Substrate network`. We model the substrate network operated by all InPs as an undirected graph and denote it by $G^S = (N^S, L^S)$, where N^S and L^S represent the set of substrate nodes and links, respectively.

For each node $n^S \in N^S$, besides its geographic location $g(n^S)$ and available CPU capacity $C(n^S)$, it is also associated with a domain index $dom(n^S)$ to denote which substrate domain it belongs to. Each substrate link $l^S(u,v) \in L^S$ of the substrate node pair (u,v) has an available bandwidth capacity $B(l^S)$. In addition, we introduce the inter-domain link set L_e^S

defined in (1). Accordingly, the intra-domain link set is the relative complement of L_e^S in L^S.

$$L_e^S = \{l^S(u,v) \mid dom(u) \neq dom(v), u, v \in N^S\} \quad (1)$$

Let $\mathcal{P}^S(m^S, n^S)$ be the set of all possible substrate paths from the source node m^S to the destination node n^S. The residual bandwidth of a substrate path $P^S \in \mathcal{P}^S(m^S, n^S)$, denoted by $R(P^S)$, is defined as the minimum available capacity of all substrate links along the path

$$R(P^S) = \min_{l^S \in P^S} B(l^S) \quad (2)$$

`VN request`. Similar to the substrate network, we model a VN request as an undirected graph denoted by $G^V = (N^V, L^V)$, where N^V and L^V represent the set of virtual nodes and links, respectively. Each virtual node $n^V \in N^V$ is associated with its capacity requirement $C(n^V)$, desired location $g(n^V)$ and a non-negative radius d^V as the distance constraint to indicate how far it can be mapped from the location $g(n^V)$. Generally, d^V can be measured by geographical distance or round-trip time (RTT). The capacity requirement of each virtual link $l^V \in L^V$ is denoted by $B(l^V)$.

B. Augmented Network

Upon receiving a VN request $G^V = (N^V, L^V)$, VNP first identifies a candidate set $\Theta(n^V)$ for each virtual node ($n^V \in N^V$), which consists of all possible substrate nodes satisfying the distance constraint of n^V.

$$\Theta(n^V) = \{n^S \in N^S \mid dis(g(n^V), g(n^S)) \leq d^V\} \quad (3)$$

Next, the augmented network $G^A = (N^A, L^A)$ is constructed by VNP according to Equation (4),

$$\begin{aligned} N^A &= N^S \cup N^V \\ L^A &= L_e^S \cup L_{vs} \cup L_{ss} \end{aligned} \quad (4)$$

where the node set N^A is a combination of all substrate and virtual nodes, and the link set L^A consists of three types of links, namely the inter-domain substrate links, vs-pseudo links and ss-pseudo links. The two kinds of pseudo links introduced in Section II-C are all with infinite bandwidths and formally defined as follows.

$$\begin{aligned} L_{vs} &= \{(n^V, n^S) \mid n^V \in N^V, n^S \in \Theta(n^V)\} \\ L_{ss} &= \{(m^S, n^S) \mid m^S, n^S \in N^S, dom(m^S) = dom(n^S)\} \end{aligned}$$

We also define an augmented path set $\mathcal{P}^A(m^A, n^A)$ as the set of all possible paths from the source node m^A to the destination node n^A in the augmented network. The residual bandwidth of a path $P^A \in \mathcal{P}^A(m^A, n^A)$, denoted by $R(P^A)$, is defined as the minimum available capacity of all augmented links along the path.

$$R(P^A) = \min_{l^A \in P^A} B(l^A) \quad (5)$$

C. VN Request Assignment

When a VN request G^V arrives, VNP should decide the assignment of G^V onto the augmented network, based on which the entire VN request is decomposed into multiple sub-VN requests. Then, each InP involved determines the assignment of a specific sub-VN request received from VNP. Therefore, the assignment of an entire VN request can be divided into two phases, namely the VNP assignment phase and the InP assignment phase.

1) *VNP assignment phase*. Since VNP only has the information of partial substrate resources, it conducts the mapping from the VN request onto the augmented network.

`VNP Node Mapping`. Although a substrate node can host multiple virtual nodes from different VN requests, it can only be allocated to at most one virtual node in a single VN request. Therefore, the node mapping $\mathcal{M}_N : N^V \mapsto N^A$ from virtual nodes to augmented nodes is, for all $n^V, m^V \in N^V$

$$\begin{aligned} &\mathcal{M}_N(n^V) \in \Theta(n^V) \\ &\mathcal{M}_N(m^V) = \mathcal{M}_N(n^V), \quad \text{iff } m^V = n^V \end{aligned}$$

subject to

$$C(n^V) \leq C(\mathcal{M}_N(n^V))$$

`VNP Link Mapping`. A virtual link can be mapped to either a single augmented path for an unsplittable flow or a set of augmented paths for a splittable flow between the two substrate nodes that host m^V and n^V, respectively. Although the splittable link mapping is easier, we consider the unsplittable case, because the current inter-domain routing protocol (*i.e.*, BGP) limits each router to using a single route for each destination substrate domain. However, we do allow an InP to use the splitting technique within its own domain so as to gain more flexibility, which is also a common assumption for the Intra-domain embedding [7, 8].

For each virtual link $l^V = (m^V, n^V) \in L^V$, the mapping $\mathcal{M}_L : L^V \mapsto P^A$ is to find an augmented path such that

$$\mathcal{M}_L(m^V, n^V) \subseteq P^A(\mathcal{M}_N(m^V), \mathcal{M}_N(n^V)),$$

subject to

$$B(l^V) \leq R(P^A)$$

2) *InP Assignment Phase*. After the VNP assignment, the VN request G^V is decomposed into sub-VN requests, the one of which for Domian i is denoted by $G_i^{\bar{V}} = (N_i^{\bar{V}}, L_i^{\bar{V}})$.

