Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 358647, 17 pages
doi:10.1155/2012/358647

Research Article

Virtual Network Embedding: A Hybrid Vertex Mapping Solution for Dynamic Resource Allocation

Adil Razzaq, Markus Hidell, and Peter Sjödin

School of ICT, KTH Royal Institute of Technology, 16440 Kista, Sweden

Correspondence should be addressed to Adil Razzaq, arazzaq@kth.se

Received 2 March 2012; Revised 14 May 2012; Accepted 16 May 2012

Academic Editor: Shuo Guo

Virtual network embedding (VNE) is a key area in network virtualization, and the overall purpose of VNE is to map virtual networks onto an underlying physical network referred to as a substrate. Typically, the virtual networks have certain demands, such as resource requirements, that need to be satisfied by the mapping process. A virtual network (VN) can be described in terms of vertices (nodes) and edges (links) with certain resource requirements, and, to embed a VN, substrate resources are assigned to these vertices and edges. Substrate networks have finite resources and utilizing them efficiently is an important objective for a VNE method. This paper analyzes two existing vertex mapping approaches—one which only considers if enough node resources are available for the current VN mapping and one which considers to what degree a node already is utilized by existing VN embeddings before doing the vertex mapping. The paper also proposes a new vertex mapping approach which minimizes complete exhaustion of substrate nodes while still providing good overall resource utilization. Experimental results are presented to show under what circumstances the proposed vertex mapping approach can provide superior VN embedding properties compared to the other approaches.

1. Introduction

Internet is being utilized to provide a wide range of services. Over a period of time (which is not too long), it has become vital component/core architecture to provide services for global commerce, media, and defense [1].

In spite of the success attributed with the current Internet, it has some flaws which need to be addressed. The "everything over IP" [2], as well as "best-effort" packet delivery does not suit all the services being provided on current Internet, whereas security, routing stability, and control and QoS (quality of service) guarantees are also some of the major concerns [1].

However, there are many limitations/obstacles in overcoming the above mentioned flaws. Some of these include appropriate changes in routers and host software, as well as joint agreement of all the ISPs on any architectural change [3]. Capital investment, competing interests of stakeholders as well as end-to-end design of IP, calls for a worldwide agreement to introduce any changes [1]. Since, it is very rare that a single ISP controls complete end-to-end path, new services have only been employed/tested within small geographic locations [4].

The challenges/requirements to overcome the Internet impasse/ossification were outlined in [3–5]. Requirements mentioned in [3] include, ease of experimentation with new architectures on live traffic, provisioning of a plausible deployment path for an architecture, and focusing of an architectural solution on a broad range of problems. The challenges described in [4] are, discovering the resources of a physical infrastructure, assigning virtual networks to underlying physical networks, and accounting of resources. Isolation, performance, scalability, flexibility, evolvability, management, and applications were the challenges identified for new generation network architectures (future network) in [5].

Network virtualization is at the heart of proposals for addressing the Internet ossification [1, 3, 4]. It can be utilized in experimental research facilities [6–8] as well as in

provision of customized end-to-end services over a shared infrastructure [1, 4].

A primary feature of the future Internet would be to assign substrate network resources to the requested virtual networks. Therefore, virtual network embedding (VNE) is the key area in network virtualization. In order to embed/assign/map a virtual network onto the substrate/physical network, each virtual node is mapped to a physical node and each virtual link is embedded on a substrate path. A number of virtual networks (VNs) can be deployed on top of the physical network (or *substrate*), depending on the capability of the substrate and the demands of VNs.

Virtual network embedding (VNE) problem is NP-hard [9, 10] where several constraints need to be satisfied. In order to map a VN onto the substrate, requirements of both its vertices as well as edges should be fulfilled. In addition to this, VNs can arrive at different times, in any order and can be based on any standard network topology (e.g., star, bus, ring, or mesh). The substrate network also has a limited amount of resources. Thus, we need to embed or map a VN with resource constraints onto the substrate network (SN) which has finite resources.

In this paper, we evaluate three different vertex mapping approaches for VNE. Our first approach deals with mapping virtual vertices onto any available substrate nodes which can satisfy their demand. This method does not take into account the possibility of a node becoming bottleneck at the time of mapping a VN's vertex, is named as baseline approach (BLA), and was presented in [11]. Second approach is focused on mapping virtual vertices to the substrate nodes with maximum resources, is called as greedy node mapping (GNM), and was presented in [9]. The advantage of using GNM is that it can minimize the use of substrate resources from bottleneck nodes. Drawback of GNM can be that vertices may get mapped in such a way that more bandwidth resources may be needed to map a VN as compared to BLA. Third approach is being proposed in this paper and is named as HBNRM (hybrid BLA bottleneck node reduced mapping). Main focus of this new approach is to utilize benefits of both BLA and GNM while minimizing their disadvantages.

Main contributions of this paper are to evaluate different vertex mapping approaches and investigate their impact on VN embedding, how substrate's nodes become bottleneck and get exhausted. Resource utilization as a result of mapping vertices at different substrate locations by three vertex mapping approaches is also analyzed. In order to thoroughly investigate the impact of vertex mapping by any approach, evaluations are done by mapping sparsely and densely connected VNs on sparsely as well as densely connected substrate networks. The proposed solution starts with the hypothesis that it should be possible to avoid complete exhaustion of node resources at a lesser cost, which can improve VN embedding possibilities.

Rest of the paper is organized as follows. Section 2 defines the problem while Section 3 presents work done in the area of VNE. Section 4 describes our solution whereas, Section 5 presents simulation results. Section 6 concludes the paper.

2. Network Model and Problem Description

The proposed solution represents virtual as well as substrate networks as undirected graphs. The substrate network is represented by $S = (N, A^N, L, A^L)$, whereas the network to be mapped; that is, the VN is shown by $M = (V, D^V, E, D^E)$. Notations for describing the VN mapping problem are summarized in Table 1.

Throughout this document when a reference is made to a link or node, it means that it belongs to the substrate, while VN's link and node will be termed as edge and vertex, respectively. We consider central processing unit (CPU) as a resource for nodes and vertices, and bandwidth to be the resource for edges and links.

Figure 1 shows a substrate network whereas Figure 2 represents a VN request. Notation for describing node and link capacities is similar to the one proposed in [9].

VN will only be mapped on the substrate if requirements of each of its vertex as well as edge are satisfied. After mapping vertices onto the nodes which satisfy vertex demand, paths need to be calculated for each pair of nodes in the VN. Then link resources in the path are compared with the edge demand. At this point, if the path satisfies edge request, then VN is completely mapped. After satisfying requests of vertices and edges of a VN, a residual graph (S_{res}) is obtained which contains remaining capacities of nodes and links of the substrate [12].

In the beginning of VN embedding process, we initialize residual capacities of nodes and links with the following actual capacities:

$$\begin{aligned} R^N &= A^N, \\ R^L &= A^L. \end{aligned} \tag{1}$$

Therefore, when a node or link is mapping a vertex or an edge for the first time then, its residual capacity is equal to its original capacity ($r^n = a^n \wedge r^l = a^l$) and the vertex or edge demand (d^v or d^e) is matched with it. After initial mapping of a vertex or edge is made, the new residual capacity (r^n) of a node ($n \in N$) is obtained by subtracting vertex demand (d^v) from the node resource, whereas remaining capacity (r^l) of a link ($l \in L$) is found by deducting edge request (d^e) from the link resource:

$$r^n \longleftarrow r^n - d^v, \tag{2a}$$

$$r^l \longleftarrow r^l - d^e. \tag{2b}$$

Resources need to be returned to the substrate if, after mapping initial vertices or edges of a VN, there comes a point when requirements of a certain edge or vertex cannot be satisfied. This means that in such a scenario the initial or base graph (S_{base}) is regenerated:

$$r^n \longleftarrow r^n + d^v, \tag{3a}$$

$$r^l \longleftarrow r^l + d^e. \tag{3b}$$

Table 1: Notations of VNE problem.

