Virtual Network Mapping with Traffic Matrices

Cong Wang, Shashank Shanbhag and Tilman Wolf
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA
{cwang,sshanbha,wolf}@ecs.umass.edu

***Abstract*—Network virtualization is a core technology in next-generation networks to overcome the ossification problem that is observed in the current Internet. The key idea of network virtualization is to split physical network resources into multiple logical networks, each supporting different network services and functionalities. One of the key challenges for virtual network infrastructure providers is to efficiently allocate network resources based on virtual network requests, which is referred to as the virtual network mapping problem. While several algorithms have been proposed previously to solve this mapping problem, their effectiveness is limited since virtual requests specify the internal topology of the virtual network. In this paper, we argue that such internal topologies lead to unnecessary constraints and less efficient solutions. Instead, we propose an alternate formulation of the problem that represents virtual network requests as traffic matrices. We provide a solutions to solving this traffic-matrix-based mapping problem using a mixed integer programming formulation. Our simulation results show that our approach can map significantly more virtual network requests on a physical network infrastructure than previous mapping algorithms and thus improves the efficient use of networking resources in virtual networks.**

***Index Terms*—mixed integer programming, network virtualization, traffic matrix, virtual network embedding**

I. Introduction

In the past few decades, the Internet has become a globally successful system for data communication. However, the current Internet architecture is based on the assumption that a single network-layer protocol is used among all devices [4]. This design constraint has become one of the main obstacles to deploying innovative new protocols at the network layer.

Network virtualization provides a novel solution for solving this network ossification problem by splitting a single physical network (substrate network) into multiple, independent logical networks (virtual networks) [1]. This technique is currently used in experimental testbeds (e.g., Emulab [11] and GENI [8]) to allow researchers to experiment with networks using different protocol stacks in parallel. This idea of sharing one infrastructure among multiple virtual networks can be expected to be extended to the entire future Internet.

One of the important unsolved problems in network virtualization is the allocation of substrate network resources to virtual network requests. When a new virtual network request arrives, the infrastructure providers needs to assign each node and link of the virtual network to the corresponding substrate node and link (or path) in the physical network. In related work, several efficient virtual network mapping algorithms have been developed for providing solution to this problem (e.g., D-ViNe [2] and vnmFlib [7]).

A key shortcoming of existing solutions to the network virtualization problem is that these algorithms use *network topologies* to express the structure of a virtual network. These topologies are then mapped to the physical network. The main problem with this approach is that the use of a topology places many (often artificial) constraints on the solutions that the virtual network mapping algorithm can find. Specifically, there are two major flaws in current topology-based virtual network mapping: (1) *Unnecessary topology constraints:* Nodes and links on the "inside" of the virtual network request determine the topology of the request. However, these nodes and links may not match well with the underlying substrate network. When attempting to map this request, the substrate may need to use excessive resources to accommodate these internal nodes and links. (2) *Inconsistent internal topology:* For each node and link of a topology-based request, the bandwidth and computational resources are specified independently. It is possible that a request exhibits internal inconsistencies where resource requests do not match with feasible traffic patterns. This case may especially exist in experiments that uses synthetic virtual requests as the input.

In practice, the internal topology of a virtual network should not matter as long as the mapped structure can provide connectivity at a sufficient level of service (e.g., bandwidth). Therefore, there is an opportunity to fundamentally reconsider the virtual network mapping problem and provide a problem formulation and solution that can achieve much more efficient use of networking resources.

In this paper, we provide this alternate formulation of the virtual network mapping problem. Our main idea is to use *traffic matrices* instead of topologies to represent the requirements for a virtual network. We present how such virtual network requests can be mapped with a novel mapping algorithm. Our evaluation results show that the traffic-matrix-based algorithm can find more efficient mapping solutions than existing algorithms and therefore can accommodate 63%-196% more virtual networks on a physical network infrastructure (while providing the same level of end-to-end connectivity and bandwidth). This improvement in resource utilization can considerably improve the operation of virtual networks in the future Internet.

The remainder of this paper is organized as follows. Section II provides an overview of the related work. Following that, Section III Models the network model and the virtual network mapping assignment. In Section IV, we provide the MIP formulation for solving the traffic-matrix-based virtual

network mapping problem. Section V presents simulation results for evaluation of the proposed formulation. Section VI summarizes and concludes this paper.