`InP Node Mapping`. Actually, the mapping of each node $n^{\bar{V}} \in N_i^{\bar{V}}$ onto the substrate node in Domain i has already been determined by VNP in its node mapping step. Therefore, the InP of Domain i only needs to select n^S as the host of $n^{\bar{V}}$ such that

$$\mathcal{M}_{\bar{N}}(n^{\bar{V}}) = n^S$$

subject to

$$dom(n^S) = i \quad \text{and} \quad g(n^S) = g(n^{\bar{V}})$$

`InP Link Mapping`. For each link $l^{\bar{V}} = (m^{\bar{V}}, n^{\bar{V}}) \in L_i^{\bar{V}}$, the link mapping $\mathcal{M}_{\bar{L}} : L_i^{\bar{V}} \mapsto \mathcal{P}^S$ is to find a set of substrate paths, such that

$$\mathcal{M}_{\bar{L}}(m^{\bar{V}}, n^{\bar{V}}) \subseteq \mathcal{P}^S(\mathcal{M}_{\bar{N}}(m^{\bar{V}}), \mathcal{M}_{\bar{N}}(n^{\bar{V}}))$$

subject to

$$B(l^{\bar{V}}) \leq \sum_{P^S \in \mathcal{M}_{\bar{L}}(l^{\bar{V}})} R(P^S)$$

A VN request will be accepted, if the VN assignment and the InP assignment of each sub-VN request are all successfully finished. Otherwise, the request will be rejected.

IV. Problem Formulation

In this section, we formulate two key problems in the inter-domain embedding, namely VNP assignment (*i.e.*, VN decomposition) and InP assignment (*i.e.*, sub-VN embedding).

A. Objectives of VNP and InPs

With the business model in Section II, the successful embedding of a VN request brings revenue and cost simultaneously to VNP and InPs. Assume an entire VN request $G^V = (N^V, L^V)$ has already been decomposed into sub-VN requests $G_i^{\bar{V}} = (N_i^{\bar{V}}, L_i^{\bar{V}}), \forall i$. We now explicitly define the objectives of each role from the bottom up.

The cost to the i-th InP in embedding $G_i^{\bar{V}}$ is denoted by $\mathbb{C}^-(G_i^{\bar{V}})$, which is defined as the total amount of substrate resources allocated to $G_i^{\bar{V}}$ multiplied by their costs $c(\cdot)$.

$$\mathbb{C}^-(G_i^{\bar{V}}) = \sum_{n^{\bar{V}} \mapsto n^S, n^{\bar{V}} \in N_i^{\bar{V}}} c(n^S) C(n^{\bar{V}}) + \sum_{l^{\bar{V}} \in L_i^{\bar{V}}} \sum_{l^S \in L^S} c(l^S) \phi_{l^S}^{l^{\bar{V}}} B(l^{\bar{V}})$$

where $\phi_{l^S}^{l^{\bar{V}}}$ represents the fraction of $l^{\bar{V}}$'s bandwidth requirement satisfied by the substrate link l^S.

The revenue of the i-th InP denoted by $\mathbb{R}^-(G_i^{\bar{V}})$ is defined as the resources that $G_i^{\bar{V}}$ requires multiplied by their prices.

$$\mathbb{R}^-(G_i^{\bar{V}}) = \sum_{n^{\bar{V}} \in N_i^{\bar{V}}} p(n^{\bar{V}}) C(n^{\bar{V}}) + \sum_{l^{\bar{V}} \in L_i^{\bar{V}}} p(l^{\bar{V}}) B(l^{\bar{V}})$$

Since VNP is charged by InPs for corresponding sub-VN requests, the cost to VNP in decomposing G^V is the total revenues of InPs in embedding all sub-VN requests of G^V, which is defined as $\mathbb{C}^+(G^V) = \sum_i \mathbb{R}^-(G_i^{\bar{V}})$.

The revenue of VNP to successfully accept a VN request G^V, which is denoted by $\mathbb{R}^+(G^V)$, is defined as the virtual resources required in G^V multiplied by their prices.

$$\mathbb{R}^+(G^V) = \sum_{n^V \in N^V} p(n^V) C(n^V) + \sum_{l^V \in L^V} p(l^V) B(l^V)$$

B. VNP Formulation

As illustrated in Section II-C, a major challenge in the VNP assignment is to minimize the total decomposition cost by coordinating the node and link mapping together. If we treat each virtual link as a commodity flow in the augmented network G^A, the VNP assignment can be formulated as a conventional multi-commodity flow problem, with additional binary constraints to ensure 1) a unique placement for each virtual node and 2) the unsplittable flow for each virtual link.

Let f_{uv}^k be the fraction of the k-th commodity flow routed via the augmented link (u, v), and x_{uv} be a binary variable indicating whether a node $u \in N^A \setminus N^S$ is mapped onto an augmented node v in its candidate set. The minimum-cost VNP assignment is formulated as follows.