S	Substrate network
N	Set of nodes belonging to the substrate network
A^N	Attribute associated with substrate nodes
L	Set of links joining two nodes ($n_i, n_j \in N$), of the substrate network
A^L	Attribute associated with substrate links
$P^S_{m,n}$	Substrate path from source node m, to destination node n
M	Virtual network
V	Set of vertices of virtual network
D^V	Demand associated with vertices
E	Set of edges connecting two vertices ($v_s, v_t \in V$), of the virtual network
D^E	Demand associated with edges

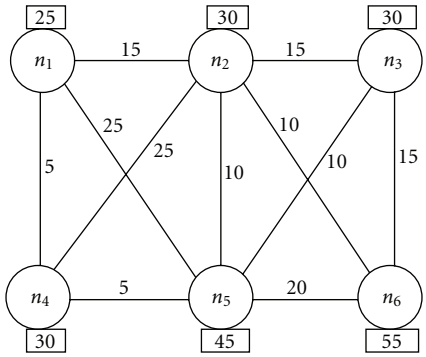

Figure 1: Substrate network.

We define the cost of mapping a VN, as sum of overall substrate resources assigned to its vertices and edges, in the same way, as previously presented in [13]. Our cost function is similar to the one given in [13]:

$$C(M) = \sum_{e \in E} \sum_{l \in L} \left(b_e^l\right) + \sum_{v \in V} (d^v). \tag{4}$$

A vertex will only be mapped on a single node whereas an edge can be mapped on a substrate path ($P^S_{m,n}$) containing one or more than one links. The term (b_e^l) in (4) indicates bandwidth allocated to an edge ($e \in E$) from a substrate link ($l \in L$).

3. Literature Review

This section is divided into two parts. It starts with a description of constraints associated with VNE, while the second part describes and categorizes work done in this area.

3.1. Constraints Associated with VNE. The virtual network embedding problem is NP-hard [9, 10] where several constraints need to be satisfied. In addition to vertex and edge constraints, the VNs can arrive at different times and in any order whereas substrate resources are also finite. Therefore, before going through the details of various solutions to the VNE problem, it is necessary to first have a look at the constraints associated with it.

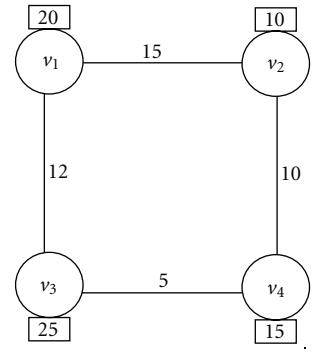

Figure 2: Virtual network.

3.1.1. Node Constraints. Two types of node constraints may be associated with a VN request.

(i) Capacity. A VN request may be constrained by a certain amount of resources on its nodes. Nodes of a VN may require a fixed number of processing or memory resources, for example, in order to run an experiment, 500 MHZ of CPU may be required for each virtual node of the VN [9].

(ii) Location. In addition to the capacity, placement of VN's nodes may be required in certain locations. This constraint may be imposed if, VN's nodes are part of a service which requires this feature, for example, CDN (Content Distribution Network), gaming service.

3.1.2. Link Constraints. Two types of link constraints may be associated with a VN request.

(i) Bandwidth. In order to run an experiment or provide a service, a VN may require certain amount of bandwidth on each of its links [9].

(ii) Link Propagation Delay. In addition to the demand of bandwidth on its links a VN may also be constrained by link propagation delay, for example, a VN carrying delay sensitive traffic to provide a service such as QoS (Quality of Service) [14].

3.1.3. Admission Control. The substrate network has finite resources on its nodes and links. Admission control process needs to be implemented for two reasons.

(i) It ensures that demands of newly arrived VNs can be fulfilled by the substrate.

(ii) Resource allocation made to already mapped VNs is not violated.

Therefore, VN requests may be rejected or postponed if the substrate does not have sufficient resources to satisfy demands of a VN at the time of arrival [9].

3.2. Virtual Network Embedding Approaches. After going through challenges associated with the VNE problem, we can now have a look at how they have been taken care of by various solutions. Details of such solutions will be given first; later on Table 2 also depicts this process. Description of each of these types of solutions is presented below.

3.2.1. Constraints. We consider node capacity, link bandwidth, and admission control (defined above) as basic constraints to the VNE problem. Some solutions to the VNE problem handle all basic constraints while others only provide solution to a subset of these constraints.

The solutions presented in [18, 20] do not perform admission control. Node capacity and admission control constraint are not considered in [16, 17]. Assuming that vertex mapping is known in advance, the authors have only provided solution for edge mapping in [19], that is, they have not taken care of node capacity constraint.

The approaches presented in [9, 10, 13, 14, 21] take care of all basic constraints.

3.2.2. Method. In order to embed a VN onto the substrate, we need to find appropriate mappings for both its vertices as well as edges. Therefore, the VNE problem can be decomposed into vertex and edge mapping, and for this, various approaches can be adopted.

(A) Vertex Mapping

(i) Iterative Method. The iterative method of mapping VNs onto substrate networks was presented in [16], where nodes are categorized as backbone and access nodes. In this method, first backbone nodes are mapped onto the substrate, then access nodes are connected to backbone nodes and shortest paths are computed between these nodes, after this, link capacities are calculated and in the end it is ensured that the backbone nodes have been mapped optimally.

The vertex mapping approach in [14] can also be considered to be iterative, as it selects one node (of highest degree) in each step. The process is repeated for remaining nodes moving on from nodes with highest to lowest degrees until all of them get mapped on the substrate.

Vertex mapping approach in [20] divides the entire VN topology into a set of elementary clusters. The decomposition of VNs is based on star topology, where nodes are characterized as hub and spoke. Mapping of a VN is done sequentially by assigning the decomposed star topology based VNs to the substrate, one at a time.

(ii) Simulated Annealing. Simulated annealing approach has been used to find the optimal topology for a given communication pattern in [17], where the goal is to find optimal reconfiguration policies.

(iii) Greedy Node Mapping. Greedy node mapping approach maps vertices on nodes with maximum resources [9, 10, 18]. The advantage of using this method is to minimize the use of substrate resources at bottleneck nodes/links, which helps in satisfying the requirements of future VN requests which demand fewer resources.

(iv) Baseline Approach (BLA). The baseline approach (BLA) of mapping vertices on any available substrate nodes which can satisfy their demand (by only evaluating if, $r^n \geq d^v$) was presented in [11]. VNs embedded using this approach can incur less cost as compared to GNM. However, BLA does not take into account the possibility of a node becoming bottleneck at the time of mapping a VN's vertex.

(v) Mixed Integer Programming. In [13], the authors have formulated a solution to the VNE problem by using mixed integer programming (MIP) formulation. Vertex mapping in this solution is done by using two techniques. In first algorithm vertex mapping is done deterministically and is called D-Vine (deterministic rounding based virtual network embedding algorithm) while second algorithm does it randomly and is presented as R-Vine (randomized rounding based virtual network embedding algorithm).

(B) Edge Mapping. The edge mapping approaches can be devised based on flows. The flows can be categorized as either unsplittable or splittable.

(i) Shortest Path Mapping (SPM). The shortest path mapping is a cost efficient approach of mapping edges on substrate paths. It has been used as a primary approach for edge mapping in a number of solutions. The solutions proposed in [14–18, 20, 21] have used SPM for edge mapping while the one given in [9] also uses it for unsplittable flows.

(ii) Multicommodity Flow. In case of splittable flows, the multicommodity flow based approach has been used for edge mapping [9, 10, 13, 19].

3.2.3. VN Requests. Virtual network requests can be either specified in advance (offline problem) or arrive as part of a dynamic process (online problem). The solutions given in [16, 18, 20] solve offline version of VNE problem, while the ones proposed in [9, 13, 14] solve it as an online problem.

3.2.4. Type of Mapping (TOM). A VN embedding algorithm may be carried out either in a distributed or centralized manner. The solutions proposed in [9, 10, 16–18, 21] map VN requests in a centralized way while the ones proposed in [15, 20] assign VN requests to substrate networks using a distributed process.

3.2.5. Adaptability. After VN requests get mapped on the substrate, a VNE solution may need to provide the feature of adaptability, that is, respond to variations in either substrates or VNs. This may be required in either of the following scenarios.

(i) A user may add new requirements for an embedded VN request. A set of new candidate resources may need to be identified in response to the additional requirements.

