An Opportunistic Resource Sharing and Topology-Aware Mapping Framework for Virtual Networks

Sheng Zhang†, Zhuzhong Qian†, Jie Wu‡, and Sanglu Lu†
† State Key Lab. for Novel Software Technology, Nanjing University, China
‡ Department of Computer and Information Sciences, Temple University, USA
† zhangsheng@dislab.nju.edu.cn, {qzz,sanglu}@nju.edu.cn, ‡ jiewu@temple.edu

***Abstract*—Network virtualization provides a promising way to overcome Internet ossification. A major challenge is virtual network mapping, i.e., how to embed multiple virtual network requests with resource constraints into a substrate network, such that physical resources are utilized in an efficient and effective manner. Since this problem is known to be NP-complete, a variety of heuristic algorithms have been proposed. In this paper, we re-examine this problem and propose a virtual network mapping framework, *ORSTA*, which is based on Opportunistic Resource Sharing and Topology-Aware node ranking. Opportunistic resource sharing is taken into consideration at the entire network level for the first time and we develop an online approximation algorithm, *FFA*, for solving the corresponding time slot assignment problem. To measure the topology importance of a substrate node, a node ranking method, *MCRank*, based on Markov chain is presented. We also devise a simple and practical method to estimate the residual resource of a substrate node/link. Extensive simulation experiments demonstrate that the proposed framework enables the substrate network to achieve efficient physical resource utilization and to accept many more virtual network requests over time.**

***Index Terms*—virtual network mapping; opportunistic resource sharing; bin packing; topology-aware; markov chain**

I. Introduction

The Internet has been extremely successful in optimizing the way we exchange and process information. However, due to the competing policies and interests of its stakeholders and the ever-expanding scale of Internet use, the Internet has become resistant to fundamental changes [1, 2]. *Network virtualization* has been proposed as a promising approach that claims to overcome the current ossification of the Internet [1–4]. In a network virtualization environment, *infrastructure providers* (InPs) maintain *physical/substrate networks* (SNs), while *service providers* (SPs) purchase slices of physical resources (e.g., CPU, bandwidth, memory space, disk storage) from InPs and then create customized *virtual networks* (VNs) to offer their own value-added services to end users. This decoupling of traditional Internet service providers brings a layered service architecture, which provides flexibility and diversity to everyone.

One of the fundamental problems in network virtualization is the *virtual network embedding/mapping* (*VNE*) problem, i.e., how to embed multiple virtual network requests with resource constraints into a substrate network, such that physical resources are utilized in an efficient and effective manner. As this problem is proven to be NP-complete [5], many heuristic algorithms [6–15] have been proposed.

However, these early algorithms did not take workload fluctuation, e.g., Auckland Data Trace [16], into consideration. Most web-based service providers potentially target users all over the world, so it is extremely difficult to predict the workload before they are ready to serve end users. To cope with a peak workload on demand, service providers often over-purchase physical resources, which may lead to a considerable waste of resources for a normal workload. What is more, infrastructure providers may lose some upcoming customers due to inefficient resource utilization.

In this paper, we re-examine the virtual network mapping problem through two novel aspects, Opportunistic Resource Sharing and Topology-Aware node ranking, and we propose a novel framework, *ORSTA*, which provides efficient physical resource utilization and deployment.

Inspired by opportunistic spectrum access [17], we envision opportunistic resource sharing. Generally speaking, we model the workload in a virtual network as the combination of a basic sub-workload, which always exists, and a variable sub-workload, which occurs with some probability. Then, multiple variable sub-workloads from different virtual networks are allowed to share some common resources to achieve efficient resource utilization, while for a basic sub-workload, we allocate the corresponding, required resources as usual.

Furthermore, topology-awareness is considered due to the following fact: suppose that there are two substrate nodes with the same residual CPU resources, but the residual resources on their neighbors vary a lot. It is then reasonable to choose the substrate node with more resources in its proximity, so as to heuristically reduce the length of embedded virtual links. Hence, to facilitate the efficient embedding of virtual networks, we develop a Markov Chain-based substrate node ranking algorithm to measure the "importance" of each substrate node by assigning a numerical value.

The contributions of this paper are the following. (i) To the best of our knowledge, this is the first attempt that considers virtual network mapping in the context of opportunistic resource sharing at the entire network level. In our previous work [18], we only considered how to share bandwidth among

multiple virtual links in a single substrate link. (ii) We propose a formal formulation of opportunistic resource sharing-based time slot assignment and develop an online approximation solution, FFA. (iii) We take topology into consideration and design a Markov chain-based node ranking method, *MCRank*, to measure the "importance" of a substrate node. (iv) A simple and practical method for estimating the residual resource of a substrate node/link is devised. (v) Extensive simulations validate the effectiveness of $ORSTA$.