II. Related Work

The virtual network mapping problem has been surveyed extensively in [3]. The problem statement is similar to the virtual private network (VPN) design problem, which is shown in [5], [6]. Both of them aim to allocate physical network resources based on requested topologies. However, only link utilization is considered in the VPN embedding problem, where traffic matrices are specified without any constraints on network node resources. Thus, most VPN design algorithms are based on seeking the optimal path for each source/destination pair.

In the network virtualization problem, the mapping algorithms need to further consider the node resource utilization (e.g., to perform the processing associated with the protocol stack used in the virtual network). As a result, the traditional virtual network mapping algorithms use the notion of "topology" to specify both the substrate network and virtual network, e.g., [2], [7]. A typical topology contains the specific layout pattern of interconnections of the various network elements (e.g., links, nodes) with a set of specific constraints (e.g., CPU capacity, link bandwidth). Using a topology-based approach, the entity specifying a virtual network request needs to fully understand the physical network infrastructure in order to provide a request that can make optimal use of resources. When lacking knowledge of the substrate network structure, the topology specified in the request may result in low mapping quality or even mapping failure.

In our work, we propose the use of traffic matrices as input for virtual network mapping. Instead of specifically describing each node and link condition of the whole virtual network, the traffic matrix only specifies the necessary the traffic that flows between each pair of the end nodes. The internal topology of a virtual network then is determined by the mapping algorithm with a minimum number of constraints. To the best of our knowledge, this is the first attempt to base virtual network requests on traffic matrices to solve the virtual network mapping problem.

III. Problem Formulation

In this section, we describe the general virtual network embedding problem model, including the substrate network model, virtual network request model, and the initial approach for virtual network mapping process.

A. Substrate Network

We define the substrate network as an undirected graph $G^s = (N^s, E^s)$, where N^s denotes the set of nodes, and E^s denotes the set of edges in the physical substrate network. The substrate nodes and edges are associated with their constraints, respectively. We model the constraints as a set $C^s = C_1^s, C_2^s, ..., C_n^s$ for each node/link in G^s. The substrate node $n^s \in N^s$ is associated with the CPU capacity weight value $c(n^s)$ and geographic location $loc(n^s)$, which is represented by the coordinate of nodes. Each of the substrate link $e^s(i,j) \in E^s$ between two substrate nodes i and j is associated with the bandwidth capacity weight value $b(e^s)$ representing the total amount of bandwidth on that link. The element weights are calculated as the sum of all the individual constraints or weights of edges.

B. Virtual Network Request

The virtual network requests are modeled as a matrix treating the virtual network as fully connected, but it only contains the end nodes and requested bandwidth between each pair of end nodes. We denote a virtual network request as $G^v = (N^v, E^v)$ and have the constraint set $C^v = C_1^v, C_2^v, ..., C_n^v$ for each of the node/link in G^v. For each node/link $(n^s, e^s) \in (N^s, E^s)$ with available CPU and bandwidth resources, $c(n^v)$ and $b(n^v)$ represent the request for corresponding resources on that node/link, respectively. A sample traffic matrix is shown in Fig. 1.

C. The Virtual Network Mapping Assignment

The key question for virtual network mapping is to design an algorithm, with given substrate topology $G^s = (V^s, E^s)$. When the virtual network request is to be mapped, we first determine whether the request can be handled by the substrate network. If the request is accepted, then we assign the substrate nodes and links and their resources to the request. The allocated resources are released once the virtual network request expires. In section IV, we provide detailed description of fulfilling virtual network mapping task.