The above mentioned change in user's requirements was taken care of and a solution in this regard was proposed in [15]. A solution to the problem of dynamically reconfiguring topology of an overlay network in response to changes in communication requirements was also presented in [17].

(ii) A physical node/link may be hosting many virtual nodes/links. In case, a problem occurs with a single physical node/link then several virtual nodes/links will be affected. Therefore, the physical/virtual node and link failures should

Table 2: Solution to VNE constraints and approaches adopted by various proposals.

Reference	ABC		VN requests	OM	Adaptability	Method		Opt obj
	C	NC				Vertex	Edge	
[9]	✓		ONL	Cent	PM	GNM	SPM/MCF	MRU
[14]	✓		ONL	Cent	LF	IM	SPM	MQR
[15]	✓		ONL	Dist	NLF, CUR	IM	SPM	MAT
[16]		NCap, AC	OFL	Cent	NC	IM	SPM	MNC
[17]		NCap, AC	ONL	Cent	CUR	SA	SPM	MOUC
[18]		AC	OFL	Cent	CNC	GNM	SPM	MRU
[13]	✓		ONL	Cent	NC	D-Vine/R-Vine	MCF	MARR
[19]		NCap	OFL	Cent	NC	—	MCF	MAR
[20]		AC	OFL	Dist	NC	IM	SPM	MNC
[21]	✓		ONL	Cent	NC	SA	SPM	MMT
[10]	✓		ONL	Cent	Re-opt	GNM/ D-Vine/R-Vine	SPM/MCF	MAR

Where in Table 2—ABC: all basic constraints data, NC: not considered, AC: admission control, OFL: offline, Dist: distributed, LF: link failures, CUR: change in user's requirements, IM: iterative method, GNM: greedy node mapping, D-Vine: deterministic rounding based virtual network embedding algorithm, R-Vine: randomized rounding based virtual network embedding algorithm, MCF: multicommodity flow, Opt Obj: optimization objective, MQR: maximize quality and resilience, MNC: minimize network cost, MARR: maximize acceptance ratio and revenue, MAR: maximize acceptance ratio, C: considered, NCap: node capacity, ONL: online, Cent: centralized, PM: path migration, NLF: node and link failures, CNC: change in network conditions, SA: simulated annealing, SPM: shortest path mapping, TOM: type of mapping, MRU: maximize resource usage, MAT: minimize adaptation time, MOUC: minimize overlay usage cost, Re-opt: reoptimization, MMT: minimize mapping time.

always be kept into consideration and virtual nodes and links should be re-mapped if a failure occurs.

An approach to take care of vertex/node as well as edge/link failures of VNs and substrates, and remap VN's vertices and edges on alternate nodes and links has been presented in [15]. Provision of path resiliency by constructing alternate one-hop overlay routes via intermediary nodes was part of the solution proposed in [14].

(iii) The concept of "path migration" by either changing the splitting ratios of existing paths or selecting new underlying paths can enable a substrate to accommodate a newly arrived VN.

The idea of path migration was presented in [9].

(iv) After being mapped, a VN may be reconfigured to be assigned to a different set of substrate nodes and links upon arrival of a new VN request.

In [18], a solution termed as "VN assignment with reconfiguration" has been proposed which states that node and link assignments to an embedded VN request are not fixed for its lifetime and may be changed at the arrival of a new VN request in order to better utilize substrate resources.

3.2.6. Optimization Objective. VNE is a resource constrained problem and in addition to the main objective of optimizing the use of substrate resources, proposed solutions for this problem have focused on several other factors as well.

The solutions proposed in [9, 18] focus on maximizing the usage of substrate resources, while in [14] the focus has been on mapping of virtual networks to achieve high quality and resilience.

In case, a substrate node or link fails, the virtual vertices or edges mapped on it should be moved quickly enough (adaptation time should be minimized) to other nodes or links which can satisfy resource requirements, which was the objective of distributed fault-tolerant embedding algorithm, as proposed in [15].

The objective of mapping virtual networks onto a common substrate in such a way that can enable a network to support any traffic pattern allowed by a general set of constraints while minimizing the network cost was presented in [16].

Using dynamic overlay topology reconfiguration, a solution was proposed in [17] to minimize the cost of using an overlay. The two types of costs considered were occupancy cost and reconfiguration cost.

The objective of maximizing the acceptance ratio and revenue was achieved by doing coordinated node and link mapping as presented in [13].

The goal of maximizing the number of accepted VNs by preallocating resources for nodes and solving link mapping based on multicommodity flow was proposed in [19].

The objective of minimizing the network cost by mapping VNs using a distributed method and in the process achieving balanced load-sharing among all substrate nodes was the focus of solution proposed in [20].

The goal of minimizing mapping time was achieved by using simulated annealing technique and presented in [21].

Table 2 shows how solutions to the VNE problem have handled challenges associated with it.

4. Hybrid BLA-BNRM Approach (HBNRM)

VN mapping process starts by assigning vertices to nodes, then proceeds on to find k-shortest paths [22] between each pair of mapped nodes, and finishes by mapping edges onto paths that satisfy their demand. Our approach is inspired to some extent by [9] as we use similar notations to denote both virtual and substrate networks. However, we use a different k-shortest paths algorithm [22] than edge disjoint

paths [9], as it gives us a better choice of mapping an edge on substrate paths. The proposed approach solves VNE problem by considering all basic constraints (Section 3.2.1), handles VN requests online (Section 3.2.3), type of mapping is centralized (Section 3.2.4) whereas optimization objective (Section 3.2.6) is to maximize acceptance ratio (MAR). This section will initially present a description of our vertex mapping approach and in the second phase, edge mapping approach will be described.

4.1. Vertex Mapping. First step of the proposed solution starts by finding candidate nodes of the substrate, which can map vertices by satisfying their demands. In this phase, each vertex ($v \in V \wedge d^v \in D^V$) has to be mapped to a different node ($n \in N \wedge a^n \in A^N$). Several approaches can be adopted for this purpose and each will affect how VNs get mapped as well as substrate resources are utilized in the process. In this paper, two existing (i.e., BLA and GNM) and one proposed approach (HBNRM) will be evaluated.

Before going through the details of our vertex mapping approach (HBNRM), it is important to give definition of bottleneck as well as exhausted nodes.

4.1.1. Bottleneck Nodes (B(N)). The idea to minimize the use of substrate resources from bottleneck nodes and links was presented in [9], while, the concept of bottleneck links was also mentioned in [23]. Nodes and links having lack of residual capacities to map vertices/edges and hence resulting in rejection of a VN request were termed as bottlenecks in [10]. We proposed definitions for bottleneck nodes and links of a substrate in terms of their capability of mapping vertices and edges of VNs due to arrive in future in the mapping process in [11].

We define a node as a bottleneck if, it is unable to map two vertices (of highest capacity) of future VN requests. In other words, residual capacity of a bottleneck node is less than a certain value ($r^n < \text{val}_n$).

4.1.2. Exhausted Nodes (E(N)). An exhausted node is a bottleneck node, whose resources get completely utilized ($r^n = 0$).

We now describe our vertex mapping approach. In future work of [11] it was mentioned that one possible extension of that work could be to investigate how the two approaches (i.e., BLA and BNRM) could be combined in order to maximize the number of virtual networks that are mapped, while still avoiding bottleneck nodes. Another objective of the new approach should be to utilize benefits of both BLA (baseline approach) and GNM (greedy node mapping approach) while trying to minimize their disadvantages. In other words, we should be able to minimize bottleneck nodes of a substrate (GNM's advantage) while trying to minimize the cost to map a VN (BLA's advantage). We have named this approach as HBNRM (Hybrid BLA-BNRM) approach which is presented below.

One important component of HBNRM is the use of node exhaustion limit (nel) values. Nel is a value which is used to make sure that a node does not become bottleneck. The vertex is only mapped on the node if, after mapping, the node has resources equal to or greater than nel. Nel values are used according to the rule defined below.

80/50 Rule for NEL Values. We start by using a nel value (val_n, defined above) which ensures that a node does not become bottleneck after mapping a vertex. This value is increased or decreased according to the following criteria.

(i) When about eighty percent (80%) of nodes reach the set nel value in an interval (or request window), it is decreased to the next level and then same rule gets applied to the new value.

