Virtual Network Embedding with Coordinated Node and Link Mapping

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

Muntasir Raihan Rahman
Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
Email: mr2rahman@uwaterloo.ca

Raouf Boutaba
Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
Email: rboutaba@uwaterloo.ca

***Abstract*—Recently network virtualization has been proposed as a promising way to overcome the current ossification of the Internet by allowing multiple heterogeneous virtual networks (VNs) to coexist on a shared infrastructure. A major challenge in this respect is the VN embedding problem that deals with efficient mapping of virtual nodes and virtual links onto the substrate network resources. Since this problem is known to be $\mathcal{NP}$-hard, previous research focused on designing heuristic-based algorithms which had clear separation between the node mapping and the link mapping phases.**

This paper proposes VN embedding algorithms with better coordination between the two phases. We formulate the VN embedding problem as a mixed integer program through substrate network augmentation. We then relax the integer constraints to obtain a linear program, and devise two VN embedding algorithms D-ViNE and R-ViNE using deterministic and randomized rounding techniques, respectively. Simulation experiments show that the proposed algorithms increase the acceptance ratio and the revenue while decreasing the cost incurred by the substrate network in the long run.

I. Introduction

Since its inception, the Internet has proven its worth by supporting myriads of distributed applications and heterogeneous networking technologies. However, due to the existence of multiple stakeholders with conflicting goals and policies, alterations, even necessary ones, to the present architecture have now become almost impossible to achieve. To fend off the rigidity of the existing Internet once and for all, network virtualization has been propounded as a fundamental diversifying attribute of the future inter-networking paradigm that will allow multiple heterogeneous network architectures to coexist on a shared substrate [1], [2]. In a network virtualization environment (NVE), multiple service providers (SPs) will be able to create heterogeneous virtual networks (VNs) to offer customized end-to-end services to the end users by leasing shared resources from one or more infrastructure providers (InPs) without significant investment in physical infrastructure [2]–[4].

Each VN in the NVE is a collection of virtual nodes (e.g., virtual routers) that are hosted on different substrate nodes and interconnected by physical paths in the substrate network corresponding to virtual links. Since multiple VNs can share the same underlying physical resources, efficient and effective embedding/mapping/assignment [1] of each of the *online* VN requests is of utmost importance in order to increase *utilization* of the substrate network resources and consequently revenues of the InPs.

However, the VN embedding problem, with constraints on virtual nodes and virtual links, can be reduced to the $\mathcal{NP}$-hard *multi-way separator problem* [5], even if all the VN requests are known in advance. Even when all the virtual nodes are already mapped, embedding the virtual links with bandwidth constraints onto substrate paths is still $\mathcal{NP}$-hard in the *unsplittable flow* scenario. As a result, a number of heuristic-based algorithms have appeared in the relevant literature [6]–[9]. Most of these proposals focused primarily on edge mapping (using, for example, shortest path, k-shortest paths, and multi-commodity flow algorithms) after employing greedy methods to *preselect* the node mappings. However, preselecting node mappings without considering its relation to the link mapping stage restricts the solution space, and may result in poor performance.

In this paper, we introduce better correlation between the node mapping and the link mapping phases by proposing two new VN embedding algorithms D-ViNE (Deterministic VN Embedding) and R-ViNE (Randomized VN Embedding). In these algorithms, we map the virtual nodes to substrate nodes in a way that facilitates the mapping of virtual links to physical paths in the subsequent phase. To this end, we extend the physical network graph by introducing meta-nodes for each virtual node and connect the meta-nodes to a selected subset of physical nodes (Section IV-A). We then treat each virtual link with bandwidth constraints as a commodity consisting of a pair of meta-nodes. As a result, finding an optimal flow for the commodity is equivalent to mapping the virtual link in an optimal way. If we introduce additional binary constraints that force only one meta-edge to be selected for each meta-node, we can effectively select exactly one substrate node for each meta-node corresponding to a particular virtual node. We use mixed integer programming (MIP) formulation [10] to solve the embedding problem with binary constraints on the meta-edges and linear constraints on actual substrate network

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

978-1-4244-3513-5/09/$25.00 ©2009 IEEE

edges. Since solving an MIP is known to be $\mathcal{NP}$-hard [10], finding an optimal VN embedding using MIP becomes $\mathcal{NP}$-hard as well. As a result, we relax the integer program to obtain a linear programming formulation which can be solved in polynomial time. We then use deterministic and randomized rounding techniques on the solution of the linear program to approximate the values of the binary variables in the original MIP. Once all the virtual nodes have been mapped, we use the multi-commodity flow algorithm to map the virtual links onto the substrate network between the mapped virtual nodes [9], [11]. This can also be solved in polynomial-time since we assume that path splitting is supported by the substrate network [9].

The rest of this paper is organized as follows. Section II provides an overview of the related work. Following that, Section III formalizes the network model and the VN embedding problem itself. In Section IV, we provide the optimal MIP formulation for the VN embedding problem using substrate network augmentation. Section V relaxes the MIP formulation to obtain a linear program and presents D-ViNE and R-ViNE using deterministic and randomized rounding techniques. Section VI presents simulation results that evaluate the proposed algorithms, and we conclude in Section VII identifying future research directions.

II. Related Work

The VN assignment problem is similar to the previous works on embedding Virtual Private Networks (VPN) in a shared provider topology and the network testbed mapping problem [12], [13]. However, a typical VPN request consists only of bandwidth requirements, specified in terms of a traffic matrix, without any constraint on its nodes. As a result, most VPN design algorithms come down to finding paths for source/destination pairs. On the other hand, the *Assign* algorithm [13] used in the Emulab testbed considers bandwidth constraints alongside constraints on exclusive use of nodes, i.e., different VNs cannot share a substrate node. But in network virtualization, there are capacity and placement requirements on both the virtual nodes and the virtual links; in addition, substrate nodes and links can be shared by multiple VNs.