VNP_MCF

$$\textbf{minimize} \sum_{u \in N^A \setminus N^S} C(u) \sum_{v \in \theta(u)} x_{uv} p(v) + \sum_{(u,v) \in L^A} p(u,v) \sum_k f_{uv}^k B(l_k^V) \tag{6a}$$

subject to :

$$\sum_k (f_{uv}^k + f_{vu}^k) B(l_k^V) \leq \begin{cases} C(u,v) x_{uv}, & \forall u \in N^A \setminus N^S \\ C(u,v), & \text{otherwise} \end{cases} \tag{6b}$$

$$x_{uv} C(u) \leq C(v), \qquad \forall u \in N^A \setminus N^S, v \in \theta(u) \tag{6c}$$

$$\sum_{u:(i,u) \in L^A} f_{iu}^k - \sum_{u:(u,j) \in L^A} f_{uj}^k = d_u^k, \quad \forall k, \forall u \in N^A \tag{6d}$$

$$\sum_{u:v \in \theta(u)} x_{uv} \leq 1, \qquad \forall v \in N^S \tag{6e}$$

$$\sum_{v \in \theta(u)} x_{uv} = 1, \qquad \forall u \in N^A \setminus N^S \tag{6f}$$

$$x_{uv} \in \{0,1\}, \qquad \forall u \in N^A \setminus N^S, v \in \theta(u) \tag{6g}$$

$$f_{uv}^k \in \{0,1\}, \qquad \forall u, v \in N^A \tag{6h}$$

- Eqs. (6b) and (6c) are capacity constraints on the augmented links and nodes, respectively. In particular, when $u \in N^A \setminus N^S$, the load on both directions of the undirected link (u, v) (*i.e.*, summing up f_{uv}^k and f_{vu}^k for all k) is also under the control of the variable x_{uv}.
- Eq. (6d) is the flow conservation constraint, where the value of d_u^k is -1 for the source of the k-th flow, 1 for the destination of the k-th flow, or 0 otherwise.
- Eq. (6e)-(6g) ensure each virtual node is mapped onto a unique node in its candidate set. Eq. (6h) corresponds to the unspittable flow assumption in Section III-C.

C. InP Formulation

Upon receiving a sub-VN request $G_i^{\bar{V}} = (N_i^{\bar{V}}, L_i^{\bar{V}})$, the InP tries to get $G_i^{\bar{V}}$ by minimizing the link embedding cost, because the node mapping has already been determined. Similar to many existing techniques, we formulate the process conducted by an InP as a multi-commodity flow problem, which can be solved in polynomial time. Each link $l^{\bar{V}} \in L_i^{\bar{V}}$ is considered as a flow. We define ϕ_{mn}^k $(k = 1, \ldots, |L_i^{\bar{V}}|)$ as the fraction of the k-th flow routed via the substrate link $l^S(m,n) \in L^S$. We present the InP formulation as follows.

InP_MCF

$$\textbf{minimize} \sum_{(m,n) \in L^S} p(m,n) \sum_k \phi_{mn}^k B(l_k^{\bar{V}}) \tag{7a}$$

subject to :

$$\sum_{u:(i,u)\in L^S} \phi_{iu}^k - \sum_{u:(u,j)\in L^S} \phi_{uj}^k = d_u^k, \quad \forall k, \forall u \in N^S \quad (7b)$$

$$\sum_k (\phi_{mn}^k + \phi_{nm}^k) B(l_k^{\bar{V}}) \le R(m,n), \quad \forall m,n \in N^S \quad (7c)$$

$$0 \le \phi_{mn}^k \le 1, \quad \forall m,n \in N^S \quad (7d)$$

– Eq. (7b) describes the flow conservation constraint, where the value of d_u^k is -1 for the source of the k-th flow, 1 for the destination of the k-th flow, or 0 otherwise.

– Eq. (7c) describes the capacity constraints. Here the embedding objective function can be extended to more complicated ones (*e.g.*, considering load balancing simultaneously).

V. Inter-Domain VN Embedding Algorithm

Based on the formulation of both VNP and InPs, we present the VN embedding algorithm in this section.

A. Algorithm Description

Traditional techniques for solving **VNP_MCF** (*e.g.*, dynamic programming) are computationally intractable for large-scale substrate networks [7, 12], because the search space is huge. Hence, we relax the integer constraints to obtain a linear programming **VNP_Relax**, and then employ additional heuristics to get binary values for f_{uv}^k and x_{uv}.

VNP_Relax has the same objective function and constraints as **VNP_MCF**, except for the integer constraints in (6g) and (6h) being replaced with the following ones:

$$\begin{aligned} &0 \le x_{uv} \le 1, \quad u \in N^A \setminus N^S, v \in \theta(u) \\ &0 \le f_{uv}^k \le 1, \quad \forall u,v \in N^A \end{aligned} \quad (8)$$

The procedure for the VNP assignment is presented in Algorithm 1. It begins by creating an augmented network G^A and initiating the embedding status of the current VN request as `FALSE`. Then, **VNP_Relax** is solved to get a fractional, instead of an integer, solution to the minimization of the total embedding cost. The binary $\mu(n^S)$ for each $n^S \in N^S$ is introduced to ensure that each substrate node will be used at most once for the same VN request.

Next, the node mapping phase begins (Lines 4-13). The binary constraints in **VNP_MCF** implicitly indicate the inter-dependency between f and x. That is, if a virtual node u is mapped onto node v (*i.e.*, $x_{uv} = 1$), all flows from or to u must be routed via the vs-pseudo link (u,v). However, this correlation is lost in the derived solution of **VNP_Relax**, because f and x can be any values between 0 and 1 as long as the objective function is minimized. Therefore, for each virtual node u, we introduce $\omega(v)$ as the likelihood of mapping from u to its candidate node v, which is defined as the product of x_{uv} and the total traffic fraction on (u,v) in both directions. The procedure maps u onto an available candidate node v (*i.e.*, $\mu(v) = 0$ and $C(v) \ge C(u)$) with the maximum ω value. If no substrate node in u's candidate set is available, the embedding process will terminate immediately (Lines 9-11).