Experiments have shown that once about eighty percent nodes reach a set nel value then it may be decreased otherwise, VNs may get dropped for next interval even though sufficient node resources maybe present in the substrate.

(ii) If greater than fifty percent (50%) VNs get dropped in an interval in early stages of VN mapping process, then, nel value is increased to the next level.

When VNs get rejected in early stages of mapping process then, it could mean that sufficient node resources are present in the substrate but link resources have started to exhaust as a result of mapping vertices on same nodes repeatedly. By increasing nel value it can be ensured that nodes which were not selected previously can now be selected for mapping of new VN requests and as a result more link resources could be made available.

(iii) If greater than fifty percent (50%) VNs get dropped in an interval in later stages of VN mapping process, then, nel value is decreased to the next level.

When VNs get rejected in later stages of mapping process, then it could mean that link resources have started to exhaust and a decrease in nel value may map vertices on different nodes and as a result some unused links may become available for edge mapping.

So, nel value is increased or decreased when either of above conditions occurs. In Section 5.2, we will explain cases where nel values were increased or decreased and which condition of 80/50 rule was applied.

Another important point about 80/50 rule is that it can be modified according to number of VNs considered for an interval (request window). In this paper, 50 VNs constitute an interval (request window), if number of VNs are reduced to 40 for an interval this could change the rule to 85/55. Similarly if VN requests are increased to 60 then rule might become 75/40.

The vertex mapping function for HBNRM can be given as

$$V^M : (V, D^V) \longrightarrow (N, A^N). \quad \text{Subject to } ((r^n \geq d^v), ((r^n - d^v) \geq \text{nel}_t)). \tag{5}$$

1: Take a VN request.
2: Find a unique substrate node, for every vertex, having sufficient resources to satisfy the CPU demand,
(according to the selected vertex mapping function, BLA, HBNRM, or GNM), start from the vertex demanding most resources.
3: **If** all the vertices can be mapped at this stage, **then**, generate residual node capacities according to (2a), **else**, **GOTO** 5.
4: Call "edge mapping algorithm."
5: **If** this was the last request, **then**, **stop**, **else**, **GOTO** 1.

Algorithm 1: Vertex mapping.

1: Take the request which has successfully passed the vertex mapping stage.
2: **for** (each_edge_in_the_request) **do**
2.1: Search k-shortest paths incrementally between pair of vertices connected by the edge
(mapped on nodes by the vertex mapping algorithm).
2.2: **Stop** searching in above Step 2.1, when either edge demand is satisfied (according to edge
mapping function, SPM) or all paths have been searched.
2.3: **If**, edge request is not satisfied, **then**, **GOTO** 3, **else**, generate residual link capacities of the
selected path according to (2b) and **GOTO** 4.
2.4: **end for**
3: **If**, this was the first edge of VN, **then,** return node resources to the substrate according to
(3a), **else**, return both the link and node resources to the substrate (as defined in (3a) and (3b)).
4: **If** this was the last request, **then**, **stop**, **else,** call "vertex mapping algorithm."

Algorithm 2: Edge mapping.

The nel_t in (5), defines the time at which a VN arrives and is compared with a certain nel value. Vertex mapping function in (5) starts by checking all the vertices (V) and first selects the one which demands most processor resources ($\max(D^V)$). The benefit of doing this is that if, the substrate cannot satisfy the demand of this vertex then the mapping process stops here for this VN and requirements of remaining vertices do not need to be checked, which saves amount of computations according to number of vertices in a VN. In this way, if demand of the first vertex is satisfied then the process is repeated for all remaining vertices moving on from vertex demanding most to the least resources.

The vertex mapping algorithm is defined in Algorithm 1.

VN requests are satisfied by using first come first served (FCFS) approach and the process begins by assigning vertices to unique substrate nodes according to the selected vertex mapping function (BLA, HBNRM, or GNM). In the next step, residual capacities of nodes (selected for mapping vertices of the VN) are generated. The edge mapping algorithm is called only if, all vertex requests of a VN can be satisfied at this stage.

4.2. Edge Mapping. The next step is to map an edge ($e \in E \wedge d^e \in D^E$) on a substrate path ($P^S_{m,n}$) containing one or more than one links. In the proposed solution, k-shortest paths [22] are found for each edge. The next step is to calculate resources on a path. To achieve this objective, we take link with minimum resources in the path ($\min(P^S_{m,n})$) and match edge demand with it. If this link can satisfy edge request, then remaining links will surely be able to do that.

The approach of mapping an edge on shortest of k-shortest paths, which can satisfy its demand, termed as shortest path mapping (SPM), was presented in [9, 11]. Edge mapping function for the SPM can be given as

$$\begin{aligned} &E^M : (E, D^E) \longrightarrow P^S_{m,n}. \\ \text{Subject to} \quad &\left(d^e \leq \left(\min\left(P^S_{m,n}\right)\right)\right). \end{aligned} \tag{6}$$

The edge mapping algorithm is defined in Algorithm 2.

Edge mapping phase of the solution assigns all edges to the substrate paths and is executed number of times the total edges in a VN. It starts by finding out shortest of k-shortest paths (for $k = 1$) for each edge of the VN. In the next step, resources on that path are calculated and edge demand is matched with it. If this path has sufficient resources to satisfy the edge demand, then it is selected for mapping that particular edge. Otherwise, the process continues till either a path among the k-shortest paths can satisfy edge request or no path has sufficient resources to satisfy edge demand. In case, after mapping the initial edges of a VN, there comes a point when the requirements of a certain edge cannot be satisfied by the substrate, then the resources reserved initially for edges and vertices of a VN need to be returned to the substrate (Step 3). At this point, if there are sufficient path resources to satisfy request of all the edges then the VN is completely mapped (VN request satisfied).

5. Experimental Setup and Evaluation

This section is divided into two parts; first describes experimental setup while second presents evaluation results. The proposed solution has been implemented using Matlab.

5.1. Experimental Setup. Substrate networks have been generated using the BRITE tool [24] whereas, virtual networks have been created using Matlab.

5.1.1. Substrate Networks. The proposed solution has been tested on four different substrate networks (S_1, S_2, S_3, and S_4). Two of these networks (S_1 and S_2) consist of 100 nodes and about 500 links [9, 11, 12], while S_3 and S_4 comprise of 100 nodes and about 300 links [25]. The node resource (CPU) as well as link resource (bandwidth) is assigned different values from 10 to 100 units. The size of substrates (S_1 and S_2) can be compared with that of a medium sized ISP [9].

S_1: Node and link resources are randomly chosen from 20 to 100.

S_2: Node and link resources are randomly chosen from 10 to 100.

S_3: Node and link resources are randomly chosen from 20 to 100.

S_4: Node and link resources are randomly chosen from 30 to 100.

The substrates S_1 and S_2 have more links (are densely connected) but contain different number of node and link resources. The substrates S_3 and S_4 have fewer links (are sparsely connected) and also contain different number of node and link resources.

5.1.2. Virtual Networks. Two different sets of VNs have been mapped onto substrate networks; each set differs from the other, by number of edges.

Set 1. Number of vertices of a VN is randomly chosen between 2 and 10 and vertices are randomly connected with the probability 0.3.

Set 2. Number of vertices of a VN is randomly chosen between 2 and 10 and vertices are randomly connected with the probability 0.5.

Set 2 is similar to the setup presented in [9, 11, 12, 18] while Set 1 resembles with the one given in [11, 25]. Vertex and edge resources in both sets are randomly chosen from 1 to 5. Two sets of VNs put different demand on substrate's paths and can impact on how VNs are mapped.

5.2. Evaluation. The main focus of evaluation is to analyze the effect of using three different vertex mapping approaches (BLA, GNM, and HBNRM). Evaluation is done by mapping sparsely and densely connected VNs on sparsely and densely connected substrates. Results are presented in the form of graphs and tables which show the impact of using each approach on mapping VN requests, the way resources are utilized, nodes become bottleneck and get exhausted.

Results for densely connected substrates (S_1 and S_2) will be presented first which will be followed by a discussion about sparsely connected substrates (S_3 and S_4). The section ends with a summary of results.