In order to reduce the hardness of the VN assignment problem and to enable efficient heuristics, existing research has been restricting the problem space in different dimensions, which include:

1) considering offline version of the problem (i.e., all the VN requests are known in advance) [7], [8],
2) ignoring either node requirements or link requirements [6], [7],
3) assuming infinite capacity of the substrate nodes and links to obviate admission control [6]–[8], and
4) focusing on specific VN topologies [7].

The authors in [9] consider all these issues, except for the location constraints on the virtual nodes, by envisioning support from the substrate network through node and link migration as well as multi-path routing. We do not restrict the problem space by assuming infinite capacity of the substrate network resources, nor do we assume any specialized VN topologies.

Contrary to the algorithms proposed in this paper, all the existing algorithms can clearly be separated into two basic phases:

1) assigning virtual nodes using some greedy heuristics, e.g., assign virtual nodes with higher processing requirements to substrate nodes with more available resources [8], [9], and
2) embedding virtual links onto substrate paths using shortest path algorithms [8] in case of unsplittable flows, or using *multi-commodity flow* algorithms in case of splittable flows [9], [11].

The authors in [14] have proposed a distributed algorithm that simultaneously maps virtual nodes and virtual links without any centralized controller, but the scalability and performance of their algorithm is still not comparable with the centralized ones.

The integer and mixed integer programming approaches have been applied to a number of resource allocation and optimization problems in the networking area. In [15], the authors have proposed an integer programming model to solve the VPN tree computation problem for bandwidth provisioning in VPNs. Techniques of randomized rounding for linear programming relaxations to obtain approximation algorithms was first introduced in [16].

In this paper, we take a formal approach to solve the online VN embedding problem using a mixed integer programming formulation. To the best of our knowledge, this is the first attempt to apply mathematical programming to this problem.

III. Network Model and Problem Description

A. Substrate Network

We model the substrate network as a weighted undirected graph and denote it by $G^S = \left(N^S, E^S\right)$, where N^S is the set of substrate nodes and E^S is the set of substrate links. Each substrate node $n^S \in N^S$ is associated with the CPU capacity weight value $c\left(n^S\right)$ and geographic location $loc\left(n^S\right)$. Each substrate link $e^S\left(i,j\right) \in E^S$ between two substrate nodes i and j is associated with the bandwidth capacity weight value $b\left(e^S\right)$ denoting the total amount of bandwidth.

We denote the set of all substrate paths by $\mathcal{P}^S$, and the set of substrate paths from the source node s to the destination node t by $\mathcal{P}^S\left(s,t\right)$.

Fig. 1 shows a substrate network, where the numbers over the links represent available bandwidths and the numbers in rectangles represent available CPU resources [2].

B. VN Request

Similar to the substrate network, we model VN requests as weighted undirected graphs and denote a VN request by $G^V = \left(N^V, E^V\right)$. We express the requirements on virtual nodes and virtual links in terms of the attributes of the nodes and links of the substrate network. Each VN request has an

[2]We use notations similar to [9] to denote capacities and requirements.

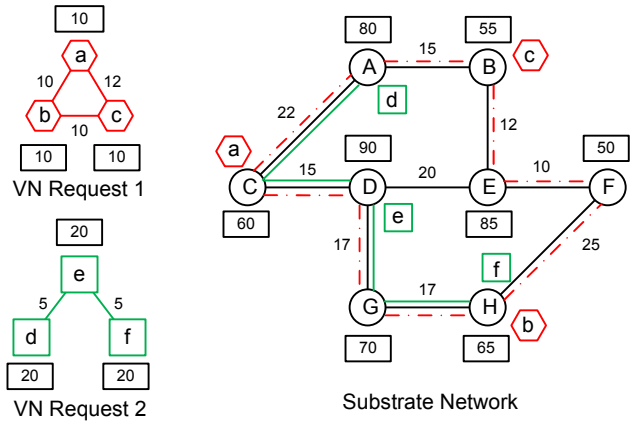

Fig. 1. Mapping of VN requests onto a shared substrate network.

associated non-negative value D^V expressing how far a virtual node $n^V \in N^V$ can be placed from the location specified by $loc\left(n^V\right)$. D^V can be expressed in terms of physical distance or in terms of permissible delay (e.g., RTT) from $loc\left(n^V\right)$. Fig. 1 shows two VN requests with node and link constraints.

C. Measurement of Substrate Network Resources

The *residual* or the available capacity of a substrate node, $R_N\left(n^S\right)$ is defined as the available CPU capacity of the substrate node $n^S \in N^S$.

$$R_N\left(n^S\right) = c\left(n^S\right) - \sum_{\forall n^V \uparrow n^S} c(n^V)$$

where $x \uparrow y$ denotes that the virtual node x is hosted on the substrate node y.

Similarly, the residual capacity of a substrate link, $R_E\left(e^S\right)$ is defined as the total amount of bandwidth available on the substrate link $e^S \in E^S$.

$$R_E\left(e^S\right) = b\left(e^S\right) - \sum_{\forall e^V \uparrow e^S} b(e^V)$$

where $x \uparrow y$ denotes that the substrate path of the virtual link x passes through the substrate link y.

The available bandwidth capacity of a substrate path $P \in \mathcal{P}^S$ is given by

$$R_E\left(P\right) = \min_{e^S \in P} R_E\left(e^S\right)$$

D. VN Assignment

When a VN request arrives, the substrate network has to determine whether to accept the request or not. If the request is accepted, the substrate network then determines a suitable assignment for the VN and allocates network resources on the substrate nodes and paths selected by that assignment. The allocated resources are released once the VN expires.