Following that is the link mapping phase (Lines 14-20). Finding an optimal mapping from a virtual link to a single augmented path with fixed node mapping reduces to the unsplittable flow problem, which is NP-hard [15]. Therefore, for each virtual link (u,v), the procedure searches for a single minimum-cost path with enough residual capacity between the substrate nodes $\mathcal{M}_N(u)$ and $\mathcal{M}_N(v)$ that are determined in the node mapping phase. In practice, Dijkstra's algorithm can efficiently find such a path, by using the prices of the augmented links as their weights and skipping the links without enough available capacities. If it fails to find an available path, the embedding process will stop. Virtual nodes (or links) are processed in the descending order of their resource requirements, because it is easier to satisfy a virtual node (or link) with less requirements.

Algorithm 1: VNP_Embedding

Input: $G^V = (N^V, L^V)$, N^S, L_e^S
Output: $Status$

1 Create the augmented network $G^A = (N^A, L^A)$ in (4)
2 Set $Status \leftarrow$ `FALSE` and solve **VNP_Relax** in (8)
3 Initialize $\mu(n^S) \leftarrow 0$ for all $n^S \in N^S$
4 **for** $u \in N^V$ in *descending* order of $C(u)$ **do**
5 **for** $v \in \theta(u)$ **do**
6 $\omega(v) \leftarrow x_{uv} \sum_k (f_{uv}^k + f_{vu}^k)$
7 **end**
8 $v \leftarrow \arg\max_v \{\omega(v) \mid \mu(v) = 0, C(v) \ge C(u)\}$
9 **if** $v =$ `NULL` **then**
10 **return**
11 **end**
12 $\mu(v) \leftarrow 1$; $\mathcal{M}_N(u) \leftarrow v$
13 **end**
14 **for** $l(u,v) \in L^V$ in *descending* order of $B(l)$ **do**
15 $P \leftarrow \arg\min_{P \in \mathcal{P}(\mathcal{M}_N(u), \mathcal{M}_N(v))} \{Cost(P) \mid R(P) \ge B(l)\}$
16 **if** $P =$ `NULL` **then**
17 **return**
18 **end**
19 $\mathcal{M}_L(l) \leftarrow P$
20 **end**
21 Decompose G^V into sub-VN requests $G^{\bar{V}} = (N^{\bar{V}}, V^{\bar{V}})$
22 **for** each $G_i^{\bar{V}} \in G^{\bar{V}}$ **do**
23 $S(G_i^{\bar{V}}) \leftarrow$ **InP_Embedding**$(G_i^{\bar{V}})$ by the i-th **InP**
24 **end**
25 **if** all $S(G_i^{\bar{V}}) =$ `TRUE` **then**
26 $Status \leftarrow$ `TRUE`
27 **end**

Based on a successful node and link mapping, the entire VN request is decomposed into sub-VN requests, each of which is assigned to a particular InP for further processing (Lines 22-24). The VN request will be accepted by setting its status to `TRUE`, if all its sub-VN requests can be successfully embedded by InPs, otherwise it will be rejected.

The procedure in Algorithm 2, which is conducted by an individual InP, tries to map a sub-VN request onto its substrate network by solving **InP_MCF** and then returns VNP with the status. Note that if InP receives a positive final confirmation from VNP, the recorded assignment (Line 3) can be directly used to allocate substrate resources to the current sub-VN request. Here we simply assume that each InP has the same goal in processing a sub-VN request, *i.e.*, minimizing its embedding cost, but it can be easily extended by associating each InP with its specific embedding goal.

The inter-domain VN embedding algorithm consists of **VNP_Embedding** in Algorithm 1 and **InP_Embedding** in

Algorithm 2: InP_Embedding

Input: $G_i^{\bar{V}} = (N_i^{\bar{V}}, L_i^{\bar{V}})$
Output: $S(G_i^{\bar{V}})$

1 Set $S(G_i^{\bar{V}}) \leftarrow$ `FALSE` and solve **InP_MCF** in (7)
2 **if InP_MCF** succeeded **then**
3 $S(G_i^{\bar{V}}) \leftarrow$ `TRUE` and record the assignment of $G_i^{\bar{V}}$
4 **end**

Algorithm 2. Now we analyze its time complexity. The dominating components in the running time of Algorithm 1 are the time to solve **VNP_Relax**, to execute the link mapping phase and to invoke Algorithm 2 for the sub-VN embedding, respectively. The multi-commodity flow problem **VNP_Relax** can be solved in time $O((|L^A|(|L^V|+1))^{3.5}Y^2)$, where Y is the number of bits per variable in the input [14]. The for loop in Line 14 executes Dijkstra's algorithm for $|L^V|$ times and thus results in a total time complexity of $O(|L^V|(|L^A| + |N^A|\log|N^A|))$. Since Algorithm 2 can be run by each InP in parallel, the total time of processing all sub-VN requests is actually approximate to the time of solving the multi-commodity flow problem **InP_MCF** once, which is $O((|L^A||L^V|)^{3.5}Y^2)$. Therefore, the total time complexity of Algorithm 1 is $O((|L^A|(|L^V|+1))^{3.5}Y^2)$.

B. Discussions

Different mapping choices. We assume virtual links are unsplittable in the link mapping phase in Algorithm 1. The reason lies in that the current inter-domain routing protocol (*i.e.*, BGP) limits each router to using a single route for each destination substrate domain (or Autonomous Systems), though applying the multi-path inter-domain routing scheme (*e.g.*, MIRO [18]) would make the problem easier.