5.2.1. Densely Connected Substrates (S_1 and S_2)

(a) VN Mapping ($Map(M_n)$). Figures 3 to 8 represent evaluation results, where requested VNs are shown on horizontal axis, whereas mapped VNs by evaluated approaches and cost incurred in the process are presented on vertical axis.

When a substrate is densely connected, has higher number of resources (i.e., S_1), and a set of sparsely connected VNs (VN-Set 1) is mapped on it, then mapping results are almost similar for all methods (Figure 3). VNs put less demand on substrate's paths and even if vertices are mapped on different locations in the substrate by each approach, sufficient resources are available for edge mapping and therefore, mapping results are almost similar. However, among three approaches, GNM has highest cost for VN mapping, whereas BLA has least cost (Figure 3).

An almost similar VN mapping trend (like that of Figure 3) can be seen when a set of densely connected VNs (VN-Set 2) is mapped on S_1 (Figure 4).

When the substrate is densely connected but has comparatively less number of resources (i.e., S_2) and a set of densely connected VNs (VN-Set 2) is mapped on it, then mapping results are quite different than previous substrate (S_1), as shown by Figure 5. In case of GNM, vertices can be mapped at a distance from each other and since the substrate has fewer resources so, the edge demand can become difficult to fulfill. In this case, BLA and HBNRM give almost similar mapping of VN requests (Figure 5).

Tables 3 to 20 present evaluation results for the rest of evaluation criteria where number of mapped VNs in each interval, cost incurred in the process, percentage of overall utilized resources as well as bottleneck and exhausted nodes are shown. Mapped VNs ($\text{Map}(M_n)$) and cost of mapped VNs ($C(M_n)$) are shown for each interval in Tables 3 to 20, as compared to overall mapped VNs and mapping cost, shown in Figures 3 to 8. Percentage of utilized resources ($U(S_r)$), bottleneck and exhausted nodes ($B(N)$, $E(N)$) are also important to analyze, as they represent how each approach utilizes substrate's resources.

(b) Resource Utilization ($U(S_r)$). Resource utilization for Tables 3 to 20 should be viewed based on the following factors.

(i) When same number of VNs are mapped by evaluated methods at a particular instant of time.

The actual cost comparison between any compared approaches can be seen when same number of mapped VNs are analyzed from initial set of VNs. Cost is only incurred when VNs are mapped and once VN mapping decreases so will incurred cost. Secondly, since we have used random VNs

Table 3: BLA resource utilization (VN-Set1; substrate N/W: S_1).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	1662	4.30%	9%	9%
51–100	100%	1773	4.59%	11%	11%
101–150	100%	1878	4.86%	15%	13%
151–200	100%	2056	5.32%	11%	12%
201–250	100%	2003	5.18%	12%	12%
251–300	98%	2110	5.46%	13%	14%
301–350	98%	2241	5.80%	12%	10%
351–400	94%	1932	5.00%	12%	13%
1–400	98.75%	15655	40.50%	95%	94%

Table 4: GNM resource utilization (VN-Set 1; substrate N/W: S_1).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	1956	5.06%	0	0
51–100	100%	2024	5.24%	0	0
101–150	100%	1996	5.16%	0	0
151–200	100%	2083	5.39%	0	0
201–250	100%	2071	5.36%	0	0
251–300	100%	2011	5.20%	0	0
301–350	100%	2084	5.39%	78%	0
351–400	86%	1780	4.60%	22%	9%
1–400	98.25%	16005	41.40%	100%	9%

Table 5: HBNRM resource utilization (VN-Set 1; substrate N/W: S_1).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	1703	4.41%	0	0
51–100	100%	1843	4.77%	0	0
101–150	100%	1996	5.16%	0	0
151–200	100%	1970	5.10%	0	0
201–250	100%	2165	5.60%	0	0
251–300	100%	2288	5.92%	0	0
301–350	100%	2027	5.24%	92%	0
351–400	92%	1747	4.52%	4%	0
1–400	99%	15739	40.72%	96%	0

Table 6: BLA resource utilization (VN-Set 2; substrate N/W: S_1).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	2707	7%	9%	9%
51–100	100%	2998	7.76%	13%	12%
101–150	100%	3238	8.38%	14%	14%
151–200	100%	3502	9.06%	13%	13%
201–250	98%	3655	9.46%	14%	14%
251–300	68%	1627	4.21%	8%	8%
301–350	70%	1695	4.38%	8%	8%
351–400	70%	2506	6.48%	8%	8%
1–400	88.25%	21928	56.73%	87%	86%

Table 7: GNM resource utilization (VN-Set 2; substrate N/W: S_1).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	3314	8.57%	0	0
51–100	100%	3469	8.97%	0	0
101–150	98%	3338	8.64%	0	0
151–200	100%	3649	9.44%	0	0
201–250	96%	3074	7.95%	0	0
251–300	84%	2499	6.46%	0	0
301–350	74%	1943	5.03%	44%	0
351–400	60%	1213	3.14%	56%	0
1–400	89%	22499	58.20%	100%	0

Table 8: HBNRM resource utilization (VN-Set 2; substrate N/W: S_1).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	2708	7.01%	0	0
51–100	100%	3052	7.90%	0	0
101–150	100%	3536	9.15%	0	0
151–200	98%	3565	9.22%	0	0
201–250	74%	2250	5.82%	0	0
251–300	80%	3035	7.85%	0	0
301–350	84%	2570	6.65%	87%	0
351–400	78%	2340	6.05%	5%	0
1–400	89.25%	23056	59.65%	92%	0

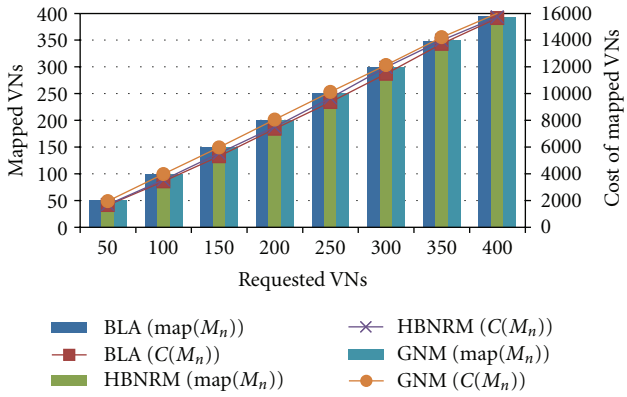

Figure 3: VN-Set 1; Substrate N/W: S_1.

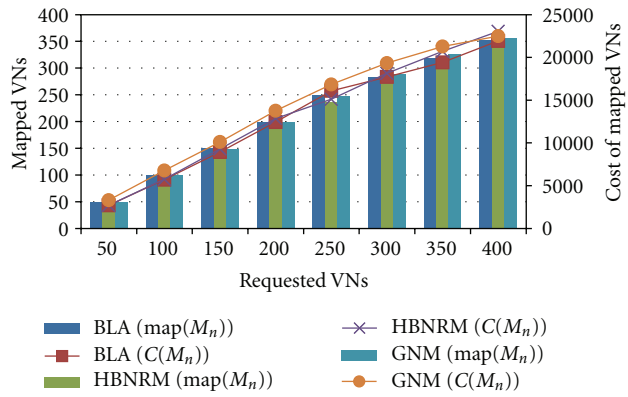

Figure 4: VN-Set 2; Substrate N/W: S_1.

so it would be unfair to compare an approach which maps more VNs having higher number of nodes and in the process incur more cost with the one which maps more VNs with lesser number of nodes and also costs less. Since, mapped VNs from initial set would be same for all the approaches so we will make cost comparison based on that.

(ii) The overall VNs mapped by using a certain method.

When more overall VNs are mapped by using a particular method then, it can incur more cost.