The assignment of the VN request V to the substrate network can be decomposed into two major components:

1) Node assignment: Each virtual node from the same VN request [3] is assigned to a *different* substrate node by a mapping $\mathcal{M}_N : N^V \rightarrow N^S$ from virtual nodes to substrate nodes such that for all $n^V, m^V \in N^V$,

$$\mathcal{M}_N\left(n^V\right) \in N^S$$
$$\mathcal{M}_N\left(m^V\right) = \mathcal{M}_N\left(n^V\right), \textit{iff } m^V = n^V$$

subject to

$$c\left(n^V\right) \leq R_N\left(\mathcal{M}_N\left(n^V\right)\right) \quad (1a)$$
$$dis\Big(loc\left(n^V\right), loc\left(\mathcal{M}_N\left(n^V\right)\right)\Big) \leq D^V \quad (1b)$$

where $dis(i,j)$ measures the distance between the location of two substrate nodes i and j.

In Fig. 1, the first VN request has the node mapping $\{a \rightarrow C, b \rightarrow H, c \rightarrow B\}$ and the second VN request has $\{d \rightarrow A, e \rightarrow D, f \rightarrow H\}$. Note that two virtual nodes b and f from different VN requests are mapped onto the same substrate node H.

2) Link assignment: Each virtual link is mapped to a substrate path (unsplittable flow) or a set of substrate paths (splittable flow) between the corresponding substrate nodes that host the end virtual nodes of that virtual link. It is defined by a mapping $\mathcal{M}_E : E^V \rightarrow \mathcal{P}^S$ from virtual links to substrate paths such that for all $e^V = \left(m^V, n^V\right) \in E^V$,

$$\mathcal{M}_E\left(m^V, n^V\right) \subseteq \mathcal{P}^S\Big(\mathcal{M}_N\left(m^V\right), \mathcal{M}_N\left(n^V\right)\Big)$$

subject to

$$R_E\left(P\right) \geq b\left(e^V\right), \forall P \in \mathcal{M}_E\left(e^V\right) \quad (2)$$

The first VN request in Fig. 1 has been assigned the link mapping $\Big\{(a,b) \rightarrow \{(C,D),(D,G),(G,H)\}, (a,c) \rightarrow \{(C,A),(A,B)\}, (b,c) \rightarrow \{(H,F),(F,E),(E,B)\}\Big\}$, and the second VN request has the link mapping $\Big\{(d,e) \rightarrow \{(A,C),(C,D)\}, (e,f) \rightarrow \{(D,G),(G,H)\}\Big\}$.

E. Objectives

Our main interest in this paper is to propose online VN embedding algorithms that map multiple VN requests with node and link constraints. We also want to increase revenue and decrease cost of the InP in the long run, in addition to balancing load of the substrate network resources.

Similar to the previous work in [8], [9], we define the revenue of a VN request as:

$$\mathbb{R}(G^V) = \sum_{e^V \in E^V} b\left(e^V\right) + \sum_{n^V \in N^V} c\left(n^V\right) \quad (3)$$

While revenue gives an insight into how much an InP will gain by accepting a VN request, it is not very useful without knowing the cost the InP will incur for embedding that request.

[3]Even though multiple virtual nodes from different VN requests can be mapped to the same substrate node.

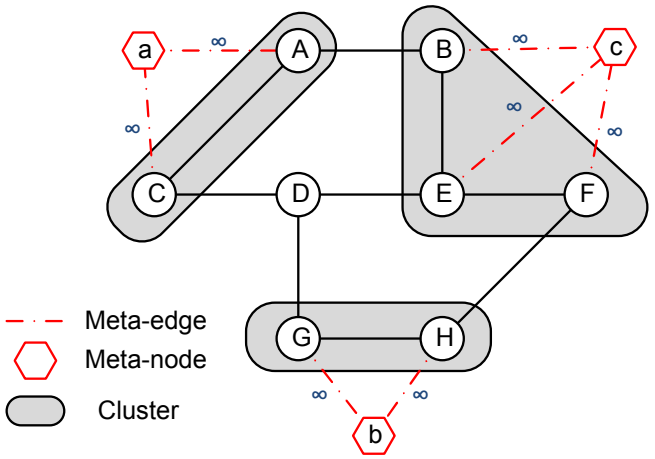

Fig. 2. Construction of an augmented substrate graph with meta-nodes and meta-edges for a VN request.

We define the cost of embedding a VN request as the sum of total substrate resources allocated to that VN.

$$\mathbb{C}(G^V) = \sum_{e^V \in E^V} \sum_{e^S \in E^S} f_{e^S}^{e^V} + \sum_{n^V \in N^V} c(n^V) \qquad (4)$$

where $f_{e^S}^{e^V}$ denotes the total amount of bandwidth allocated on the substrate link e^S for virtual link e^V. We use a modified version of (4) as the objective function of our MIP formulation.

IV. Mixed Integer Programming Formulation for Optimal VN Embedding

A. Augmented Substrate Graph Construction

In order to coordinate the node mapping phase with its link mapping counterpart, the base substrate network must be extended to create an *augmented substrate graph* using the location requirement of the virtual nodes as the basis for the extension. Since each $n^V \in N^V$ has an associated constraint $loc\left(n^V\right)$ on its possible placement, we can create one *cluster* for each virtual node ($\left|N^V\right|$ in total) in the substrate network with radius D^V. We denote such a cluster by $\Omega(n^V)$ and call it the Ω set of the virtual node n^V.

$$\Omega(n^V) = \left\{ n^S \in N^S | dis\left(loc\left(n^V\right), loc\left(n^S\right)\right) \leq D^V \right\}$$

In Fig. 2, substrate nodes B, E, and F are within D^V distance of the virtual node c, hence $\Omega(c) = \{B, E, F\}$.