D. Objective and Metrics for Virtual Network Mapping

One goal of our work is to select appropriate metrics to evaluate the performance of virtual network mapping process and to design an efficient algorithm that maps multiple virtual network requests under node and link constraints. Currently, there exist several metrics for measuring the quality of virtual network mapping:

1) *Number of successful mappings:* A most basic metric for evaluating the overall performance of virtual network mapping is the number of virtual requests that can be allocated to a infrastructure with limited resources. This metric is typically measured by repeatedly allocating virtual requests. Since for most algorithms the first failure means that most substrate resources are exhausted, we usually consider the number of successive successful mapping until the first failure for this metric.
2) *Virtual mapping revenue:* We use the same definition of virtual mapping revenue and cost as shown in some of the related works such as [12]. The mapping revenue of a virtual request can be defined as:

$$R(G^v) = \sum_{e^v \in E^v} b(e^v) + \sum_{n^v \in N^v} c(n^v) \qquad (1)$$

As shown in the equation, the revenue is calculated as the sum of all CPU and bandwidth gained by each virtual network mapping. This corresponds to what a customer would pay for.

3) *Virtual mapping cost:* The mapping cost reflects the resources that need to be used to accommodate a request. We define the cost of mapping a virtual network request as:

$$C(G^v) = \sum_{e^v \in E^v} \sum_{e^s \in E^s} f_{e^s}^{e^v} + \sum_{n^v \in N^v} c(n^v), \quad (2)$$

where $f_{e^s}^{e^v}$ represents the total bandwidths that have been allocated from the substrate link e^s to the virtual request link e^v. The total cost is calculated as the sum of total substrate resources that have been allocated to the virtual request.

In practice, a virtual infrastructure provider is likely concerned about mapping many requests, but also at achieving a high revenue-to-cost ratio.

IV. Virtual Network Mapping with Traffic Matrices

To demonstrate the effectiveness of traffic-matrix-based mapping, we implemented a single-stage virtual network mapping formulation $VHub$, that can map virtual network requests based on traffic matrices onto the physical network. This work is based on an existing topology-based virtual network mapping formulation [10]. Our traffic-matrix-based VHub approach formulates the traffic-matrix-based mapping problem as a p-Hub location problem.

A. The p-Hub Location Problem

The facility location problems seek to place resources at user-selected locations such that the desired objective is minimized. Hub location, a special case of facility location, seeks to obtain the optimized placement of hubs in hub-and-spoke networks so that the demand-weighted flow cost between the demand nodes is minimized. In our work, we treat the end nodes in the traffic matrix as the demand nodes and the substrate nodes as the hub nodes. A typical solution for the p-Hub location problem is to formulate it into a integer program whose objective is to minimize the transportation cost, subject to user-defined constraints.

B. Augmented Graph

In order to coordinate the end node mapping phase with its link mapping counterpart, it is helpful to extend the substrate network to create an augmented substrate graph using the location requirement of the virtual nodes as the basis for the extension. This idea was first introduced in [2].

We combine the virtual network and substrate network through a p-hub median problem that serves the purpose of placing hubs nodes (the virtual nodes) onto the substrate for optimizing the cost of transferring resources between end nodes that are assigned to those hubs. We introduce $G_A = (N_A, E_A)$ to be the augmented graph, which is generated by adding the virtual requests onto substrate according to the location constraints $MaxD$ (the spatial coordinates and the maximum allocation distance). Thereby, we can get the node and edge sets $N = (N^s \cup N^v)$ and $E = (E^s \cup E^l)$, where,

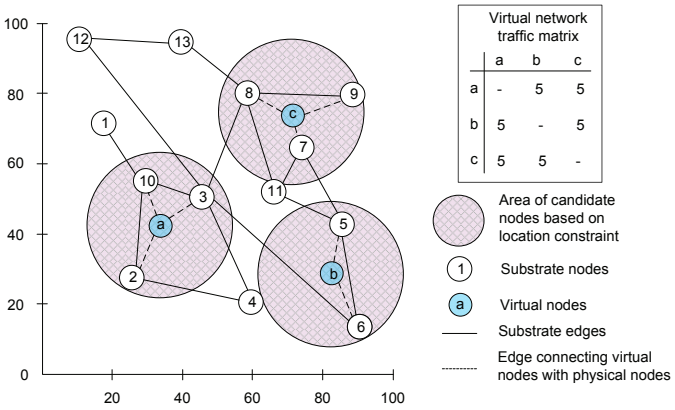

Fig. 1. The construction of augmented graph

$$E^l = \{(v, x) : v \in N^v \cap x \in L(n^v) \cap c(v) \leq c(x)\} \quad (3)$$

In this equation, we define the set $L(n^v)$ as the candidate physical nodes at which a virtual node $v \in N^v$ can be placed:

$$L(n^v) = \{x \in N^s : \Delta_{v,x} \leq MaxD\} \quad (4)$$

where $\Delta_{v,x}$ represents the Euclidean distance between node v and x.

