VNMBench: A Benchmark for Virtual Network Mapping Algorithms

Jin Zhu and Tilman Wolf
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA
{jzhu,wolf}@ecs.umass.edu

***Abstract*—The network architecture of the current Internet cannot accommodate the deployment of novel network-layer protocols. To address this fundamental problem, network virtualization has been proposed, where a single physical infrastructure is shared among different virtual network slices. A key operational problem in network virtualization is the need to allocate physical node and link resources to virtual network requests. While several different virtual network mapping algorithms have been proposed in literature, it is difficult to compare their performance due to differences in the evaluation methods used. In this paper, we proposed VNMBench, a virtual network mapping benchmark that provides a set of standardized inputs and evaluation metrics. Using this benchmark, different algorithms can be evaluated and compared objectively. We present such an evaluation using three existing virtual network mapping algorithms. We compare the evaluation results of our synthetic benchmark with those of actual Emulab requests to show that VNMBench is sufficiently realistic. We believe this work provides an important foundation to quantitatively evaluating the performance of a critical component in the operation of virtual networks.**

***Index Terms*—future Internet, network virtualization, virtual network embedding, benchmark, evaluation**

I. Introduction

As the Internet has grown to a global communication infrastructure that connects large numbers of diverse distributed applications, new requirements for functionality and performance have emerged. These requirements include security (e.g., protection against address spoofing), diverse communication paradigms (e.g., content-addressable networks), and quality of service (e.g., performance guarantees for streaming media).

The current Internet architecture cannot accommodate these requirements since the Internet Protocol (IP), which is used by all systems in the Internet, cannot be changed without creating incompatibilities [1]. While some shortcomings of IP can be addressed by "middle-boxes" [2], such as firewalls [3] or network address translators [4], they do not provide a general solution that can adapt to fundamentally different network layer protocols, such as content-addressable networking [5].

An alternative network architecture that provides fundamental flexibility in the protocols that are deployed, including the network layer protocol, is network virtualization [6], [7]. In network virtualization, a single physical infrastructure is shared among multiple virtual network "slices" (similar to how operating system virtualization allows multiple different operating systems to share a single computer).

A key challenge in operating virtual networks is the problem of assigning logical nodes and links to physical resources. This "virtual network mapping problem" has received considerable attention in recent years [8]–[12] because it is a fundamentally hard problem and because improvements in the quality of mapping algorithm can significantly improve how many virtual networks can be accommodated in an infrastructure. Despite the attention focused on the virtual network mapping problem, it is still difficult to judge how well different algorithms perform under realistic operational conditions.

To address this problem, we propose a *benchmark for virtual network mapping algorithms* that can provide the basis for quantitative comparisons between competing algorithms. Important criteria for this benchmark are that the inputs (i.e., the virtual network substrate and the virtual network requests) are representative of what would be encountered in a real virtual network system. We discuss this issue in detail in this paper. We also show results of how the benchmark can be used to evaluate specific algorithms.

The specific contributions of this paper are:

- Design of a benchmark, VNMBench, for evaluating virtual network mapping algorithms. The benchmark consists of a variety of substrate network configurations and virtual network requests and a set of evaluation metrics that are meaningful in this context.
- Discussion of the representativeness of VNMBench (i.e., why the specific inputs used by the benchmark can be considered typical for a virtual network mapping environment).
- Evaluation of three existing virtual network mapping algorithms using VNMBench. The results demonstrate the types of quantitative results that can be obtained for determining the performance of any given virtual network mapping algorithm.
- Comparison of mapping results obtained from VNMBench with results obtained from real network virtualization environment (i.e., Emulab [13]). The comparison shows that the synthetic VNMBench requests lead to similar mapping results, thus showing that VNMBench generates a representative workload.

The remainder of the paper is organized as follows: Section II discusses related work. Section III present the benchmark, VNMBench. We present an evaluation of three algo-

rithms using VNMBench in Section IV and make a comparison between actual Emulab traces and our benchmark. Section V summarizes and concludes the paper.

II. Related Work

The concept of network virtualization was introduced in [6], [7]. Router systems that support network virtualization have been presented in [14]–[16]. These routers can isolate link resources and forwarding tables between slices. A mechanism for isolating protocol processing resources between slices is discussed in [17]. Network virtualization has been deployed in several research testbeds, including GENI [18] and Emulab [13].

The virtual network mapping problem requires the allocation of virtual network nodes and links to a physical substrate network. A survey of network virtualization and virtual network mapping algorithms is given in [19]. One of the first solutions developed by Ricci et al. uses simulated annealing to find mapping solutions [8]. Some algorithms assume that the substrate network resources are unlimited, such as Fan and Ammar [9]. To propose a solution for determining dynamic topology reconfiguration, Zhu et al. [10] calculate the whole mapping periodically.