For each $n^V \in N^V$ we create a corresponding *meta-node* $\mu(n^V)$, and we connect $\mu(n^V)$ with all the substrate nodes belonging to $\Omega(n^V)$ using *meta-edges* with infinite bandwidth. We will sometimes write the Ω set as $\Omega(m)$ instead of $\Omega(n^V)$, where $m = \mu(n^V)$. We combine all the meta-nodes and meta-edges with the substrate graph to create the augmented substrate graph $G^{S'} = (N^{S'}, E^{S'})$, where

$$N^{S'} = N^S \cup \{\mu(n^V) \mid n^V \in N^V\}$$
$$E^{S'} = E^S \cup \{(\mu(n^V), n^S) \mid n^V \in N^V, n^S \in \Omega(n^V)\}$$

An example of augmented graph construction is shown in Fig. 2.

B. MIP Formulation

The VN embedding problem can now be formulated as a mixed integer $|E^V|$-commodity flow problem. We consider each virtual edge e_i^V $\left(1 \leq i \leq |E^V|\right)$ as a commodity with source and destination nodes s_i and t_i, respectively $\left(\forall i,\ s_i, t_i \in N^{S'} \setminus N^S\right)$. Each flow, in this setting, starts from a meta-node and ends in another meta-node. By introducing restrictions on the meta-edges, each meta-node $\mu(n^V)$ can be forced to choose only one meta-edge to connect itself to an actual substrate node in $\Omega(n^V)$. This effectively selects a substrate node for each meta-node, i.e., maps the virtual node corresponding to that meta-node to a substrate node. At the same time, all the virtual edges (i.e., flows) are also mapped efficiently inside the substrate network with the help of path splitting. We present the MIP formulation in the following manner.

VNE_MIP

Variables:

- f_{uv}^i: A flow variable denoting the total amount of flow from u to v on the substrate edge (u, v) for the i'th virtual edge.
- x_{uv}: A binary variable, which has the value '1' if $\sum_i \left(f_{uv}^i + f_{vu}^i\right) > 0$; otherwise, it is set to '0'.

Objective:

$$\text{minimize} \quad \sum_{uv \in E^S} \frac{\alpha_{uv}}{R_E(u,v) + \delta} \sum_i f_{uv}^i + \sum_{w \in N^S} \frac{\beta_w}{R_N(w) + \delta} \sum_{m \in N^{S'} \setminus N^S} x_{mw}\, c(m) \qquad (5)$$

Constraints:

- Capacity Constraints:

$$\sum_i (f_{uv}^i + f_{vu}^i) \leq R_E(u,v)\ x_{u,v}, \forall u, v \in N^{S'} \qquad (6)$$

$$R_N(w) \geq x_{mw}\, c(m), \forall m \in N^{S'} \setminus N^S, \forall w \in N^S \qquad (7)$$

- Flow Related Constraints:

$$\sum_{w \in N^{S'}} f_{uw}^i - \sum_{w \in N^{S'}} f_{wu}^i = 0, \forall i, \forall u \in N^{S'} \setminus \{s_i, t_i\} \qquad (8)$$

$$\sum_{w \in N^{S'}} f_{s_i w}^i - \sum_{w \in N^{S'}} f_{w s_i}^i = b(e_i^V), \forall i \qquad (9)$$

$$\sum_{w \in N^{S'}} f_{t_i w}^i - \sum_{w \in N^{S'}} f_{w t_i}^i = -b(e_i^V), \forall i \qquad (10)$$

- Meta and Binary Constraints:

$$\sum_{w \in \Omega(m)} x_{mw} = 1, \forall m \in N^{S'} \setminus N^S \quad (11)$$

$$\sum_{m \in N^{S'} \setminus N^S} x_{mw} \leq 1, \forall w \in N^S \quad (12)$$

$$x_{uv} \leq R_E(u,v), \forall u,v \in N^{S'} \quad (13)$$

$$x_{uv} = x_{vu}, \forall u,v \in N^{S'} \quad (14)$$

- Domain Constraints:

$$f^i_{uv} \geq 0, \forall u,v \in N^{S'} \quad (15)$$

$$x_{uv} \in \{0,1\}, \forall u,v \in N^{S'} \quad (16)$$

Remarks:

- The objective function (5) of the MIP tries to minimize the cost of embedding the VN request as well as balance the load. By dividing the cost with the residual capacity, it is ensured that the resources with more residual capacities are preferred over the resources with less residual capacities. $1 \leq \alpha_{uv} \leq R_E(u,v)$ and $1 \leq \beta_w \leq R_N(w)$ are parameters to control the importance of load balancing while embedding a request. $\delta \rightarrow 0$ is a small positive constant to avoid dividing by zero in computing the objective function.
- Constraint set (6) and (7) contains the node and edge capacity bounds. Summing up f^i_{uv} and f^i_{vu} in (6) ensures that the summation of flows on both directions of the undirected edge (u,v) remains within its available bandwidth.
- Constraint set (8), (9), and (10) refer to the flow conservation conditions, which denote that the net flow to a node is zero, except for the source node s_i and the sink node t_i.
- Constraint sets (11) and (12) are related to the augmented portion of the substrate graph. Constraint set (11) makes sure that only one substrate node is selected for each meta-node, whereas constraint set (12) ensures that no more than one meta-node is placed on a substrate node.
- Constraint sets (13) and (14) together with (2) ensure that x_{uv} is set whenever there is any flow in either direction of the substrate edge (u,v).
- Finally, constraint sets (15) and (16) denote the real and binary domain constraints on the variables f^i_{uv} and x_{uv}, respectively.

V. LP Relaxation and Rounding-Based Algorithms

Since solving an MIP is known to be computationally intractable [10], simultaneous node and link embedding using **VNE_MIP** is practically infeasible. Hence we relax the integer constraints (16) of the MIP, and obtain the following linear program (**VNE_LP_RELAX**). Once we have the LP solution, we use deterministic and randomized rounding techniques to obtain integer values for the variable x and embed VN requests.