Since E^l are imaginary edges, we assume that the bandwidth capacity is $b(v, x) = \infty$.

Fig. 1 shows the augmented graph of an example virtual network request. The blue and white nodes represent the nodes in virtual network and substrate network, respectively. The virtual edges connects the virtual nodes to their candidate substrate nodes within the range of $MaxD$.

C. Virtual Mapping Through p-Hub Median Problem

We consider the substrate nodes as hub nodes, and the edge nodes specified in the traffic matrices as non-hub nodes, thus the augmented graph can be modeled into a hub & spoke network. The primary goal is to find an optimal path to transfer the flow starts from the origin node and end with the destination node, the task of hub nodes is only to transfer flow between the origin node and destination node. Thus, the traffic matrix mapping problem can be formulated.

Physical nodes in the augmented graph $G_A = (N_A, E_A)$ correspond to origin/destination pairs and possible hub locations. If we denote the Bandwidth requested between virtual nodes u and v as $d_{u,v}$, we can then define the origin and destination flows as follows: Let O_u be the total flow originating at node u given by

$$O_u = \sum_{v \in N^v} d_{u,v}, \forall u \in N_A. \quad (5)$$

Similarly, if D_u is the total flow destined towards node u given by the equation:

$$D_u = \sum_{v \in N^v} d_{v,u}, \forall u \in N_A. \quad (6)$$

We denote each origin-to-destination path with three components as:

- *origin-hub:* Transfer of flow from origin node to a hub with unit transfer cost, χ.
- *hub-hub:* Transfer of flow from the first hub to the last hub connected to the destination with unit transfer cost, α.
- *hub-destination:* Transfer of flow from the last hub to the destination node, with unit transfer cost, δ.

Thus, we could get the total cost of unit flow transferring from a non-hub origin node u to a non-hub destination node v through hubs x and y as $\chi\Delta(u,x) + \alpha\Delta(x,y) + \delta\Delta(y,v)$.

For deciding the flow between each pair of hub nodes, we define the decision variable $Y^u_{x,y}$ as the flow from hub x to hub y originates at u.

Also, we have the binary decision variable $Z_{u,x} : Z_{u,x} = 1$ if node u is assigned to hub node x; $Z_{u,x} = 0$ otherwise. For virtual network mapping, cost is closely related to the distance between each pair of hubs and the flow capacity being transferred between hubs. Thus, in our model of traffic matrix mapping problem, the primary objective of mapping is to minimize the total cost of mapping by mapping the virtual network nodes onto the substrate nodes that are as close together as possible. We express the object function as:

$$minimize \sum_{u\in N^v}\sum_{x\in N^s}\frac{1}{c(x)}\Delta_{u,x}Z_{u,x}(\chi O_u + \delta D_u) + \sum_{u\in N^v}\sum_{x\in N^s}\sum_{y\in N^s}\frac{1}{b(x,y)}\alpha\Delta_{x,y}Y^u_{x,y} \quad (7)$$

By minimizing $\Delta_{u,x}$ and $\Delta_{x,y}$, the objective function tries to map the virtual network path onto shortest available physical paths. Also, by selecting hubs that capacitated with larger processing capacities and larger residual bandwidths, the formulation can gain the purpose of balancing the overall CPU and bandwidth capacity of the substrate network. Therefore, the objective function ensures that a virtual network is allocated using the least possible amount of resources beyond its demanded resources and occupies portions of the physical network that are least loaded.

The object function is subject to the following constraints:

1) *Hub selection constraint:*

$$\sum_{u\in N^v, x\in N^s} Z_{u,x} = p \quad (8)$$

This constraint ensures that the number of hubs selected is strictly equal to the number of virtual nodes, i.e., only p physical nodes act as hubs while other nodes act as transit nodes for flows.