Besides the traffic-fraction (TF) selection based on ω values, the node mapping phase of Algorithm 1 can use other heuristic alternatives, such as best-fit (BF) or first-fit (FF). To host a virtual node, BF selects the candidate substrate node with the lowest price, whereas FF chooses the first available one. We will evaluate these choices in Section VI.

Multiple embedding attempts. To improve the acceptance probability of a VN request, an SP can assign it with a deadline extension ratio, which indicates the maximum period compared with its lifetime it can wait for acceptance. Then, a failed VN request will be stored by VNP for more embedding attempts rather than being rejected immediately. Whenever substrate resources are released by a finished VN request, VNP schedules the unexpired waiting VN requests for the next embedding attempt. We also explore the impact of the deadline extension ratio in the next section.

VI. Performance Evaluation

In this section, we focus on evaluating the performance of the proposed algorithm and comparing it with other algorithms in terms of the VN request acceptance ratio and so forth.

A. Evaluation Settings

Topology. The substrate network in our experiments is randomly generated in a 50x50 grid using GT-ITM [21]. It consists of 4 domains connected via 28 inter-domain links. Each domain has on average 30 nodes and 72 intra-domain links, which emulates a medium-scale network domain (*e.g.*, GÉANT topology with 23 nodes and 74 links [22]). The capacities for nodes, intra-domain links and inter-domain links are random integers that are uniformly distributed over the interval [100,150], [100,150] and [300,400], respectively.

TABLE I. Algorithms in Comparison

Abbr.	Description
BM	Traffic-fraction node mapping with *full* information
TF	Traffic-fraction node mapping with partial information
BF	Best-fit node mapping with partial information
FF	First-fit node mapping with partial information

The VN requests are also generated using GT-ITM in the same grid. The number of virtual nodes in each request is randomly determined by a uniform distribution between 5 and 10. The probability of having a link between each pair of virtual nodes is 0.4. The capacity requirements of virtual nodes and links are both uniformly distributed between 0 and 20.

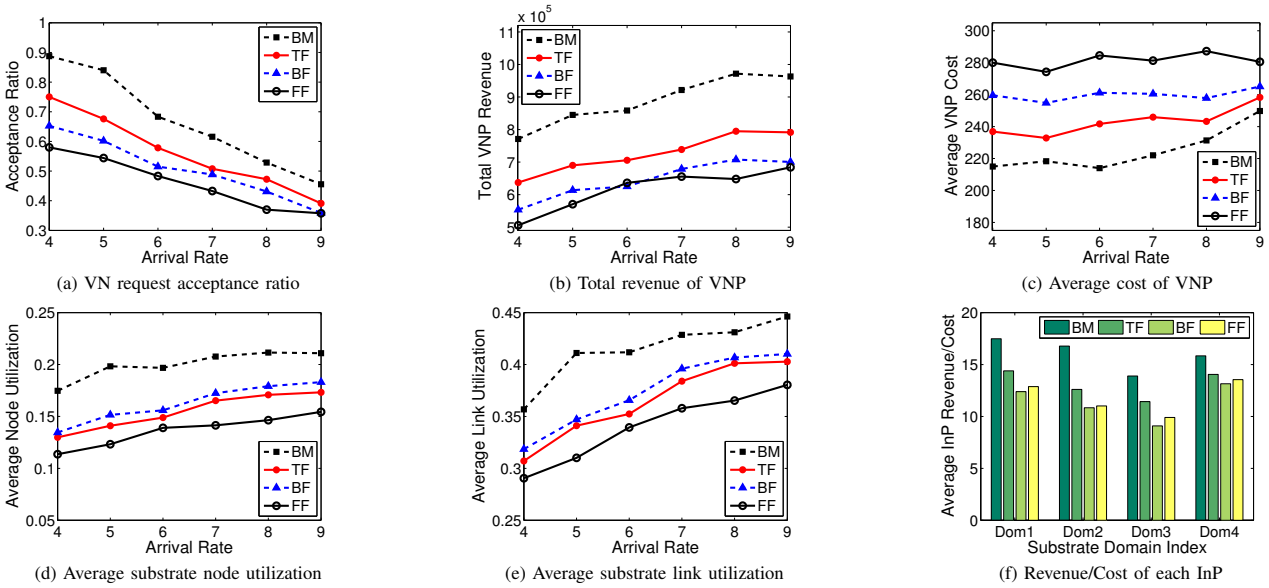
Parameters. We assume that the arrival of VN requests follows a Poisson model and vary the average request arrival rate λ from 4 to 9 per 100 time units. The lifetime of each VN follows an exponential distribution with an average of 1000 time units. Each simulation lasts for 30,000 time units.

Compared Algorithms. Due to the lack of relevant algorithms under the same information sharing scheme, we resort to an *ideal* case as a benchmark (BM in Table I) where all substrate information is available to VNP. TF is the algorithm proposed in this paper. In BF and FF, VNP obtains the same substrate information with TF, whereas adopts different node mapping strategies introduced in Section V.

Comparison Metrics. We use the following metrics to quantify the results of each algorithm,

- *VN request acceptance ratio*. It is the fraction of accepted VN requests in all VN requests. A higher acceptance ratio intuitively brings higher revenue to VNP and InPs.
- *Total revenue and average cost of VNP*. Ideally, a profitable algorithm enables VNP to receive a higher total revenue and a lower average cost.
- *Average node and link utilization*, which is measured by averaging the utilization of all nodes (links) over time.
- *Revenue and cost of InPs*. Similar to that of VNP, a profitable algorithm is also desired by InPs.