VNE_LP_RELAX

Objective:

$$\text{minimize} \quad \sum_{uv \in E^S} \frac{\alpha_{uv}}{R_E(u,v)+\delta} \sum_i f^i_{uv} + \sum_{w \in N^S} \frac{\beta_w}{R_N(w)+\delta} \sum_{m \in N^{S'} \setminus N^S} x_{mw}\, c(m) \quad (17)$$

Constraints:

- Capacity Constraints:

$$\sum_i (f^i_{uv} + f^i_{vu}) \leq R_E(u,v)\, x_{u,v}, \forall u,v \in N^{S'}$$

$$R_N(w) \geq x_{mw}\, c(m), \forall m \in N^{S'} \setminus N^S, \forall w \in N^S$$

- Flow Related Constraints:

$$\sum_{w \in N^{S'}} f^i_{uw} - \sum_{w \in N^{S'}} f^i_{wu} = 0, \forall i, \forall u \in N^{S'} \setminus \{s_i, t_i\}$$

$$\sum_{w \in N^{S'}} f^i_{s_i w} - \sum_{w \in N^{S'}} f^i_{w s_i} = b(e^V_i), \forall i$$

$$\sum_{w \in N^{S'}} f^i_{t_i w} - \sum_{w \in N^{S'}} f^i_{w t_i} = -b(e^V_i), \forall i$$

- Meta and Relaxed Binary Constraints:

$$\sum_{w \in \Omega(m)} x_{mw} = 1, \forall m \in N^{S'} \setminus N^S$$

$$\sum_{m \in N^{S'} \setminus N^S} x_{mw} \leq 1, \forall w \in N^S x_{uv} \leq R_E(u,v), \forall u,v \in N^{S'}$$

$$x_{uv} = x_{vu}, \forall u,v \in N^{S'}$$

- Domain Constraints:

$$f^i_{uv} \geq 0, \forall u,v \in N^{S'}$$

$$x_{uv} \geq 0, \forall u,v \in N^{S'} \quad (18)$$

Remarks:

- The domain constraint set (18) on the x_{uv} variables has been relaxed.
- The rest of the constraints are similar to the ones in **VNE_MIP**.

A. Deterministic Rounding Based Virtual Network Embedding Algorithm (D-ViNE)

D-ViNE (Fig. 3) takes online VN requests as inputs and maps them onto the substrate network one at a time. It takes decisions based only on the past VN requests that it has already seen, i.e., D-ViNE uses no look-ahead. Since the integer domain constraints (16) on the x variables have already been relaxed, we no longer get integer values for the x variables. Instead, we employ deterministic rounding technique to get integer values for x. We introduce $\varphi : N^S \rightarrow \{0,1\}$, which is initially set to zero for all $n^S \in N^S$ signifying that all the substrate nodes are initially unused. Whenever a virtual node

1: **procedure** D-ViNE($G^V = (N^V, E^V)$)
2: Create augmented substrate graph $G^{S'} = \left(N^{S'}, E^{S'}\right)$
3: Solve **VNE_LP_RELAX**
4: **for all** $n^S \in N^S$ **do**
5: $\varphi(n^S) \leftarrow 0$
6: **end for**
7: **for all** $n \in N^V$ **do**
8: **if** $\Omega(n) \cap \{n^S \in N^S | \varphi(n^S) = 1\} = \emptyset$ **then**
9: VN request cannot be satisfied
10: **return**
11: **end if**
12: **for all** $z \in \Omega(n)$ **do**
13: $p_z \leftarrow (\sum_i f^i_{\mu(n)z} + f^i_{z\mu(n)}) x_{\mu(n)z}$
14: **end for**
15: Let $z_{max} = \arg\max_{z \in \Omega(n)} \{p_z | \varphi(z) = 0\}$
16: set $\mathcal{M}_N(n) \leftarrow z_{max}$
17: $\varphi(z_{max}) \leftarrow 1$
18: **end for**
19: Solve MCF to map virtual edges.
20: Update residual capacities of the network resources.
21: **end procedure**

Fig. 3. D-ViNE: Deterministic rounding based Virtual Network Embedding algorithm

1: **procedure** R-ViNE($G^V = (N^V, E^V)$)
2: Create augmented substrate graph $G^{S'} = \left(N^{S'}, E^{S'}\right)$
3: Solve **VNE_LP_RELAX**
4: **for all** $n^S \in N^S$ **do**
5: $\varphi(n^S) \leftarrow 0$
6: **end for**
7: **for all** $n \in N^V$ **do**
8: **if** $\Omega(n) \cap \{n^S \in N^S | \varphi(n^S) = 1\} = \emptyset$ **then**
9: VN request cannot be satisfied
10: **return**
11: **end if**
12: **for all** $z \in \Omega(n)$ **do**
13: $p_z \leftarrow (\sum_i f^i_{\mu(n)z} + f^i_{z\mu(n)}) x_{\mu(n)z}$
14: **end for**
15: $p_{sum} \leftarrow \sum_{z \in \Omega(n)} p_z$
16: **for all** $z \in \Omega(n)$ **do**
17: $p_z \leftarrow p_z / p_{sum}$
18: **end for**
19: set $\mathcal{M}_N(n) \leftarrow z$ with probability p_z
20: $\varphi(z) \leftarrow 1$ with probability p_z
21: **end for**
22: solve MCF to map virtual edges.
23: Update residual capacities of the network resources.
24: **end procedure**

Fig. 4. R-ViNE: Randomized rounding based Virtual Network Embedding algorithm

is mapped to a particular physical node n^S, we set $\varphi(n^S)$ to 1 to ensure that no substrate node is used twice for the same VN request.