Comparing evaluation results between published results is difficult because different evaluation environments are used. Yu et al. [11] use a substrate network with 100 nodes and 500 links in a 100×100 grid, virtual network requests nodes with uniform distribution between 2 and 10, and CPU and link data rates with uniform distribution between 0 to 100 units. However, Chowdhury et al. [12] use a 50-node substrate network in a 25×25 grid, CPU requirements uniformly distributed from 0 to 20, and bandwidth resources uniformly distributed between 50 and 100. These differences affect the results since parameters settings have an important effect on the mapping result. Some mapping algorithms are more suitable for small networks with a high node degree, some algorithms have a better performance in large and sparse networks. The benchmark we develop in this paper can help in providing the basis for fair and effective comparison of algorithms and to evaluate the advantages and disadvantages of different approaches.

III. VNMBench

The benchmark we develop in this work consists of two components: (1) input sets for virtual network mapping algorithms and (2) metrics for evaluating the outputs of virtual network mapping algorithms. The inputs consist of a variety of different scenarios that cover typical virtual network mapping uses. The metrics evaluate the performance of virtual network mapping results in terms of the quality of the mapping that was achieved and the running time. The following describes the general operation of the benchmark, the input sets, and performance metrics.

A. Operation of Benchmark

The benchmark provides parameters for generating two types of inputs: (1) one substrate network, which represents the physical infrastructure on which virtual networks are mapped, and (2) multiple virtual network requests, which need to be mapped onto the substrate network. Both links and nodes have constraints associated with them. Substrate links provide a limited amount of bandwidth; substrate nodes provide a limited amount of processing. Requests nodes are constrained in their location and require processing resources. Paths between mapped request nodes require network bandwidth [11]. The mapping algorithms determine the placement of request nodes and paths while avoiding resource conflicts in the substrate.

When multiple requests need to be mapped to a substrate, there are several different ways of handling mapping operation: multiple requests could be mapped one after another or at the same time; in case of incremental mappings, prior mappings can be considered fixed or can be changed (e.g., in order to make scarce resources available to later mapping requests); once multiple mappings have completed, additional mappings can be performed either until the substrate is full or prior mappings can be removed as they expire. Clearly, these differences in operation have a considerable impact on the results obtained from an evaluation of mapping algorithms.

In our work, we aim to stress-test the algorithms under considerations. Therefore, we chose (what we call *online*) operation, where virtual network requests are mapped successively onto a substrate without ever removing (or changing) a mapping that has been completed. With every successful mapping of a virtual network request, the remaining substrate resources become increasingly sparse. Therefore, finding mapping solutions becomes increasingly difficult. In case a virtual network request cannot be accommodated, it is considered a failed mapping attempt and the next virtual network request is processed. The mapping process continues for a specified number of requests (in our case 1,000). After all virtual network requests have been processed, the number of successful mappings and other metrics are determined.

It is important to note that VNMBench can be used for any type of operation since the input sets and (to some extend) the metrics are agnostic of the specific operation. Thus, VNMBench users can adapt the benchmark for their specific use. In this paper, however, we solely consider online operation.

B. Benchmark Inputs

In VNMBench, the inputs for the virtual network mapping algorithm are a substrate network and multiple virtual network requests. We model the substrate network as a weighted undirected graph denoted by $G^S = (N^S, E^S)$, where N^S is the set of the substrate nodes and E^S is the set of the substrate links. Each substrate node $n_i^S \in N^S$ has an associated processing capacity denoted by $c(n_i^S)$. Each substrate link $e_i^S(u, v) \in E^S$ between two substrate nodes u and v is associated with the bandwidth capacity value $b(e_i^S(u, v))$. We set $\lambda(u, v)$ to denote the length of $e_i^S(u, v)$ between two substrate nodes u and v.

A virtual network request is defined by a weighted undirected graph $G^V = (N^V, E^V)$, where N^V is the set of the virtual nodes and E^V is the set of the virtual links. The processing resource requested by a virtual network node $n_i^V \in N^V$ is denoted by $c(n_i^V)$, for each virtual link $e_i^V(u,v) \in E^V$ also has the bandwidth capacity $b(e_i^V(u,v))$.

1) Topology Generation: A key challenge in VNMBench is to determine how to generate representative topologies for the network substrate and the virtual network requests. In general, networks are difficult to model. The standard approach is to generate a graph such that the key metrics match with those of typical networks. The methods used for this kind of topology generation include regular topologies, flat random topologies, and hierarchical topologies.

The flat random method connects the set of nodes with a specific probability by edges. It can control the number of edges in the graph by changing the probability of connection of random nodes, but cannot control the configuration of the edges. It is more suitable to generate small graphs because it may have high node degree when generating large graphs [20], which is not typical for the Internet. Since the Internet can be viewed as a collection of interconnected routing domains [21], the trans-stub method generate large graphs efficiently with a realistic average node degree [20]. Therefore, we choose this method to generate our substrate network. For the edge method, we use the improved Waxman method [22] (rather than the pure random method or exponential method, which are more likely to have long edges).