2) *Processing capacity constraint:*

$$\sum_{\forall u\in N^v,(x,y)\in N^s} Y^u_{x,y}\cdot R + c_i(u)\cdot Z_{u,x} \le c_i(x), \quad \forall u\in N^v, x\in N^s, y\in N^s \quad (9)$$

This constraint ensures that the substrate network node always have greater CPU capacity than the virtual node requirement, while the forwarding nodes along the path have residual processing capacity greater than the forwarding processing requirement. The notation *R* stands for processing capacity needed for forwarding network traffic. In this paper, *R* was used similar to existing literature [9].

3) *Bandwidth capacity constraints:*

$$\sum_{u\in N^v} Y^u_{y,x} \le b(y,x), \forall x,y\in N^s, \quad (10)$$

and

$$\sum_{u\in N^v} Y^u_{x,y} \le b(x,y), \forall x,y\in N^s \quad (11)$$

These constraints ensure that the flows on the substrate network on either directions has enough residual bandwidth.

4) *Flow conservation constraint:*

$$\sum_{y\in N^s} Y^u_{x,y} + D_u Z_{v,x} = \sum_{y\in N^s} Y^u_{y,x} + O_u Z_{u,x}, \quad \forall u\in N^v, x\in N^s \quad (12)$$

This constraint enforces flow conservation at the hubs, which is to ensure that all the flow at a origin virtual node is sent to the hub, whereas a destination virtual node receives only the amount as directed by the demand between the origin and destination. Combined with bandwidth capacity constraints, we ensure that a flow is split only when there is no physical edge that can support the flow capacity between the origin and the destination node pair.

5) *Domain constraints:*

$$Y^u_{x,y} \ge 0, \forall u\in N^v, x,y\in N^s, \quad (13)$$

and

$$Z_{u,x}\in\{0,1\}, \forall u\in N^v, x\in N^s \quad (14)$$

These two domain constraints ensure that the algorithm selects the substrate node that are mapped to, within the minimize internodal distance, and to edges such that the path lengths are minimized.

V. Performance Evaluation

We evaluate the performance of traffic-matrix-based VHub mapping (abbreviates as VHub-tm) on different synthetic and practical network scenarios. We compare the results with those of other virtual network mapping algorithms (vnmFlib and D-ViNe). Traditional topology-based virtual network requests are used as input for D-ViNe and vnmFlib and corresponding traffic-matrix-based virtual network requests are used for VHub-tm.

A. Experimental Setup

The performance of algorithms are evaluated within three sets of experiments. First, we provide an illustration case to show the benefit of traffic matrix based virtual network mapping using exhaustive method. Second, we use large-scale simulations to show the performance of traffic matrix based mapping within different scales of virtual network requests comparing with other existing mapping algorithms. In the third set of experiments, we show the performance comparison of algorithms using virtual requests that are got from real virtual network traces collected from Emulab.

1) The first experiment is a simple illustration case, where we use requests with five end-nodes. No internal topology was specified for the traffic-matrix-based request; and for topology-based requests, we generate *all possible topologies* with three internal nodes and link bandwidth requests that are consistent with shortest path routing of the traffic specified in the traffic matrix. For each of the request, we successively map the same request onto a 100-node substrate within a 100×100 grid until the first mapping failure appears.
2) For the second set of experiments, we use a large number of virtual network requests to evaluate the number of successful mappings and runtime for each of the algorithms. We set up 7 groups of experiments with edge node number from 5 to 11; for each group, we generate 1000 random topologies as input for D-ViNe and vnmFlib, and generated the corresponding traffic matrices as input for VHub-tm. We map the 7 groups of topologies and traffic matrices to one 100-node-substrate-network within a 100×100 grid.
3) The third set of experiments uses real virtual network requests collected from Emulab. These traces do not vary as widely in their parameter configuration, but are more representative of current virtual network use. Since the resources in Emulab are heterogeneous and our implementation of algorithms cannot be directly tested on them, we set the node processing resources and edge bandwidth resources uniformly to 100 units each and randomly placed them on a 100×100 grid. The size of topologies is restricted to 400 nodes and 600 edges. The virtual networks are restricted to topologies of up to 15 nodes. We randomly pick 1000 virtual network requests from the collection and convert the topologies to traffic matrices as input for VHub-tm.