1) Description and Discussion: The procedure begins by creating an augmented substrate graph, $G^{S'} = \left(N^{S'}, E^{S'}\right)$ for the VN request $G^V = \left(N^V, E^V\right)$ using the augmentation method described in Section IV-A. Next it solves **VNE_LP_RELAX** to get a fractional solution which is at least as good as the integer solution of **VNE_MIP**. For each virtual node, D-ViNE first checks whether there are any unmapped substrate nodes within its feasible region (the substrate nodes in the virtual nodes Ω set). If the corresponding Ω set is empty, D-ViNE stops the embedding process immediately and rejects the VN request. Otherwise the *deterministic rounding procedure* is initiated in line 12.

For each virtual node n, D-ViNE calculates a value p_z for each substrate node $z \in \Omega(n)$ in its cluster. p_z is calculated as the product of the value $x_{\mu(n)z}$ and the total flow passing through the meta-edge $\mu(n)z$ in both directions. The reason behind using this multiplication instead of just $x_{\mu(n)z}$ is as follows. In the MIP solution x_{uv} is set to binary values based on the presence of flows in either direction in the edge (u, v). When the binary constraint x is relaxed, one might expect that the fractional values of x_{uv} would also be proportional to the total flow in the edge (u, v). But during the LP relaxation process, the correlation between the flow variable f and the binary variable x is lost. It is because a linear program tries to optimize the objective function without violating the constraints; it does not care about the values as long as they are within their permitted domains. As a result, in the relaxed linear program, it is possible that the f values are very high and the corresponding x values are very low or vice versa. Multiplying the f and x values thwarts the possibility of selecting a substrate node based on high x value but very low f value on its corresponding meta-edge and vice versa. The ones that have better values for both the variables f and x are more likely to be in the solution of the MIP than others. D-ViNE maps the virtual node n onto the unmapped substrate node z (i.e., $\varphi(z) = 0$) with the highest p_z value, breaking ties arbitrarily.

Once all the virtual nodes have been mapped to different substrate nodes, D-ViNE applies the multi-commodity flow algorithm to map the virtual edges in E^V onto the substrate paths. One can also use shortest path algorithms when path splitting is not supported by the substrate network. Finally, D-ViNE updates the residual capacities of the substrate nodes and links to prepare for the next request.

2) Time Complexity: An important aspect of D-ViNE is that the multi-commodity flow algorithm is executed twice; first, during the node mapping phase (since **VNE_LP_RELAX** is a linear programming relaxation of the original mixed integer multi-commodity flow problem), and second, during the edge mapping phase. Since the multi-commodity flow algorithm can be solved in polynomial-time using either the ellipsoid algorithm or Karmarkar's interior point algorithm for linear programming [10]; hence, D-ViNE is a polynomial time algorithm.

B. Randomized Rounding Based Virtual Network Embedding Algorithm (R-ViNE)

R-ViNE (Fig. 4) is quite similar to D-ViNE except that it uses randomized rounding instead of deterministic rounding. Once the p_z values are calculated as in D-ViNE, R-ViNE normalizes those values within 0 to 1 range. The normalized values for each $z \in \Omega(n)$ correspond to the probabilities of n being mapped to z by the optimal MIP. R-ViNE selects a substrate node $z \in \Omega(n)$ to map a virtual node n with probability p_z. The remainder of this algorithm is similar to its deterministic counterpart, and it is clear that this algorithm also runs in polynomial-time.

VI. Performance Evaluation

In this section, we first describe the evaluation environment, and then present our main evaluation results. Our evaluation focuses primarily on quantifying the advantage of coordinating node mapping and link mapping phases in terms of acceptance ratio, revenue and cost. We also compare D-ViNE and R-ViNE with existing algorithms modified to fit into our model.

A. Simulation Settings

We have implemented a discrete event simulator to evaluate the performance of our algorithms which is freely available at [17]. The substrate network topologies in our experiments are randomly generated with 50 nodes using the GT-ITM tool [18] in (25×25) grids. Each pair of substrate nodes is randomly connected with probability 0.5. The cpu and bandwidth resources of the substrate nodes and links are real numbers uniformly distributed between 50 and 100. We assume that VN requests arrive in a Poisson process with an average rate of 4 VNs per 100 time units, and each one has an exponentially distributed lifetime with an average of $\mu = 1000$ time units. In each VN request, the number of virtual nodes is randomly determined by a uniform distribution between 2 and 10 following similar setups to previous works [8], [9]. The average VN connectivity is fixed at 50%. The cpu requirements of the virtual nodes are real numbers uniformly distributed between 0 to 20 and the bandwidth requirements of the virtual links are uniformly distributed between 0 to 50. We have used the open source mixed integer programming library glpk [19] to solve **VNE_LP_RELAX**.

B. Comparison Method

In our evaluation, we have compared six algorithms that combine different node mapping and link mapping strategies including our contributions and algorithms from previous research [8], [9] modified to fit into our model (i.e., no reconfiguration). The notations that we use to refer to different algorithms are enumerated in Table I.

C. Evaluation Results

We use several performance metrics for evaluation purposes in our experiments. We measure the average acceptance ratio, revenue ($\mathbb{R}$), and provisioning cost ($\mathbb{C}$) for VN requests over time. We also measure the average node utilization and average

TABLE I
Compared Algorithms

Notation	Algorithm Description
D-ViNE	Deterministic Node Mapping with Splittable Link Mapping using MCF
R-ViNE	Randomized Node Mapping with Splittable Link Mapping using MCF
G-SP [8]	Greedy Node Mapping with Shortest Path Based Link Mapping
G-MCF [9]	Greedy Node Mapping with Splittable Link Mapping using MCF
D-ViNE-SP	Deterministic Node Mapping with Shortest Path Based Link Mapping
D-ViNE-LB	Deterministic Node Mapping with Splittable Link Mapping using MCF, where $\alpha_{uv} = \beta_w = 1, \forall\, u, v, w \in N^S$