To generate substrate and request topologies in VNMBench, we use the GT-ITM tool [23]. For the substrate network, we use the trans-stub topology generation method [20] and for virtual network requests, we use the Waxman method.

2) Substrate Network: To explore a range of network configurations, we consider two substrate network sizes: medium size with 100 nodes and large size with 500 nodes. Each substrate size is further separated by the distinction of link density: dense substrate network has approximate two times the number of links than the sparse substrate network.

We choose to use these two different sizes of the substrate network in our benchmark, because in real networks there are medium size networks (e.g., corporate or campus network) and large size networks (e.g., national network). We set the two sizes to differ by a factor of 5 in the number of nodes. In VNMBench, the medium substrate network's number of nodes is 100 and node placement is random within a 100×100 grid. In the large substrate network, there are 500 nodes and placed within a (200×200) grid.

The density of the links in the network plays an important role in the virtual network requests mapping, because it is easier for mapping algorithms to find suitable substrate nodes and links in the dense style, but more difficult to solve the mapping problems when the network is sparse. In VNMBench, the sparse substrate network has 129 links and the dense substrate network has 255 links in the medium size, and 639 links and 1294 links, respectively, in the large size.

In order to create a realistic network, we fix the node degree of sparse substrate network around 2.5 and dense substrate network around 5.1, which are close to the realistic average node degrees 6 [20]. When the number of nodes is 100 in sparse style, the graph has 2 transit domains, each transit domain has an average of 2 nodes, 4 stub domains per transit node, and no extra transit-stub or stub-stub edges. Each stub domain also has 6 nodes on average. According to the formula of the average node degree of the transit stub graph, we can calculate the node degree of two kinds substrate networks:

$$ND = \frac{2(E_t + (1 + E_s N_s)K)}{1 + KN_s}, \tag{1}$$

where the E_t and E_s are the edge densities of the transit and stub domains, respectively. N_s is the average nodes per stub domain, K is the average stub domains per transit node. The average node degree is approximately 2.58. In the dense style, there are 50 extra transit-stub links or stub-stub links and node degree is approximately 5.10.

TABLE I: Parameters of Substrate Network

	Medium size		Large size	
	Sparse	Dense	Sparse	Dense
Number of nodes	100	100	500	500
Number of links	129	255	639	1294
Node degree	2.58	5.10	2.55	5.17
Scale	100×100	100×100	200×200	200×200
Transit domains	2	2	2	2
Nodes per transit	2	2	5	5
Stubs per transit node	4	4	7	7
Nodes per stub	6	6	7	7
Extra links	0	50	0	250

3) Virtual Network Requests: We separate the virtual network requests into three size: 5 nodes, 10 nodes and 20 nodes. The topologies for these requests are generated using the Waxman method. In the Waxman model, the probability of an edge connecting a pair of nodes u and v is based on the Euclidean distance between them:

$$P(u,v) = \alpha e^{-\frac{d(u,v)}{\beta L}}, \tag{2}$$

where $d(u,v)$ is the distance from node u to node v, $\alpha > 0$, $\beta \leq 1$, and L is the maximum distance between any two edges. Parameter α determines the number of edges in the graph, and parameter β determines the ratio of long edges to short edges [22]. In our benchmark, we set α is in the range from 0.3 to 0.8 and β from 0.15 to 0.25. Thus, for every virtual network request, the node degree is in the range of 2.5 to 5, which is typical of actual networks.

4) Parameter Settings: The VNMBench parameters are summarized Tables I and II. The processing resource of all requests are uniformly distributed (between 0 and 20) and link bandwidth requests are also uniformly distributed (between 0 and 50). A total of 1,000 virtual network requests are generated for each run of the benchmark.

C. Benchmark Metrics

In VNMBench, we consider three metrics to compare and evaluate mapping algorithm performance: number of successful mappings, revenue-to-cost ratio, and running time. In the

TABLE II: Parameters of Virtual Network Request

	Small	Medium	Large
Number of nodes	5	10	20
Number of links	[4,6]	[9,15]	[19,47]
Edge method	Waxman		
Processing resources	uniform distribution [0,20]		
Bandwidth resources	uniform distribution [0,50]		
Range of α	[0.3,0.8]		
Range of β	[0.15,0.25]		

Section IV, we compare three specific mapping algorithms with respect to these metrics to show that meaningful inferences about algorithm performance are possible.

1) Number of Successful Mappings: One of the main goals of the network virtualization is enable many virtual networks to coexist together on a shared substrate network. Even though each virtual network request has a different topology, different processing resource requirements, and different bandwidth requirements, a mapping algorithm should try to maximize the number of coexisting virtual network requests [24]. Therefore, we choose the total number of successful mappings as one of the key metrics for VNMBench.