B. Evaluation Results for Synthetic Requests

For the first experiment, Fig. 2 shows the number of virtual networks that can be mapped successfully and the processing time for all variations of the topologies based requests and for the unique traffic matrix. The vnmFlib algorithm successfully mapped 47 virtual requests (for best case), D-ViNe successfully mapped 26 virtual requests (for best case), and traffic-matrix-based VHub-tm successfully mapped 77 virtual request (which is 63% more than vnmFlib and 196% more than D-ViNe). Since there is no single topology that provides an ideal basis for mapping under all conditions, traffic-matrix-based mapping clearly outperforms topology-based mapping. The computation time for VHub-tm is slightly longer than the average run time for topology-based mapping, but the significant improvement in resource utilization can easily justify this additional cost.

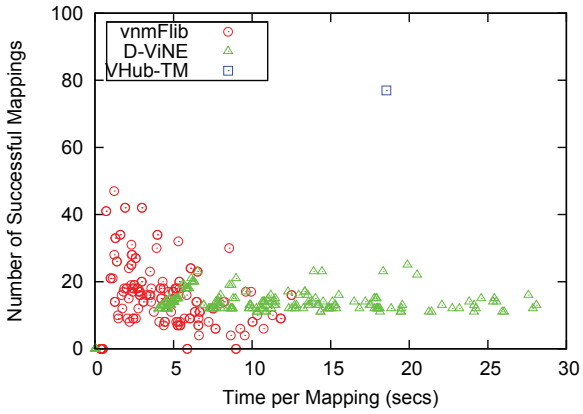

Fig. 2. Case 1 successful mapping vs. time

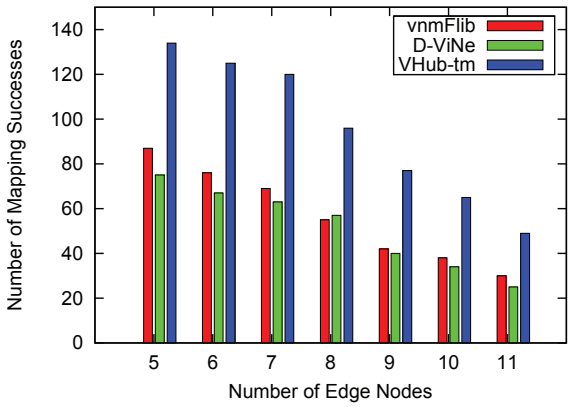

Fig. 3. Case 2 successful mapping comparison

For the second experiment, Fig. 3 and Fig. 4 show the number of successful mappings and runtime of the 7 groups of virtual network requests (with number of end nodes varying from 5 to 11), respectively. As can be observed, the VHub-tm approach spends almost the same time with the other two topology-based mapping algorithms, but can accommodate more virtual network requests then any other topology based algorithms.

C. Evaluation Results for Emulab Requests

For the set of real virtual network requests from Emulab, we show the number of successful mapping and revenue to cost ratio in Fig. 5 and Fig. 6. D-ViNe and vnmFlib can achieve roughly 350 successful mappings, but traffic-matrix-based formulation can achieve up to 459 successful mappings, which corresponds to 31% more virtual networks on the same