For sparsely connected set of VNs (VN-Set 1) three approaches map similar number of VNs when initial 250 VN requests arrive on S_1 (Tables 3 to 5). At this point, average cost of mapping a VN, $C(M_{\text{avg}})$ for BLA is 37.49, for GNM is 40.52, whereas for HBNRM is 38.71 units of substrate

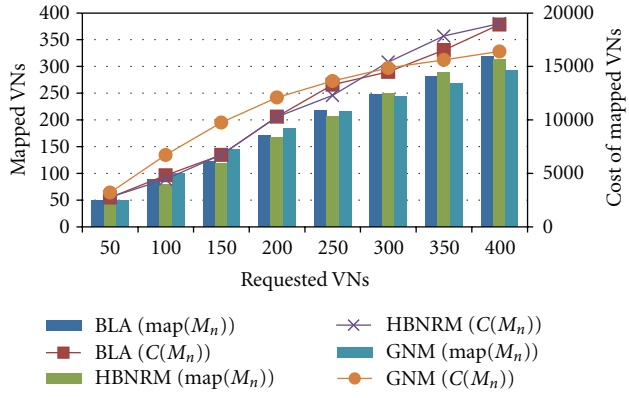

Figure 5: VN-Set 2; Substrate N/W: S_2.

Table 9: BLA resource utilization (VN-Set 2; substrate N/W: S_2).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	2744	7.83%	11%	9%
51–100	76%	2080	5.93%	9%	10%
101–150	70%	1907	5.44%	10%	9%
151–200	98%	3580	10.21%	14%	15%
201–250	90%	2990	8.53%	14%	14%
251–300	60%	1198	3.42%	7%	7%
301–350	70%	2038	5.81%	11%	10%
351–400	72%	2387	6.81%	9%	9%
1–400	79.5%	18924	53.98%	85%	83%

Table 10: GNM resource utilization (VN-Set 2; substrate N/W: S_2).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	3212	9.16%	0	0
51–100	100%	3511	10.02%	0	0
101–150	90%	3035	8.66%	0	0
151–200	78%	2348	6.70%	0	0
201–250	64%	1535	4.38%	0	0
251–300	54%	1218	3.47%	0	0
301–350	52%	761	2.17%	0	0
351–400	46%	785	2.24%	0	0
1–400	73%	16405	46.80%	0	0

Table 11: HBNRM resource utilization (VN-Set 2; substrate N/W: S_2).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	2775	7.92%	0	0
51–100	58%	1679	4.79%	0	0
101–150	78%	2293	6.54%	0	0
151–200	100%	3538	10.09%	0	0
201–250	78%	2010	5.73%	0	0
251–300	84%	3118	8.89%	0	0
301–350	80%	2432	6.94%	0	0
351–400	50%	1170	3.34%	76%	0
1–400	78.5%	19015	54.24%	76%	0

Table 12: BLA resource utilization (VN-Set 1; substrate N/W: S_3).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	1707	6.34%	9%	9%
51–100	100%	2181	8.09%	11%	11%
101–150	94%	2166	8.04%	13%	11%
151–200	78%	1663	6.17%	9%	11%
201–250	96%	2335	8.67%	13%	12%
251–300	94%	2257	8.38%	9%	10%
301–350	90%	2164	8.03%	14%	12%
351–400	64%	1229	4.56%	6%	7%
1–400	89.5%	15702	58.28%	84%	83%

Table 13: GNM resource utilization (VN-Set 1; substrate N/W: S_3).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	2230	8.28%	0	0
51–100	100%	2274	8.44%	0	0
101–150	100%	2373	8.81%	0	0
151–200	96%	2188	8.12%	0	0
201–250	86%	2116	7.85%	0	0
251–300	88%	2110	7.83%	0	0
301–350	60%	1110	4.12%	0	0
351–400	58%	1133	4.21%	19%	0
1–400	86%	15534	57.66%	19%	0

Table 14: HBNRM resource utilization (VN-Set 1; substrate N/W: S_3).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	1753	6.51%	0	0
51–100	100%	2129	7.90%	0	0
101–150	100%	2279	8.46%	0	0
151–200	84%	1952	7.24%	0	0
201–250	100%	2476	9.19%	0	0
251–300	100%	2582	9.58%	0	0
301–350	64%	1233	4.58%	82%	0
351–400	54%	902	3.35%	1%	0
1–400	87.75%	15306	56.81%	83%	0

Table 15: BLA resource utilization (VN-Set 2; substrate N/W: S_3).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	98%	2837	10.53%	9%	9%
51–100	78%	2455	9.11%	9%	8%
101–150	28%	919	3.41%	4%	3%
151–200	42%	541	2.01%	3%	4%
201–250	42%	1080	4.01%	5%	4%
251–300	10%	183	0.68%	1%	2%
301–350	16%	651	2.42%	4%	2%
351–400	62%	1637	6.08%	5%	6%
1–400	47%	10303	38.24%	40%	38%

Table 16: GNM resource utilization (VN-Set 2; substrate N/W: S_3).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	3956	14.68%	0	0
51–100	90%	2974	11.04%	0	0
101–150	78%	2613	9.70%	0	0
151–200	52%	1019	3.78%	0	0
201–250	46%	1122	4.16%	0	0
251–300	40%	704	2.61%	0	0
301–350	24%	437	1.62%	0	0
351–400	22%	317	1.18%	0	0
1–400	56.5%	13142	48.78%	0	0

Table 17: HBNRM resource utilization (VN-Set 2; substrate N/W: S_3).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	96%	2794	10.37%	0	0
51–100	82%	2935	10.89%	0	0
101–150	52%	871	3.23%	0	0
151–200	50%	1282	4.76%	0	0
201–250	68%	1727	6.41%	0	0
251–300	60%	1808	6.71%	0	0
301–350	52%	1365	5.07%	0	0
351–400	32%	958	3.56%	0	0
1–400	61.5%	13740	51.00%	0	0

Table 18: BLA resource utilization (VN-Set 2; substrate N/W: S_4).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	2884	9.92%	11%	11%
51–100	100%	3423	11.78%	13%	12%
101–150	70%	1961	6.75%	7%	7%
151–200	48%	1442	4.96%	5%	6%
201–250	60%	1821	6.27%	7%	7%
251–300	22%	304	1.05%	1%	1%
301–350	22%	582	2.00%	3%	2%
351–400	34%	1009	3.47%	3%	4%
1–400	57%	13426	46.20%	50%	50%

Table 19: GNM resource utilization (VN-Set 2; substrate N/W: S_4).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	4143	14.26%	0	0
51–100	92%	3287	11.31%	0	0
101–150	66%	2341	8.05%	0	0
151–200	64%	1916	6.59%	0	0
201–250	52%	1202	4.14%	0	0
251–300	40%	1552	5.34%	0	0
301–350	46%	1338	4.60%	0	0
351–400	22%	372	1.28%	0	0
1–400	60.25%	16151	55.57%	0	0

Table 20: HBNRM resource utilization (VN-Set 2; substrate N/W: S_4).

Req (M_n)	Map (M_n)	C (M_n)	U (S_r)	B (N)	E (N)
1–50	100%	2997	10.31%	0	0
51–100	96%	3449	11.87%	0	0
101–150	68%	2170	7.47%	0	0
151–200	48%	1688	5.81%	0	0
201–250	62%	2365	8.14%	0	0
251–300	72%	2406	8.28%	0	0
301–350	50%	1169	4.02%	0	0
351–400	48%	778	2.68%	0	0
1–400	68%	17022	58.57%	0	0

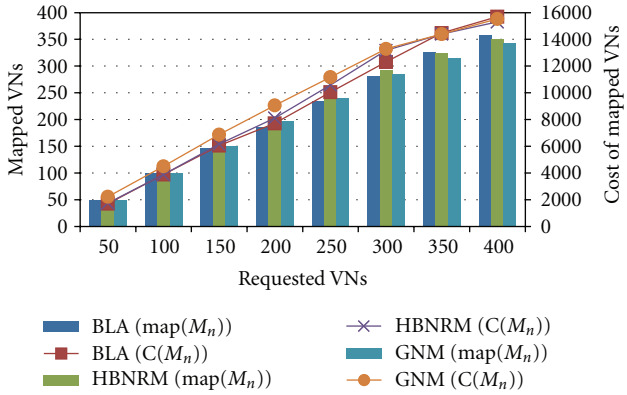

Figure 6: VN-Set 1; Substrate N/W: S_3.