Note that this metric assumes online operation as discussed above. Also, it is assumed that there are enough requests to completely fill up the substrate network. In case a request cannot be mapped to the substrate, it is considered a mapping failure. In this case, it can be distinguished if the mapping failed due to the lack of node resources or link resources. This distinction may help in understanding why the performance of a particular substrate network (or a particular mapping algorithm) is limited.

2) Revenue-to-Cost Ratio: Another important metric is to measure the amount of substrate resources are necessary to accommodate a request. In virtual network mapping, it is common to refer to the complexity of a request as "revenue" (i.e., how much a customer would pay for having their network installed in a substrate). The amount of substrate resources necessary are referred to as "cost" of a mapping.

We let $G_j^V = (N_j^V, E_j^V)$ denote the j^{th} virtual network request, and define the revenue, cost and the ratio of them with this request. The cost of mapping the j^{th} request is the total cost of allocation of substrate network resources to the j^{th} virtual network, defined as $C(G_j^V)$. We consider node resources and link resources, so the total cost can be separated into two parts: the node mapping cost and the link mapping cost. We use α (node mapping cost factor) and β (link mapping cost factor) as tunable weights that are set based on substrate network provider policies. We set $c(e_j^V)$ as the demand of links and $c(n_j^V)$ as the demand of nodes in the j^{th} virtual network. Thus, the total mapping cost is defined as

$$\begin{aligned} C(G_j^V) =& \alpha \cdot \sum_{n_j^V \in N_j^V} c(n_j^V) + \\ & \beta \cdot \sum_{e_j^V(u,v) \in E_j^V} \lambda(f_E(u,v)) \cdot c(e_j^V), \end{aligned} \tag{3}$$

where $\lambda(f_E(u,v))$ is the length the set of substrate links in G^S representing the path that the virtual link (u,v) is mapped to. In our work, we set $\alpha = \beta = 1$.

We define the revenue $R(G_j^V)$ generated by the j^{th} virtual network request as

$$R(G_j^V) = \sum_{n_j^V \in N_j^V} c(n_j^V) + \sum_{e_j^V(u,v) \in E_j^V} c(e_j^V) \tag{4}$$

The revenue-to-cost ratio is then based on the above definitions:

$$R/C - Ratio = \frac{R(G_j^V)}{C(G_j^V)} \tag{5}$$

the $R/C - Ratio$ has values between 0 and 1. It is set to 0 when the virtual network requests mapping fails and achieves 1 when an optimal virtual network mapping is found, which means every virtual link uses a direct physical link in the substrate.

3) Running Time: The definition of running time is the average time of processing one virtual network request by the mapping algorithm. The running time of a mapping algorithm is of practical importance – especially since virtual network requests are often handled dynamically in an online fashion. Therefore, the running time of an algorithm is also a metric in our benchmark. We determine the average time by measuring the total running time for all 1,000 requests (including failed mappings) and dividing accordingly.

IV. Comparison of Mapping Algorithms Using VNMBench

To demonstrate how VNMBench can be used in practice, we evaluate three mapping algorithms: GMCF [11], vnmFlib [25] and Dvine [12]. These algorithms are typical representatives for virtual network mapping algorithms. Thus, showing that VNMBench can be applied to all of them to provide comparative results illustrates the usefulness of our virtual network mapping benchmark.

All algorithms are set to the best possible parameters as mentioned by the authors and are presented with the input data generated by VNMBench. All algorithms can use path-splitting and thus can map a virtual link onto multiple physical paths. A virtual node can be mapped to any physical node in the substrate. The substrate resources can be mapped at an arbitrarily fine granularity to virtual networks. A total of 1,000 requests are processed by each algorithm.

For solving the Dvine, GMCF and vnmFlib formulations, we have used GLPSOL Optimization Studio. These experiments were run on and Intel Quad-core CPU running at 2.50 GHz with 3MB of level-2 cache and 3.4G of main memory.

A. Successful Mapping and Revenue-to-Cost Ratio

Figures 1 and 2 show number of successful mappings and revenue-to-cost ratio for sparse and dense substrate networks of medium size. The x-axis is the request mapping iteration from 0 to 1,000 and the y-axis is the number of successful mappings or the revenue-to-cost ratio up to that iteration. Each subgraph is shown for virtual network requests of 5 nodes, 10 nodes, and 20 nodes.