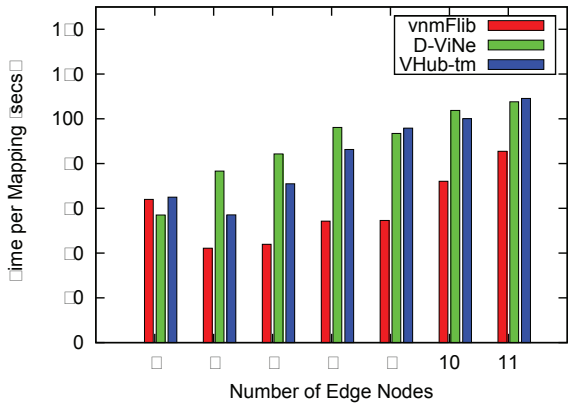

Fig. 4. Case 2 mapping time comparison

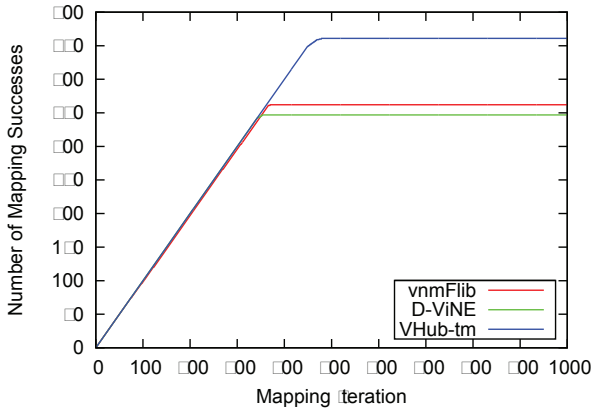

Fig. 5. Case 3 number of successful mappings

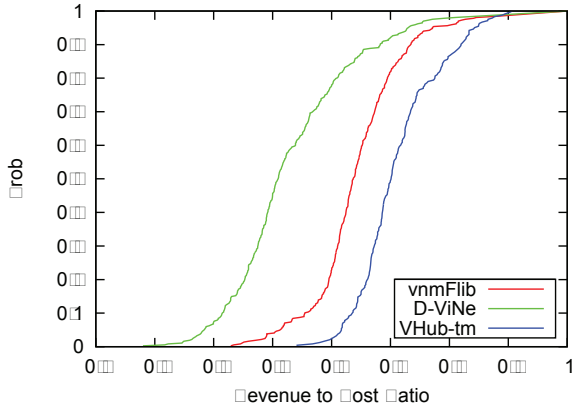

Fig. 6. Case 3 CDF of revenue to cost ratio

shared physical network infrastructure. In addition, since the internal node and link constraint are removed in the traffic matrix technique, the VHub-tm can gain higher revenue to cost ratio than other algorithms, which is beneficial for most of the ISPs.

VI. Conclusion

Network virtualization is a promising approach to share physical network resources among virtual networks with different protocols and functionality. Efficient mapping of virtual network requests to the physical network is an important and practical problem in this context. The representation of virtual network requests has previously been based on topologies. In this paper, we provide a reformulation of the virtual network mapping problem based on traffic matrices rather than complete network topologies. This approach and the mapping algorithm we present can significantly improve the effectiveness of virtual network mapping process. Our results show that the traffic-matrix-based VHub outperforms traditional topology-based algorithms and can successfully accommodate 63%-196% more virtual networks while requiring roughly the same amount of processing time. We believe that our work is an important step toward enabling the efficient operation of current and future virtualized networks.

References

[1] Anderson, T., Peterson, L., Shenker, S., and Turner, J. Overcoming the Internet impasse through virtualization. *Computer 38*, 4 (Apr. 2005), 34–41.

[2] Chowdhury, N., Rahman, M., and Boutaba, R. Virtual network embedding with coordinated node and link mapping. In *IEEE INFOCOM* (2009), pp. 5634–5640.

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

[4] Clark, D. D. The design philosophy of the DARPA Internet protocols. In *Proc. of ACM SIGCOMM 88* (Stanford, CA, Aug. 1988), pp. 106–114.

[5] Fang, Q., Cobb, J., and Leiss, E. A pre-selection routing scheme for virtual circuit networks. In *Proc. IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS* (2000).

[6] Gupta, A., Kleinberg, J., Kumar, A., Rastogi, R., and Yener, B. Provisioning a virtual private network: A network design problem for multicommodity flow. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing* (2001), pp. 389–398.

[7] Lischka, J., and Karl, H. A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures* (Aug 2009), ACM New York, NY, USA, pp. 81–88.

[8] National Science Foundation. *Global Environment for Network Innovation*. http://www.geni.net/.

[9] Ramaswamy, R., Weng, N., and Wolf, T. Analysis of network processing workloads. In *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (Austin, TX, Mar. 2005), pp. 226–235.

[10] Shanbhag, S. *On Data-Path Customization in Next-Generation Networks*. PhD thesis, University of Massachusetts Amherst, September 2011.

[11] White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., and Joglekar, A. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation* (Boston, MA, Dec. 2002), USENIX Association, pp. 255–270.

[12] Yu, M., Yi, Y., Rexford, J., and Chiang, M. Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM SIGCOMM CCR 38*, 2 (April 2008), 17–29.