The remainder of this paper is organized as follows. Section II presents our motivation. We describe the assumptions, notations, and problem formulation in Section III. Then in Sections IV and V, we give an overview of $ORSTA$ and detailed algorithms, respectively. We evaluate $ORSTA$ using experiments in Section VI. Before concluding this paper in Section VIII, we survey related work in Section VII.

II. Motivation

SPs lease physical resources from InPs to create their own virtual networks and deploy services aimed at end users. As we mentioned before, SPs over-purchase resources to cater for potentially high workload rates, otherwise, customers will suffer bad experiences when SPs purchase no more than basic resources. In doing so, SPs may waste the additional purchased resources due to traffic fluctuation. From the perspective of an InP, the physical resources he owns are limited and constant in a relatively long period, so he may wish to make efficient use of his resources and accept more virtual networks, which allows for more revenue.

A motivational example is illustrated as follows. Suppose that an infrastructure provider, InP-A, owns a physical link with a capacity of $10MB/s$, and all of the other service providers, SP-1, SP-2 and SP-3, require a virtual link with a capacity of $4MB/s$. We assume that InP-A charges 1 dollar for $1MB/s$ bandwidth in unit time. In the case without opportunistic resource sharing, InP-A can accept only two requests among three because $4MB/s * 3 = 12MB/s > 10MB/s$. Therefore, InP-A gets 8 dollars per unit time in this case.

Let us take network traffic fluctuation into consideration and assume that each $4MB/s$ traffic flow is composed of a basic sub-flow $3MB/s$, which always exists, and a variable sub-flow $1MB/s$, which occurs with probability 0.1. Now, InP-A has an allocation method to accept all three virtual links: by letting three variable sub-flows share the same $1MB/s$ and allocating the residual $9MB/s$ to three basic sub-flows. We assume that flows in different virtual links are independent of each other, which is reasonable, thus the probability that more than two variable sub-flows occur (a collision) is:

$$1 - (0.9 * 0.9 * 0.9 + 3 * 0.9 * 0.9 * 0.1) = 0.109$$

For variable sub-flows, InP-A should decrease the charge, e.g., 0.1 US dollar for $1MB/s$ in unit time. Therefore, in this case with opportunistic resource sharing, InP-A can get $(3 + 0.1) * 3 = 9.3$ dollars, which is more than the original income; SP-1 or SP-2 or SP-3 just needs to pay 3+0.1=3.1 dollars, which is less than the previous charge of 4 dollars.

In general, although opportunistic resource sharing brings performance degradation when virtual networks encounter a peak workload, it enables better utilization of physical resources, increases the InPs' revenue and decreases the SPs' rent. We believe that opportunistic resource sharing can benefit all parties through reasonable pricing. We will not discuss how to set prices in this paper as it is out of the scope.

III. Problem Formulation

A. Assumptions

CPU and bandwidth are the main constraints we consider in this paper, which is the typical case in almost all of the related literatures [6–15] on virtual network mapping so far. To unify the resource notations, we assume that the substrate network is based on *time division multiplexing*, where time is partitioned into multiple frames of equal length, and each frame is further divided into L equal time slots, $ts_1, ts_2, ..., ts_L$. In this way, both CPU and bandwidth can be expressed in time slots. An SP only needs to specify the number of the time slots for a particular virtual node/link when he makes a request for virtual network mapping. Therefore, we will only focus on opportunistic bandwidth sharing in Section V-A; the results can be directly applied to opportunistic CPU sharing without any major changes.

SPs deploy end-to-end value-added services aimed at the end users, who access the services and leave at any time. These dynamic customers lead to workload fluctuations and further cause a waste of physical resources. As it is difficult to capture the characteristics of workload fluctuation, we use a simplified model: the work load wl_i of virtual network i is composed of a basic sub-workload $basic_i$, which exists all of the time, and a variable sub-workload $varia_i$, which occurs with a small probability pwl_i in unit time. We let $bwl_i = basic_i/wl_i$ represent basic sub-workload percentage. Workloads in different virtual networks are assumed to be independent of each other.

We believe that this model is reasonable although it is simple. For example, in a layered video streaming service [19], each video frame is encoded into multiple video packets corresponding to multiple quality layers. These video packets are classified as a base or enhancement layer to meet different network conditions. The workload in this scenario is very similar to our model. We hope that this simplified model can provide some insights on the design of other *VNE* algorithms.