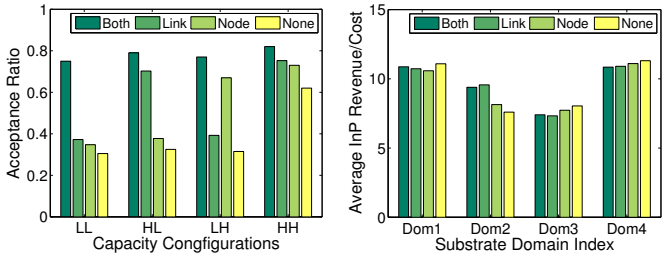
For simplicity, the costs of substrate resources (*i.e.*, $c(n^S)$ and $c(l^S)$ defined in Section IV-A) are set to real numbers between $[0, 1]$ in proportion to their capacities. The prices offered by an InP (*i.e.*, $p(n^{\bar{V}})$ and $p(l^{\bar{V}})$) are 20 times the average costs of corresponding resources in its domain. The prices of virtual resources provided by VNP to SPs (*i.e.*, $p(n^V)$ and $p(l^V)$) are all set to 50.



(a) VN request acceptance ratio (b) Total revenue of VNP (c) Average cost of VNP

(d) Average substrate node utilization (e) Average substrate link utilization (f) Revenue/Cost of each InP

Fig. 3. Comparison of the embedding algorithms against the VN request arrival rate



(a) VN request acceptance ratio (b) Revenue/Cost of InPs for LL

Fig. 2. Comparison of different information-sharing strategies.

B. *Effect of Disclosed Substrate Information*

In the information sharing scheme defined in Section II, we advocate the disclosure of available capacities for *both* substrate nodes and inter-domain links. Besides the current strategy (Both), InPs can choose from another three ones that provide VNP with available capacities for node-only (Node), inter-domain link-only (Link) or none of them (None). Now we explore how these strategies affect the embedding results.

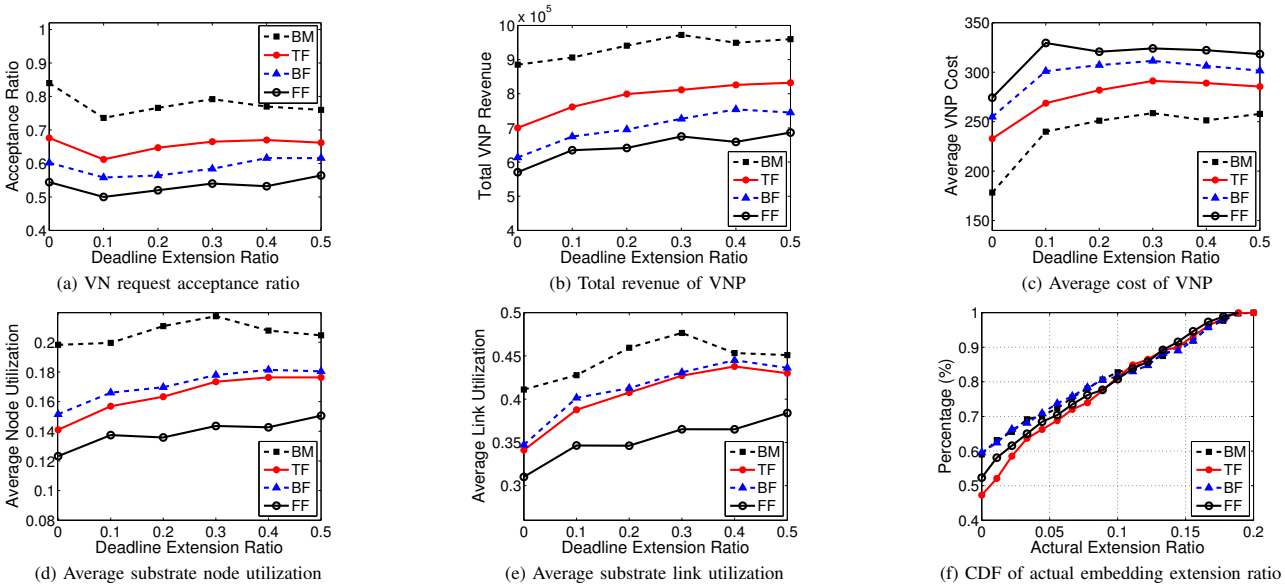
Here we only employ the TF algorithm, which is slightly modified to work with different strategies, *i.e.*, disabling the capacity constraints in the node mapping (Line 8) and the link mapping (Line 15) in Algorithm 1 when the available capacities of corresponding resources are unrevealed. We refer to the current capacity settings as LL (low-low), and enumerate three more configurations by *doubling* the capacities of node-only (HL), inter-domain link-only (LH) or both (HH).

The results with $\lambda = 4$ are shown in Fig. 2, which are summarized as follows. First, with the current capacity settings (*i.e.*, LL), the disclosure of both resources significantly outperforms the other strategies in terms of acceptance ratio. In Fig. 2 (b), we leverage a metric to capture the profit of each InP with this strategy, which is defined as the average ratio of the revenue it gains in embedding a VN request to its cost. Since the average profit of accepting a request is similar for different strategies, InPs can benefit from a high acceptance ratio. Second, disclosing the capacities of relatively scare resources greatly contributes to accommodating more requests, *i.e.*, Link in HL and Node in LH, because decomposition decisions based on the available capacities of bottleneck resources are more likely to be feasible. Finally, increasing capacities of nodes and inter-domain links simultaneously can improve the performance of None. We may intuitively infer that if the substrate resources are sufficient enough to accommodate all VN requests regardless how they are decomposed by VNP, the four strategies would have quite similar, or even the same, results. The real fact is, however, multiple VN requests compete for limited substrate resources. Therefore, disclosing available capacities of both resources is beneficial to VNP and InPs. In the following simulations, we focus on evaluating the algorithms in Table I using the Both strategy.