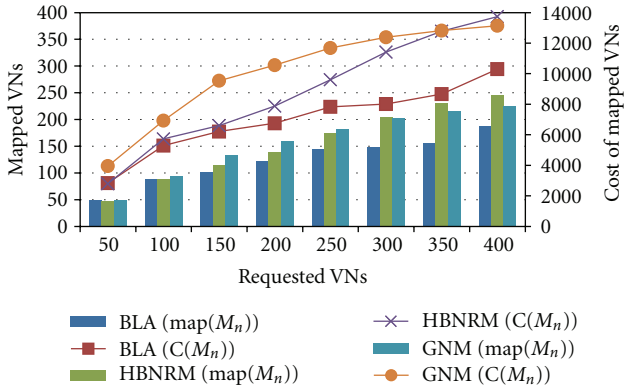

Figure 7: VN-Set 2; Substrate N/W: S_3.

resources. So BLA's cost of mapping VNs is the least whereas GNM's mapping cost is the highest among three approaches.

Overall as well, GNM uses 0.9% more resources as compared to BLA and 0.68% more when matched with HBNRM (Tables 3 to 5). For this set of VNs almost similar number of overall VNs is mapped by using either of three approaches (Tables 3 to 5).

For densely connected set of VNs (VN-Set 2) three approaches map similar number of VNs when initial 100 VN requests arrive on S_1 (Tables 6 to 8). At this point, average cost of mapping a VN, $C(M_{\text{avg}})$ for BLA is 57.05, for GNM is 67.83, whereas for HBNRM is 57.6 units of substrate resources. So, in this case as well BLA's cost of mapping VNs is least whereas GNM's mapping cost is the highest among three approaches.

Overall, HBNRM uses 2.92% more resources as compared to BLA and 1.45% more when matched with GNM (Tables 6 to 8). For this set of VNs although overall mapped VNs is almost similar by using either of three approaches but their mapping trends are different and therefore, more overall resources are utilized by HBNRM (Tables 6 to 8).

When a set of densely connected VNs (VN-Set 2) is mapped on S_2 the evaluated approaches map similar number of VNs when initial 50 VN requests arrive (Tables 9 to 11). At this point, average cost of mapping a VN, $C(M_{\text{avg}})$ for BLA is 54.88, for GNM is 64.24, whereas for HBNRM is 55.5 units

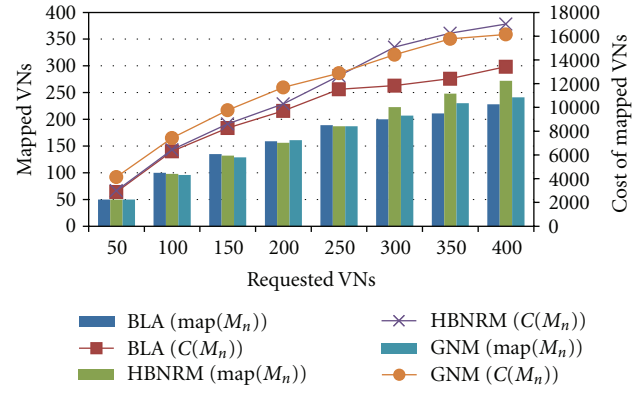

Figure 8: VN-Set 2; Substrate N/W: S_4.

of substrate resources. So, in this case as well BLA's cost of mapping VNs is the least whereas GNM's mapping cost is the highest among three approaches.

Overall, BLA uses 7.18% more resources as compared to GNM (Tables 9 and 10). However, BLA maps 6.5% more VNs as compared to GNM as well (Tables 9 and 10). When matched with HBNRM although BLA maps 1% more VNs (Tables 9 and 11) but their mapping trends are different and it uses 0.26% less resources (Tables 9 and 11).

(c) Bottleneck Nodes ($B(N)$). Nodes start to become bottleneck from VN interval-1 (VNs 1–50) when BLA is used for mapping on S_1 or S_2 (Tables 3, 6, and 9). In case of GNM there are no bottleneck nodes till the sixth interval on S_1 (Tables 4 and 7). When approach used is HBNRM then also nodes start to become bottleneck after sixth interval on S_1 (Tables 5 and 8). For HBNRM nodes do not become bottleneck as long as nel value is set according to bottleneck limit (val_n) as defined in Section 4.1. Starting nel value for nodes in Tables 5, 8, 11, 14, 17, and 20 is 10 ($\text{val}_n = 10$), since, maximum capacity of any vertex for both sets of VNs is 5. If nel value needs to be decreased to comply with either condition (i) or (iii) of 80/50 rule for nel values (Section 4.1) it is initially set to 5 for these sets of VNs ($\text{val}_n = 5$). The objective behind this new value is that a node will still be able to map at least one vertex of highest capacity of future requests. On substrate S_1 for both sets of VNs nel value needs to be decreased for seventh interval to comply with condition (i) of 80/50 rule for nel values (Tables 5 and 8). So, bottleneck nodes start to appear from there onwards.

In case of GNM there are no bottleneck nodes on S_2 (Table 10), as it maps fewer VNs than compared approaches (Tables 9 to 11). When approach used is HBNRM then, nodes start to become bottleneck after seventh interval (Table 11). According to condition (i) of 80/50 rule for nel values (Section 4.1) nel value needs to be decreased for eighth interval (Table 11). So, bottleneck nodes start to appear in that interval.

(d) Exhausted Nodes ($E(N)$). When BLA is used on S_1 or S_2, nodes start to exhaust from first interval (Tables 3, 6, and 9). In case of GNM only 9% nodes exhaust for VN-Set 1 on S_1

(Table 4) whereas no node resource exhausts for VN-Set 2 (Table 7), similar trend can be seen for HBNRM where no node resource exhausts (Tables 5 and 8). When GNM and HBNRM are used on S_2, no node resources exhaust (Tables 10 and 11).

5.2.2. Sparsely Connected Substrates (S_3 and S_4)

(a) VN Mapping ($Map(M_n)$). When a substrate is sparsely connected (i.e., S_3) and a set of sparsely connected VNs (VN-Set 1) is mapped on it, then mapping results are almost similar for all methods (Figure 6). VNs put less demand on substrate's paths and even if vertices are mapped on different locations in the substrate by each approach, sufficient resources are available for edge mapping and therefore, mapping results are almost similar (Figure 6).

When a set of densely connected VNs (VN-Set 2) is mapped on the same substrate (S_3), then mapping results are quite different than previous set of VNs (Figure 7). In case of BLA, vertices can be mapped repeatedly on same nodes and since the substrate has fewer link resources they can exhaust early. Therefore, the edge demand can become difficult to fulfill. In this case, GNM and HBNRM give almost similar mapping of VN requests until the arrival of 300 VNs (Figure 7). However HBNRM overall, maps more VNs as compared to GNM (Figure 7).

When the substrate is sparsely connected, has comparatively higher number of resources (i.e., S_4) and a set of densely connected VNs (VN-Set 2) is mapped on it, then too trend of mapping VNs by evaluated approaches is quite similar to that of previous substrate (S_3, Figure 7), as shown by Figure 8. In this case, GNM and HBNRM give almost similar mapping of VN requests until the arrival of 250 VNs (Figure 8). However HBNRM overall, maps more VNs as compared to GNM (Figure 8).

(b) Resource Utilization ($U(S_r)$). For sparsely connected set of VNs (VN-Set 1) three approaches map similar number of VNs when initial 100 VN requests arrive on S_3 (Tables 12 to 14). At this point, average cost of mapping a VN, $C(M_{\text{avg}})$ for BLA is 38.88, for GNM is 45.04, whereas for HBNRM is 38.82 units of substrate resources. In this case, BLA and HBNRM's cost of mapping VNs is almost similar whereas GNM's mapping cost is the highest among three approaches. Overall BLA uses 0.62% more resources as compared to GNM and 1.47% when matched with HBNRM (Tables 12 to 14). However, BLA maps 3.5% more VNs as compared to GNM and 1.75% more when matched with HBNRM.

When a set of densely connected VNs (VN-Set 2) is mapped on S_3 then, the evaluated approaches do not give similar mapping results from initial set of VNs (Tables 15 to 17). Therefore, cost comparison for this set of VNs cannot be presented. Overall, BLA uses 10.54% less resources as compared to GNM, and 12.76% less when matched with HBNRM (Tables 15 to 17). However in this case, BLA maps 9.5% less VNs as compared to GNM (Tables 15 and 16) and 14.5% less when matched with HBNRM (Tables 15 and 17).