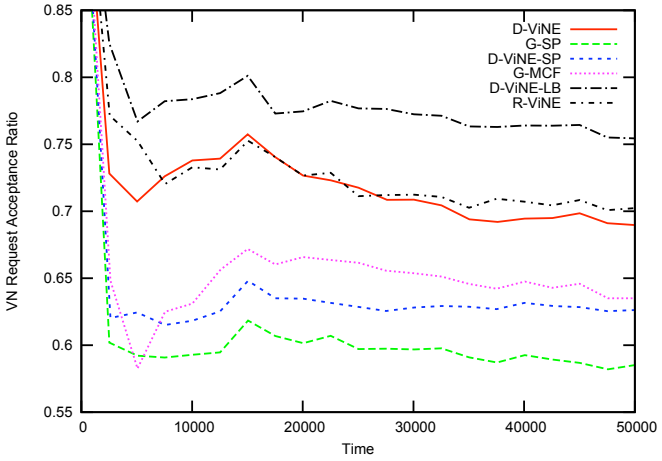

Fig. 5. VN request acceptance ratio over time

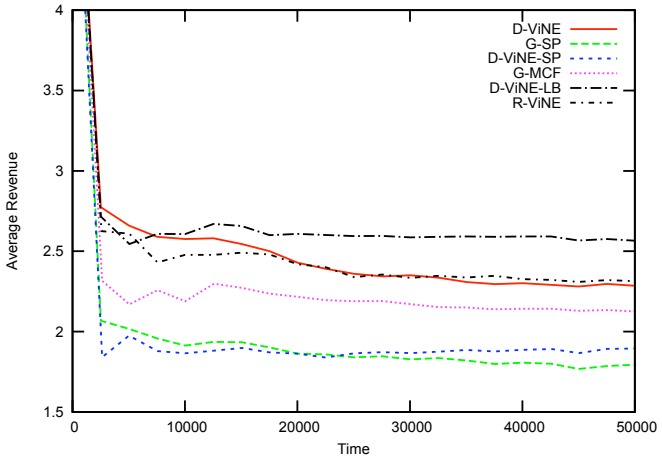

Fig. 6. Time average of generated revenue

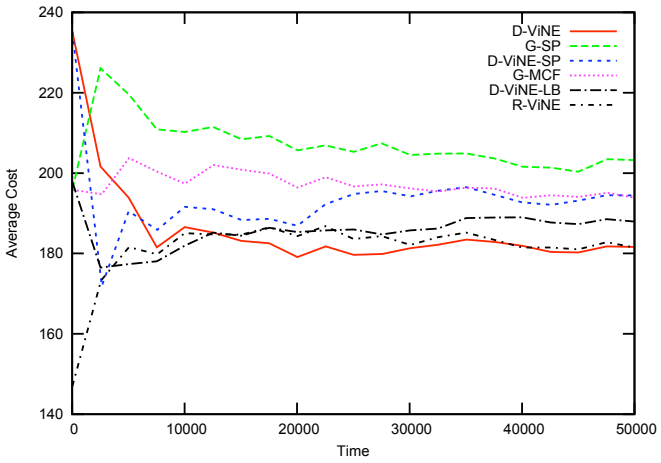

Fig. 7. Average cost of accepting VN requests over time

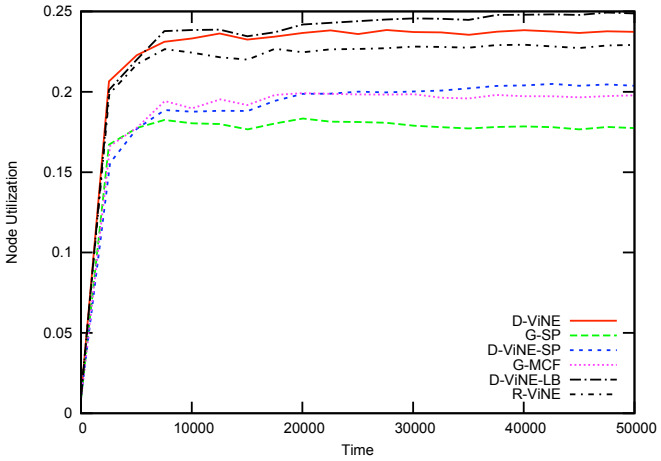

Fig. 8. Average node utilization

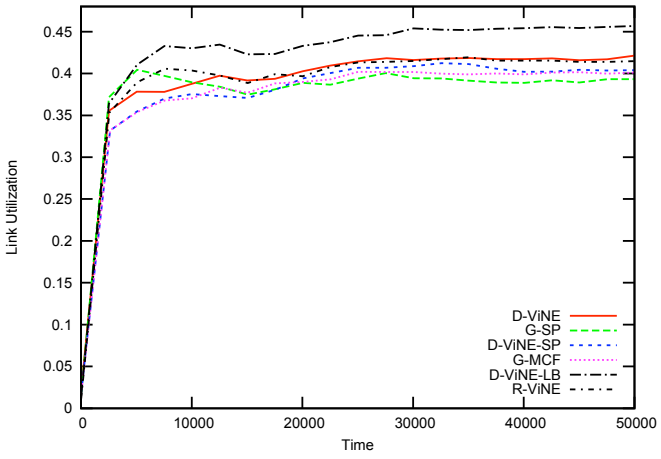

Fig. 9. Average link utilization

link utilization of the substrate network. In all these cases we plot the performance metrics against time to show how each of these algorithms (Table I) actually perform in the long run. We summarize our key observations in the following.

(1) Coordinated node and link mapping leads to higher acceptance ratio and larger revenue. Fig. 5 and Fig. 6 depict that the proposed algorithms, D-ViNE and R-ViNE, lead to better acceptance ratio as well as higher revenue than the existing algorithms (G-SP and G-MCF) through coordinated node and link mapping. Having higher revenue along with better acceptance ratio implies that our proposed algorithms actually embed VN requests that generate more revenue, instead of embedding smaller VN requests just to increase the acceptance ratio.