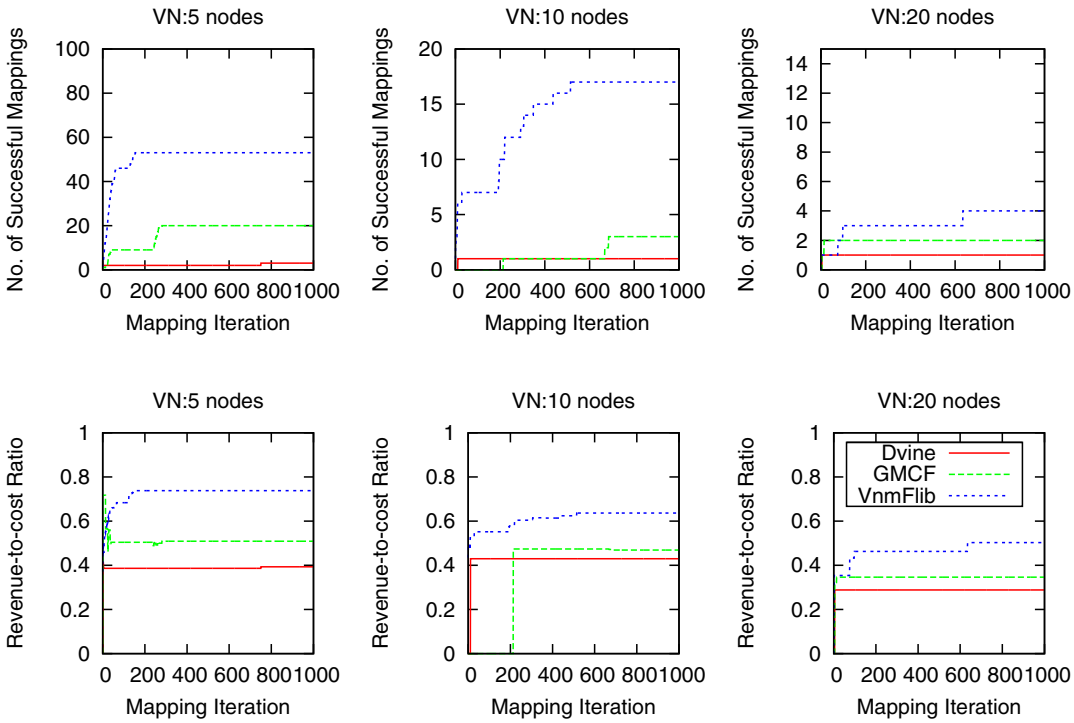

Fig. 1: Comparison of Amount of Successful Mapping and Revenue-to-Cost Ratio (Sparse Substrate Network with 100 Nodes).

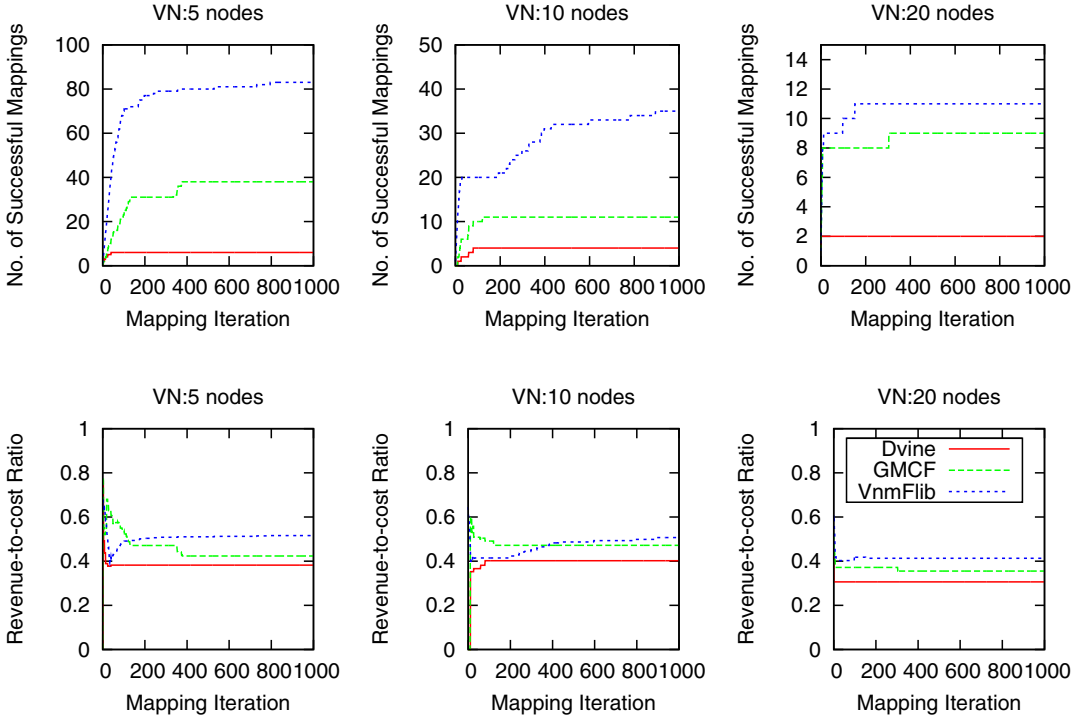

Fig. 2: Comparison of Amount of Successful Mapping and Revenue-to-Cost Ratio (Dense Substrate Network with 100 Nodes).