C. *Effect of Request Arrival Rate*

An important parameter in the VN embedding is the average VN request arrival rate λ. Fig. 3 shows the comparison on the four metrics by varying λ. At a higher arrival rate, more VN requests compete for the limited substrate resources. The acceptance ratio is thus smaller as confirmed in Fig. 3(a). Undoubtedly, BM always accepts the most requests, because it gets insight into available capacities of all substrate resources. In the other three algorithms, TF leverages the node mapping strategy from which the following link mapping will benefit and thus has the highest acceptance ratio.

The total revenue of VNP for each algorithm significantly increases as λ grows, shown in Fig. 3(b), because more requests are accepted despite of the gradually decreasing acceptance ratio. However, the average embedding cost of VNP, shown in Fig. 3(c), merely exhibits a slight growth when λ enlarges. From Fig. 3(b) and (c), we observe that TF as well as BM generate more total revenue and less per-request cost than BF and FF, respectively. Upon receiving a VN request, TF and BM coordinate the node and link mapping to minimize the total cost and thus outperform the other two heuristics.



(a) VN request acceptance ratio (b) Total revenue of VNP (c) Average cost of VNP

(d) Average substrate node utilization (e) Average substrate link utilization (f) CDF of actual embedding extension ratio

Fig. 4. Comparison of the embedding algorithms against the VN requests' deadline extension ratio, when the arrival rate is 5.

TABLE II. AVERAGE RESOURCE UTILIZATION IN EACH SINGLE DOMAIN

Al.	Node Utilization				Link Utilization			
	D1	D2	D3	D4	D1	D2	D3	D4
BM	0.156	0.147	0.247	0.254	0.321	0.343	0.525	0.467
TF	0.114	0.104	0.176	0.177	0.245	0.263	0.474	0.353
BF	0.143	0.106	0.180	0.193	0.292	0.279	0.482	0.356
FF	0.127	0.068	0.147	0.109	0.275	0.249	0.439	0.269

Fig. 3(d) and (e) depict the average utilization of nodes and links in the substrate network (over all domains) for different algorithms. Intuitively, an algorithm with a higher acceptance ratio will result in a higher resource utilization. Comparing TF with BF, however, we notice an encouraging feature of TF that it achieves higher acceptance ratios and lower resource utilizations simultaneously. That is because BF uses a greedy strategy to map each virtual node onto a cheapest candidate node, which may cause unnecessary long substrate paths to support virtual links. Due to space constraints, we only present resource utilization of each domain at $\lambda = 5$ in Table II. The results reveal the same feature of TF as mentioned above. The average profit of each InP at $\lambda = 5$ in Fig. 3(f), shows that TF is more profitable than BF and FF.

In summary, TF achieves roughly 80%-90% of the benchmarks for all metrics, which indicates the effectiveness of the proposed information sharing scheme. Meanwhile, the superiority of TF over BF and FF shows the advantage of coordinating the node and the link mapping phases.

D. Effect of Deadline Extension Ratio

This section evaluates the performance of the algorithms by varying a deadline extension ratio ρ (see Section V) from 0 to 0.5. A *deadline*-based scheduling is used to provide a request that is closer to its acceptance deadline with a higher priority for the next embedding attempt. An alternative way is to schedule a request with higher revenue first. Both scheduling produces similar overall embedding results. We fix λ to 5 and present the results in Fig. 4. Several key observations are summarized as follows.

1) Comparing with Fig. 3, the ranking of the algorithms in Fig. 4(a)-(e) remains the same. Due to space constraints, the revenue/cost ratios of InPs are not shown here.

2) Employing the deadline extension ratio does not necessarily increase the acceptance ratio. However, it does contribute to improving the revenue of VNP. We observe from Fig. 4(a) and (b) that when ρ varies from 0 to 0.1, the acceptance ratio drops whereas the total revenue of VNP increases. This is because more VN requests with larger resource demands can be satisfied by waiting a short period (*i.e.*, 1/10 of their lifetime), which results in more failures of the following VN requests with smaller demands. In practice, a trade-off must be made between accommodating more VN requests and bringing more revenue to VNP as well as InPs. For Fig. 4, an appropriate ρ would be between 0.2 and 0.3, which is also reasonable for VN requests to wait for such a period of time.

3) The actual waiting periods of the accepted requests are on average much less than their deadline extension ratios. For all the accepted VN requests when $\rho = 0.2$, we present the CDF of the ratios of their actual waiting time to their corresponding lifetime in Fig. 4(f). The four algorithms have similar results, and nearly a half of the VN requests are accepted without any delay. The average actual waiting ratios of BM, TF, BF and FF are 0.0413, 0.0477, 0.0416, 0.0447, respectively.

VII. RELATED WORK

Network virtualization is considered to be a promising way to overcome the Internet impasse [2] and has attracted many research attention. Here we only summarize existing studies relevant to VN embedding and bandwidth provisioning.