Three approaches map similar number of VNs when initial 50 VN requests arrive on S_4 (Tables 18 to 20). At this point, average cost of mapping a VN, $C(M_{\text{avg}})$ for BLA is 57.68, for GNM is 82.86, whereas for HBNRM is 59.94 units of substrate resources. So, in this case as well BLA's cost of mapping VNs is least whereas GNM's mapping cost is highest among three approaches.

Overall HBNRM uses 12.37% more resources as compared to BLA (Tables 18 and 20) and 3% more when matched with GNM (Tables 19 and 20). However in this case, HBNRM maps 11% more VNs as compared to BLA (Tables 18 and 20) and 7.75% more when matched with GNM (Tables 19 and 20).

(c) Bottleneck Nodes ($B(N)$). Nodes start to become bottleneck from VN interval-1 (VNs 1–50) when BLA is used for mapping on S_3 or S_4 (Tables 12, 15, and 18). In case of GNM there are no bottleneck nodes till the seventh interval for VN-Set 1 on S_3 (Table 13), and for VN-Set 2 no node becomes bottleneck (Table 16). When approach used is HBNRM then nodes start to become bottleneck after sixth interval in case of VN-Set 1 (Table 14). For HBNRM, nodes do not become bottleneck as long as nel value is set according to bottleneck limit (val_n) as defined in Section 4.1. However, according to condition (i) of 80/50 rule for nel values (Section 4.1) it needs to be decreased for seventh interval for VN-Set 1 (Table 14). In case of VN-Set 2 no node becomes bottleneck (Table 17). Less number of VNs gets mapped for VN-Set 2 as compared to VN-Set 1 for both GNM and HBNRM and therefore no nodes become bottleneck (Tables 16 and 17). According to condition (iii) of 80/50 rule for nel values (Section 4.1) it needs to be decreased for next interval for VN-Set 2 (Table 17) if more VN requests arrive as more than 50% VNs get dropped in eighth interval (Table 17).

In case of GNM there are no bottleneck nodes on S_4 (Table 19). When approach used is HBNRM then, also there are no bottleneck nodes (Table 20). However, according to condition (ii) of 80/50 rule for nel values (Section 4.1) nel value needs to be increased for fifth interval (Table 20). Number of mapped VNs is below 50% in the fourth interval (Table 20) and therefore nel value is increased to the next level for fifth interval and again 80/50 rule is applied on that value. Moreover, according to condition (iii) of 80/50 rule for nel values (Section 4.1) it needs to be decreased for next interval if more VN requests arrive as more than 50% VNs get dropped in eighth interval (Table 20).

(d) Exhausted Nodes ($E(N)$). When BLA is used, nodes start to exhaust from first interval on S_3 and S_4 (Tables 12, 15, and 18). In case of GNM no node resource exhausts on S_3 and S_4 (Tables 13, 16, and 19), similar trend can be seen for HBNRM where no node resource exhausts (Tables 14, 17, and 20).

Summary 1. When a substrate has higher number of resources and is densely connected, then the three approaches give almost similar results in terms of number of mapped VNs when either sparsely or densely connected VNs are mapped (Figures 3 and 4, Tables 3 to 8). Resources

by each approach are utilized in a different manner. BLA uses least whereas GNM uses highest number of resources (Figures 3 and 4, Tables 3 to 8). However, when the substrate is densely connected but has lesser number of resources and a set of densely connected VNs are mapped then BLA maps more VNs as compared to GNM (Figure 5, Tables 9 and 10).

On a sparsely connected substrate, when a set of sparsely connected VNs get mapped then also compared approaches give almost similar mapping results (Figure 6, Tables 12 to 14). However, when the substrate is sparsely connected but a set of densely connected VNs are mapped then GNM maps more VNs as compared to BLA (Figures 7 and 8, Tables 15 and 16, Tables 18 and 19).

The HBNRM approach is either close to or gives better VN mappings than compared approaches on either sparsely or densely connected substrates (Figures 3 to 8, Tables 3 to 20). The flexibility of 80/50 rule for nel values facilitates in doing better vertex mapping in changing scenarios and thus good mapping results can be achieved regardless of the type of substrate. HBNRM comes close to GNM in terms of minimizing complete exhaustion of node resources of a substrate, and also is near to BLA in terms of reducing mapping cost of VNs (Tables 3 to 20).

6. Conclusion and Future Work

We have proposed an approach to virtual network embedding which not only minimizes complete exhaustion of substrate nodes but also does that at the cost of utilizing comparatively less resources than an existing approach. Main focus of this approach is to do cost efficient mapping of vertices on those nodes of a substrate which after mapping, do not become bottleneck for future VN requests.

The proposed approach (referred to as HBNRM) has been compared with existing vertex mapping methods BLA and GNM. BLA does not take node resource exhaustion into consideration which GNM does but can map VNs at a higher cost. The number of virtual networks that can be assigned to a substrate has been investigated for varying distributions of VN requests and substrate topologies. The results show that BLA is favorable for densely connected substrates, while GNM gives better results for sparsely connected ones. HBNRM, on the other hand either gives almost similar or better VN mappings for both sparsely as well as densely connected substrates when compared with BLA and GNM.

One possible extension of this work is to include the feature of adaptability to either deal with change in user's demands after a VN gets mapped on the substrate or, handle node and link failures.

References

[1] J. S. Turner and D. E. Taylor, "Diversifying the Internet," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'05)*, pp. 755–760, December 2005.

[2] F. A. Shaikh, S. McClellan, M. Singh, and S. K. Chakravarthy, "End-to-end testing of IP QoS mechanisms," *Computer*, vol. 35, no. 5, pp. 80–87, 2002.

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

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

[5] A. Nakao, "Network virtualization as foundation for enabling new network architectures and applications," *IEICE Transactions on Communications*, vol. E93-B, no. 3, pp. 454–457, 2010.

[6] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI Veritas: realistic and controlled network experimentation," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '06)*, pp. 3–14, Pisa, Italy, 2006.

[7] http://www.geni.net/.

[8] Planetlab, http://www.planet-lab.org/.

[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, 2008.

[10] N. Farooq Butt, M. Chowdhury, and R. Boutaba, "Topology-awareness and reoptimization mechanism for virtual network embedding," *Lecture Notes in Computer Science*, vol. 6091, pp. 27–39, 2010.

[11] A. Razzaq, P. Sjödin, and M. Hidell, "Minimizing Bottleneck Nodes of a Substrate in Virtual Network Embedding," in *In Proceedings of the 2nd IFIP International Conference Network of the Future (NoF'11)*, Paris, France, November 2011.

[12] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '09)*, 2009.

[13] N. M. Mosharaf, K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *28th Conference on Computer Communications (INFOCOM '09)*, pp. 783–791, April 2009.

[14] J. Shamsi and M. Brockmeyer, "QoSMap: QoS aware mapping of virtual networks for resiliency and efficiency," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'07)*, November 2007.

[15] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, and L. Mathy, "Adaptive virtual network provisioning," in *Proceedings of the 2nd ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, pp. 41–48, ind, September 2010.

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

[17] J. Fan and M. H. Ammar, "Dynamic topology configuration in service overlay networks: a study of reconfiguration policies," in *25th IEEE International Conference on Computer Communications (INFOCOM '06)*, April 2006.

[18] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, April 2006.

[19] 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)*, pp. 3004–3008, December 2003.

[20] I. Houidi, W. Louati, and D. Zeghlache, "A distributed virtual network mapping algorithm," in *IEEE International Conference on Communications (ICC '08)*, pp. 5634–5640, May 2008.

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

[22] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[23] Y. Zhu, *Routing, resource allocation and network design for overlay networks [Ph.D. thesis]*, College of computing, Georgia Institute of Technology, 2006.

[24] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: an approach to universal topology generation," in *Proceedings of the 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '01)*, pp. 346–353, August 2001.

[25] A. Razzaq and M. S. Rathore, "An approach towards resource efficient virtual network embedding," in *Proceedings of the 2nd International Conference on Evolving Internet (Internet '10)*, pp. 68–73, esp, September 2010.