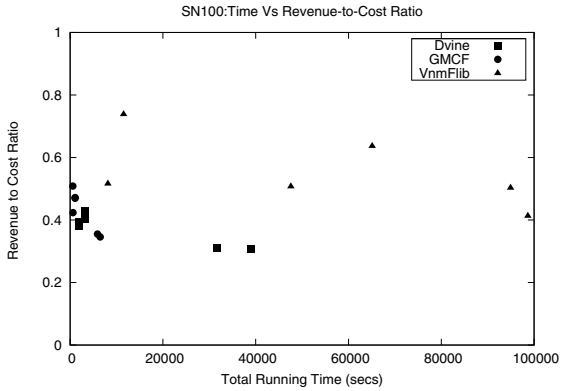

Fig. 3: Comparison of Time and Revenue-to-Cost Ratio (Substrate Network with 100 Nodes).

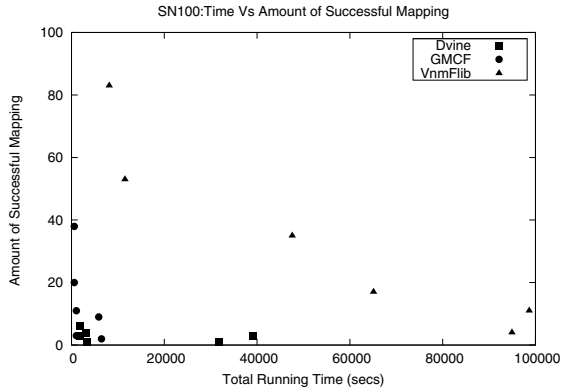

Fig. 4: Comparison of Time and Number of Successful Mapping (Substrate Network with 100 Nodes).

From these results, we can make the following observations:

- We find that the vnmFlib algorithm has the largest number of successful mappings and the highest revenue-to-cost ratio among all three algorithms. For the dense substrate network, the successful mappings of vnmFlib are approximately 3 times more than GMCF and 15 times than Dvine; For the sparse substrate network, the amount of successful mappings of vnmFlib are around 4 times than GMCF and 22 times than Dvine. The revenue-to-cost ratio is almost 1.5 times higher.
- In both the sparse and dense substrate, GMCF can satisfy more virtual network requests and has a higher revenue-to-cost ratio than Dvine. But, with an increasing number of successful mappings, the revenue-to-cost ratio of Dvine and vnmFlib also increase, but GMCF improves the number of successful mappings at the expense of the revenue-to-cost ratio. In detail, we could find GMCF and Dvine have the same failed node mapping amount, but Dvine has a higher failed edge mapping amount than GMCF.
- For a substrate network with 500 nodes, Dvine fails entirely for requests with 10 and 20 nodes, and only maps 17 requests with 5 nodes. Because the dense substrate network style has twice the node degree of the sparse style, there are more possibilities to map edges successfully for all algorithms.

Overall, for the medium size substrate network, the vnmFlib algorithm is the best choice used, but it takes a lot of time (see below). If the emphasis is on the amount of successful mapping, GMCF may be preferable. If the emphasis is on revenue-to-cost ratio, then Dvine may be preferable.

Similar results can be obtained for large substrate networks (not shown due to space constraints).

B. Running Time

Figure 3 and Figure 4 show the distribution of number of successful mappings and revenue-to-cost ratio with respect to the running time. We observe that vnmFlib has the highest number of successful mappings and revenue-to-cost ratio but also a longer running time. GMCF has the shortest running time. Dvine performs the worst in the three mapping algorithms since it takes more time than GMCF algorithm with a lower successful mapping amount.

Thus, one can choose to use the mapping algorithm with small running time and corresponding higher successful mapping performance (such as GMCF) if one emphasizes on the time to complete the mapping process. If one considers the amount of successful mapping of virtual network requests as the only requirement, one can choose the algorithm with the best performance in mapping (such as vnmFlib). In addition, Dvine and vnmFlib are both a good choice if one aims to increase the revenue-to-cost ratio with increasing successful mappings.

Clearly, the evaluation of these algorithms using VNM-Bench provides interesting insights into the operation of and tradeoffs between algorithms. (We are less focused on which particular algorithm outperforms another.)

C. Comparison of VNMBench Results and Emulab Results

A key requirement of any benchmark is that it generates a representative workload. To demonstrate that VNMBench generates a realistic set of virtual network mapping requests, we compare these requests to requests that were issued by real users in Emulab. Since it is difficult to compare the actual requests (due to topology, parameters, etc.), we compare the mapping results obtained from both scenarios. If the mapping results are similar in both cases, it can be assumed that the synthetic requests issued by VNMBench are sufficiently representative of what happens in reality in Emulab. (Of course, the benefit of a synthetic benchmark, like VNMBench, is that it can be scaled to other configurations, which is not possible with traces.)

The Emulab request traces have topologies with 2 to 15 nodes. Since the traces do not have explicit processing demands, we added them based on a uniform distribution between 0 and 20. Edge bandwidth demands are uniformly

distributed between 0 and 50. For comparison, we use the same substrate network generated in VNMBench model with node processing resources and edge bandwidth resources set to 100 units on a 100×100 grid.