Intra-domain embedding. Following the business role in [1] that decouple InPs from SPs, many algorithms are proposed for the VN embedding problem in a single substrate network

domain. They can be roughly classified into two categories: the offline [4–6] and the online [7–10] versions. The former assumes that all VN requests are known in advance, whereas the latter deals with dynamically arrived VN requests. Yu *et al.*[7] investigate a substrate network that supports path splitting and migration. Chowdhury *et al.*[8] propose the augmented network to introduce better correlation between the node mapping and link mapping phases.

Inter-domain embedding. There are few pioneer studies that extend the Intra-domain embedding to multiple domains [11–13]. They all consider dividing an end-to-end virtual network into multiple sub-VN requests, each of which can be separately embedded by intra-domain algorithms.

Chowdhury *et al.*[11] propose a distributed multi-step mechanism named PolyViNE. The embedding relies on the relay of multiple InPs, which goes against the intention of decoupling business roles. In [12], the embedding process does not consider any detailed information (*e.g.*, node location and capacity) and thus may suffer from a high failure rate. The latest study [13] presents a framework with limited substrate information. In order to facilitate the embedding process, it uses the traffic matrix, instead of the topology, to specify a VN request. However, an embedding result using traffic matrix does not necessarily satisfy the requirements of a topology-based VN request, because multiple different topologies correspond to the same traffic matrix. In addition, it does not design practical embedding algorithms.

In this paper, we first experimentally explore how to define an *appropriate* information sharing scheme based on the substrate resources, and then propose practical algorithms to embed online VN requests specified by topologies.

Bandwidth provisioning. With the emerging of QoS-sensitive services (*e.g.*, VoIP), many studies [16, 17] focus on the bandwidth provisioning problem, which is critical to guarantee QoS of end-to-end services. Duan *et al.*[16] propose and advocate the service overlay network (SON) as a promising way to address QoS issues. Fan and Ammar [17] study the dynamic reconfiguration policies in SON, with the goal of minimizing the potential overall cost of using an overlay.

The bandwidth provisioning can reduce to the link embedding without node constraints, *i.e.*, finding a path for each given ingress-egress pair. In this paper, we consider the embedding of both virtual nodes and links.

VIII. Conclusion

In this paper, we develop an efficient solution to the inter-domain VN embedding problem. To meet requirements of VNP and InPs, a new information sharing scheme is proposed to enable the VN decomposition with limited substrate information. Then, we formulate the inter-domain embedding problem as an integer programming problem and devise a heuristic algorithm to process online VN requests in polynomial time. Extensive simulation results show that our solution outperforms other counterparts and achieves 80%-90% of the benchmarks in the *ideal* scenario. The work can be further improved by employing more techniques to increase the acceptance ratio, such as considering load balancing in VN decomposition.

Acknowledgements

This work has been supported in part by NSFC Project (61170292, 61161140454), National Science and Technology Major Project (2012ZX03005001), 973 Project of China (2012CB315803), 863 Project of China (2013AA013302) and EU Marie Curie Actions EVANS (PIRSES-GA-2010-269323).

References

[1] N. Feamster, L. Gao, and J. Rexford, "How to Lease the Internet in Your Spare Time," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 61-64, Jan. 2007.

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

[3] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy, Network "Virtualization Architecture: Proposal and Initial Prototype," in *Proc. ACM SIGCOMM VISA*, August 2009.

[4] Y. Zhu and M. Ammar, "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," in *Proc. IEEE INFOCOM*, pp. 1-12, April 2006.

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

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

[7] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17-29, Apr. 2008.

[8] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE INFOCOM*, pp. 783-791, Apr. 2009.

[9] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. ACM SIGCOMM VISA*, pp. 81-88, 2009.

[10] N. F. Butt, M. Chowdhury, and R. Boutaba, "Topology-awareness and reoptimization mechanism for virtual network embedding," in *Proc. IFIP Netw.*, pp. 27-39, 2010.

[11] M. Chowdhury, F. Samuel, and R. Boutaba, "PolyViNE: policy-based virtual network embedding across multiple domains," in *Proc. ACM SIGCOMM VISA*, pp. 49-56, 2010.

[12] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Comput. Netw.*, vol. 55, no. 4, pp. 1011-1023, March 2011.

[13] D. Dietrich, A. Rizk and P. Papadimitriou, "Multi-Domain Virtual Network Embedding with Limited Information Disclosure," in *Proc. IFIP Netw.*, 2013.

[14] N. Karmarkar, "A new polynomial-time algorithm for linear programming, in *Proc. ACM STOC*, pp. 302C311, 1984.

[15] S. G. Kolliopoulos and C. Stein, "Improved approximation algorithms for unsplittable flow problems," in *Proc. IEEE Symposium on Foundations of Computer Science*, 1997.

[16] Z. Duan, Z. Zhang, and Y. T. Hou, "Service overlay networks: SLAs, QoS and bandwidth provisioning," *IEEE/ACM Trans. Netw.*, vol. 11, no. 6, pp. 870-883, Dec. 2003.

[17] J. Fan and M. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies, in *Proc. IEEE INFOCOM*, pp. 1-12, April 2006.

[18] W. Xu and J. Rexford, "MIRO: multi-path interdomain routing," in *Proc. ACM SIGCOMM*, pp. 171-182, 2006.

[19] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual Routers on the Move: Live Router Migration as a Network Management Primitive," in *Proc. ACM SIGCOMM*, 2008.

[20] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing Software Defined Networks," in *Proc. USENIX NSDI*, 2013.

[21] "GT-ITM". [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/.

[22] "GÉANT". [Online]. Available: http://www.geant.net/Pages/default.aspx