(2) Load balancing further increases the acceptance ratio and the revenue. From Fig. 5 and Fig. 6, it is evident that D-ViNE-LB generates more revenue and accepts more VN requests than the basic D-ViNE algorithm. In D-ViNE-LB, the value of the objective function (17) of **VNE_LP_RELAX** depends on the residual capacity of the network resources in addition to the provisioning cost (α and β values are set to 1 here). The lower the residual capacity of a particular node or link, the higher the value of the objective function. As a result, D-ViNE-LB tries to avoid highly utilized nodes and links as long as it can, leaving those critical resources available for the VN requests that absolutely need them.

(3) Randomization can lead to better performance. It is well established in the algorithm design literature that randomization allows efficient solutions to many intractable problems in polynomial time with low probability of error. Our experiments show that the randomized version of our VN embedding algorithm (R-ViNE) performs better than its deterministic counterpart (D-ViNE) in terms of acceptance ratio and revenue generation (Fig. 5 and Fig. 6).

In addition to that, for networks with large numbers of nodes randomization has been shown to be effective for load balancing [20]. This phenomena is also visible in our experiments, since R-ViNE performs similar to D-ViNE-LB in most scenarios.

(4) Load balancing slightly increases the average provisioning cost. While load balancing increases revenue and acceptance ratio by avoiding highly utilized resources, it runs the risk of increasing the average provisioning cost as shown in Fig 7. Since D-ViNE-LB tries to avoid highly utilized resources, sometimes it ends up suggesting a longer path to map a particular virtual edge which eventually sums up to slightly higher average provisioning cost in the long run.

(5) Coordination increases resource utilization. Fig. 8 and Fig. 9 depict the average utilization of substrate nodes and substrate links for different VN embedding algorithms. Since D-ViNE-LB has the highest acceptance ratio, it also has the highest node and link utilization.

However, D-ViNE-LB achieves a relatively higher gain in link utilization over its counterparts than in node utilization.

We believe that the reason behind this is the distributive nature of D-ViNE-LB algorithm. In order to avoid links with lower residual capacities, i.e., in order to minimize (5), D-ViNE-LB uses longer paths containing more substrate links with higher residual capacities to embed virtual links.

VII. Conclusion

To make network virtualization an integral part of the future Internet architecture, efficient and practical algorithms for VN embedding are specially required. In this paper, we proposed algorithms for VN embedding that differ from the previous algorithms by introducing coordination between node and link mapping phases. We argued that this coordination greatly increases the solution space and the quality of the heuristic algorithms. To this end we first formulated the embedding problem as a mixed integer program. We then relaxed the integer constraints and used deterministic and randomized rounding techniques to obtain polynomial-time solvable algorithms for node mapping. The node mapping phase combined with the multi-commodity flow based link mapping phase in our algorithms outperformed the existing approaches in terms of acceptance ratio, revenue, and provisioning cost, as shown through simulation.

However, there are a number of issues that remain unresolved in this work and can be good starting points for further research in this direction. First and foremost is the analysis of theoretical approximation factors of the proposed algorithms in the worst case. To this end, we are currently working on developing a primal-dual based analysis framework to obtain good lower bounds on the performance of D-ViNE and R-ViNE. Finding out advanced economic models, instead of the simple revenue model used in the existing literature, for VN pricing is another important research topic that needs further attention. Finally, available approaches to directly solve integer and mixed integer programs (e.g., column generation) can be employed to develop efficient algorithms to obtain optimal or near-optimal solutions for the original mixed integer formulation (**VNE_MIP**) without any relaxation.

Acknowledgment

This research was partially supported by the Natural Science and Engineering Council of Canada (NSERC) and partially by WCU (World Class University) program through the Korea Science and Engineering Foundation funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0).

References

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

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

[3] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI veritas: Realistic and controlled network experimentation," in *Proceedings of SIGCOMM*, 2006, pp. 3–14.

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

[5] D. Andersen, "Theoretical approaches to node assignment," Unpublished Manuscript, http://www.cs.cmu.edu/~dga/papers/andersen-assign.ps, 2002.

[6] J. Fan and M. Ammar, "Dynamic topology configuration in service overlay networks - a study of reconfiguration policies," in *Proceedings of IEEE INFOCOM*, 2006.

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

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

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

[10] A. Schrijver, *Theory of linear and integer programming*. New York, NY, USA: John Wiley & Sons, Inc., 1986.

[11] W. Szeto, Y. Iraqi, and R. Boutaba, "A multi-commodity flow based approach to virtual network resource allocation," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'03)*, 2003, pp. 3004–3008.

[12] A. Gupta, J. M. Kleinberg, A. Kumar, R. Rastogi, and B. Yener, "Provisioning a virtual private network: A network design problem for multicommodity flow," in *Proceedings of ACM STOC*, 2001, pp. 389–398.

[13] R. Ricci, C. Alfeld, and J. Lepreau, "A solver for the network testbed mapping problem," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 65–81, April 2003.

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

[15] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener, "Algorithms for provisioning virtual private networks in the hose model," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 565–578, 2002.

[16] P. Raghavan and C. D. Tompson, "Randomized rounding: a technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.

[17] "ViNE-Yard," http://www.mosharaf.com/ViNE-Yard.tar.gz.

[18] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an Internetwork," in *Proceedings of IEEE INFOCOM*, 1996, pp. 594–602.

[19] "GNU Linear Programming Kit," http://www.gnu.org/software/glpk/.

[20] M. Mitzenmacher, A. W. Richa, and R. Sitaraman, "The power of two random choices: A survey of techniques and results," in *Handbook of Randomized Computing*. Kluwer Academic Press, 2001, pp. 255–312.