We separate the experiments into two parts. First, we extract 1,000 requests with 5 nodes from the Emulab collected requests and make a comparison with 5-node virtual network requests in VNMBench model. Figure 5 shows the number of successful mappings and revenue-to-cost ratio of three mapping algorithms based on Emulab requests and VNMBench requests in the sparse substrate network with 100 nodes. From the figure, we observe the successful mapping amount of Dvine and GMCF are equal in both scenarios and vnmFlib can successfully map 48 Emulab requests and 52 VNMBench requests. For the revenue-to-cost ratio, the value and variation trend of the three mapping algorithms is also similar for Emulab requests and VNMBench requests. For both scenarios, vnmFlib performs better than the other two algorithms in these two metrics. GMCF has more successful mappings than Dvine, but with a decrease of revenue-to-cost ratio. Thus, the results for VNMBench are extremely similar to those for real Emulab requests.

When using Emulab requests with varying numbers of nodes, the comparison to VNMBench becomes more difficult. Since we offer many topologies for mapping, the algorithms will successfully map many of the smaller Emulab topologies. In contrast, VNMBench uses the same size topologies and thus does not offer "easy" mappings. In the second part, we extract 1,000 requests randomly from Emulab virtual network requests and also using 5-node requests in VNMBench model to make a comparison in the evaluation results of different mapping algorithms. The probability of node number between 2 to 5 is around 0.6 in Emulab requests, the average node number in the successful mapping requests is around 3 and average link number is around 2 when using GMCF and vnmFlib mapping algorithms. The smaller node and link number, the easier for virtual network requests mapping successfully. Therefore, the number of successful mappings is 118 for vnmFlib algorithm based on Emulab requests, which is larger than the number of mappings from VNMBench requests. For Dvine, the average node number of successful mapping is 4 and link number is 3, which is similar to the VNMBench model (number of nodes is 5 and average number of links is 4). Thus, Dvine has the same amount of successful mapping. Despite these differences, there are still many similarities in the results, especially when considering the revenue-to-cost ratio.

Thus, these comparisons show that VNMBench does generate inputs that are representative of virtual network mapping requests generated in real virtualized networks.

V. Summary and Conclusion

Network virtualization is a way of sharing a physical network among virtual networks that can differ in functionality. The mapping of virtual network requests to physical resources is an important aspect of the operation of a virtual network system. Various mapping algorithms have been developed in literature, but a detailed quantitative understanding of their performance has been difficult due to differences in their performance evaluation. In this paper, we introduce VNMBench, a benchmark for evaluating virtual network mapping algorithms that allows thorough evaluation and comparison of these algorithms. We discuss the design of the benchmark, its inputs, and its performance metrics. We evaluate three different algorithms using VNMBench and show that detailed performance results and comparisons between algorithms can be achieved. We also show that the results obtained when using VNMBench are comparable to those from real Emulab requests. We believe that this work presents an important step toward finding faster and more effective virtual network mapping algorithms in the future.

References

[1] D. D. Clark, "The design philosophy of the DARPA Internet protocols," in *Proc. of ACM SIGCOMM 88*, Stanford, CA, Aug. 1988, pp. 106–114.

[2] S. Guha and P. Francis, "An end-middle-end approach to connection establishment," in *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, Kyoto, Japan, Aug. 2007, pp. 193–204.

[3] J. C. Mogul, "Simple and flexible datagram access controls for UNIX-based gateways," in *USENIX Conference Proceedings*, Baltimore, MD, Jun. 1989, pp. 203–221.

[4] K. B. Egevang and P. Francis, "The IP network address translator (NAT)," Network Working Group, RFC 1631, May 1994.

[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT)*, Rome, Italy, Dec. 2009, pp. 1–12.

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

[7] J. S. Turner and D. E. Taylor, "Diversifying the Internet," in *Proc. of IEEE Global Communications Conference (GLOBECOM)*, vol. 2, Saint Louis, MO, Nov. 2005.

[8] R. Ricci, C. Alfeld, and J. Lepreau, "A solver for the network testbed mapping problem," *SIGCOMM Computer Communication Review*, vol. 33, pp. 65–81, Apr. 2003.

[9] J. Fan and M. H. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies," in *In Proc. IEEE INFOCOM*, 2006.

[10] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. of the Twentyfifth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2006)*, Barcelona, Spain, Apr. 2006.

[11] 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, pp. 17–29, March 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355737

[12] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping." in *INFOCOM'09*, 2009, pp. 783–791.

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

[14] Y. Liao, D. Yin, and L. Gao, "PdP: parallelizing data plane in virtual network substrate," in *Proc. of the First ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures (VISA)*, Barcelona, Spain, Aug. 2009, pp. 9–18.

[15] M. B. Anwer, M. Motiwala, M. b. Tariq, and N. Feamster, "SwitchBlade: a platform for rapid deployment of network protocols on programmable hardware," in *Proceedings of the Conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, New Delhi, India, Sep. 2010, pp. 183–194.

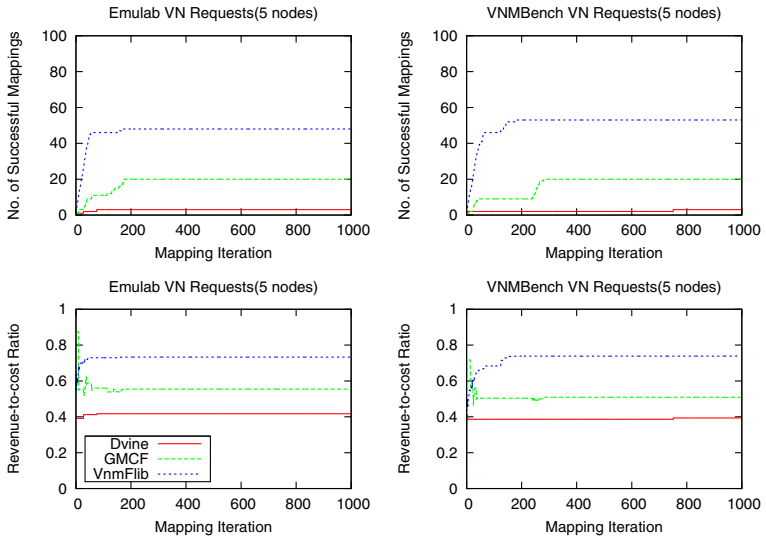

Fig. 5: Result Comparison between Emulab requests and VNMBench (Sparse Substrate Network with 100 Nodes).

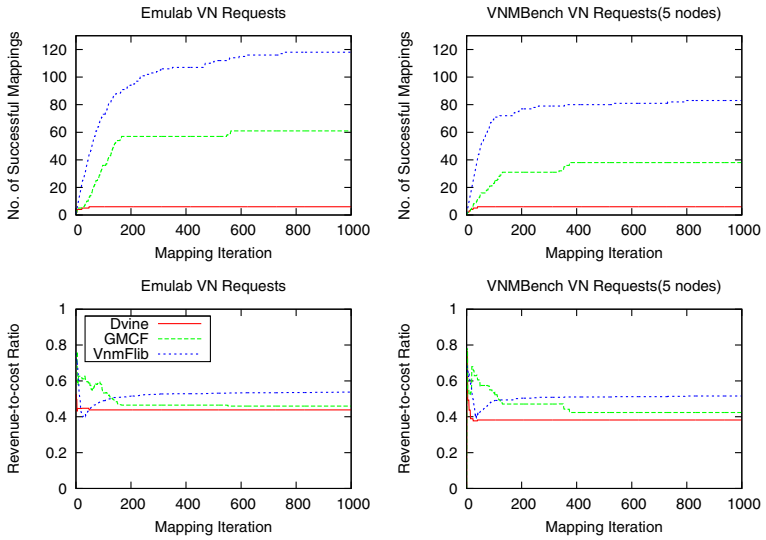

Fig. 6: Result Comparison between Emulab requests and VNMBench (Dense Substrate Network with 100 Nodes).

[16] D. Yin, D. Unnikrishnan, Y. Liao, L. Gao, and R. Tessier, "Customizing virtual networks with partial fpga reconfiguration," *SIGCOMM Computer Communication Review*, vol. 41, pp. 125–132, Jan. 2011.

[17] Q. Wu, S. Shanbhag, and T. Wolf, "Fair multithreading on packet processors for scalable network virtualization," in *Proc. of ACM/IEEE Symposium on Architectures for Networking and Communication Systems (ANCS)*, San Diego, CA, Oct. 2010.

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

[19] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, Apr. 2010.

[20] E. W. Zegura, K. L. Calvert, and M. J. Donahoo, "A quantitative comparison of graph-based models for internet topology," *IEEE/ACM Trans. Netw.*, vol. 5, pp. 770–783, December 1997. [Online]. Available: http://dx.doi.org/10.1109/90.650138

[21] D. Clark, "Policy routing in internet protocols," M.I.T. Laboratory for Computer Science, Tech. Rep. RFC1102, 1989.

[22] B. M. Waxman, *Routing of multipoint connections*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1991, pp. 347–352. [Online]. Available: http://dl.acm.org/citation.cfm?id=128960.128991

[23] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of the Fifteenth annual joint conference of the IEEE computer and communications societies conference on The conference on computer communications - Volume 2*, ser. INFOCOM'96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 594–602. [Online]. Available: http://dl.acm.org/citation.cfm?id=1895868.1895900

[24] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, pp. 862–876, April 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2009.10.017

[25] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. of the First ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures (VISA)*, Barcelona, Spain, Aug. 2009, pp. 81–88.