Since both the network traffic and CPU time will fluctuate as the workload fluctuates, both the amounts of network flow in a virtual link and the CPU time in a virtual node are assumed to be proportional to the workload for simplicity. More specifically, the flow in a virtual link is composed of a basic part, whose percentage is bwl_i, and a variable part, which occurs with probability pwl_i. And the same statement holds for the CPU time.

B. Notations

Substrate Network: a substrate network is modeled as a weighted undirected graph, $G^s = (N^s, E^s, C^s, B^s)$, where N^s

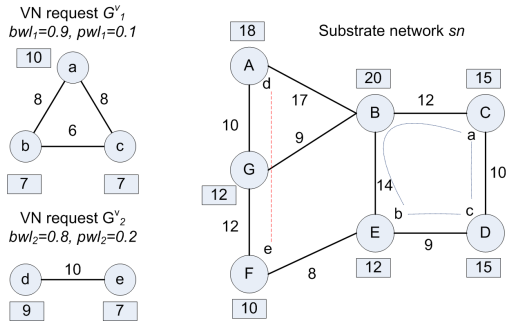

Fig. 1. Notations of virtual network embedding

is the set of substrate nodes, and E^s is the set of substrate links. C^s is the set of CPU capacity attributes, and B^s is the set of bandwidth capacity attributes. We use $RC^s(n^s)$ and $RB^s(e^s)$ to denote the residual CPU of substrate node n^s and residual bandwidth of substrate link e^s, respectively. Here, we would mention that the amount of residual resources in the presence of opportunistic resource sharing is not obvious, we will discuss it later when necessary. We use P^s to denote the set of loop-free paths in G^s.

Virtual Network: we denote a virtual network by a weighted undirected graph, $G^v_i = (N^v_i, E^v_i, C^v_i, B^v_i, bwl_i, pwl_i)$. N^v_i is the set of virtual nodes, and E^v_i is the set of virtual links. C^v_i is the set of CPU constraints, and B^v_i is the set of bandwidth constraints. bwl_i represents the percentage of a basic sub-workload in an overall workload . pwl_i means the probability of a variable sub-workload occurring in unit time.

The notations are summarized in Table I for reference. The notation system in this paper is similar to that in [8, 10, 20]. The principle behind these notations is that superscript indicates substrate/virtual networks.

Fig. 1 shows an example of these notations. The corresponding number near each vertex/edge is the CPU/bandwidth capacity/constraint. For virtual link $a-b$ in the VN request G^v_1, the basic sub-traffic is $8*0.9 = 7.2$, the variable sub-traffic is $8 - 7.2 = 0.8$ and occurs with probability 0.1.

Virtual Network Mapping: virtual network mapping is usually defined as a mapping $\mathcal{M}$ from G^v_i to a subset of G^s, such that some predefined constraints are satisfied. It can be decomposed into two major components: *node mapping* $\mathcal{M}_n$ and *link mapping* $\mathcal{M}_l$.

The node mapping $\mathcal{M}_n : N^v_i \rightarrow N^s$ maps a virtual node to a substrate node, subject to: $\forall n^v, m^v \in N^v_i$

$$\mathcal{M}_n(n^v) \in N^s$$
$$\mathcal{M}_n(n^v) = \mathcal{M}_n(m^v) \text{ iff } m^v = n^v$$
$$RC^s(\mathcal{M}_n(n^v)) \geq C^v_i(n^v)$$

The link mapping $\mathcal{M}_l : E^v_i \rightarrow P^s$ maps a virtual link to a substrate loop-free path, subject to: $\forall e^v = (m^v n^v) \in E^v_i$

$$\mathcal{M}_l(m^v n^v) \in P^S(\mathcal{M}_n(m^v), \mathcal{M}_n(n^v))$$
$$RB^s(\mathcal{M}_l(m^v n^v)) \geq B^v_i(e^v)$$

In Fig. 1, the node mapping for the VN request G^v_1 is $\{a \rightarrow C, b \rightarrow E, c \rightarrow D\}$, and the link mapping is $\{(ab) \rightarrow \{CB, BE\}, (bc) \rightarrow \{ED\}, (ca) \rightarrow \{DC\}\}$. The node mapping for the VN request G^v_2 is $\{d \rightarrow A, e \rightarrow F\}$, and the link mapping is $\{(de) \rightarrow \{AG, GF\}\}$.

TABLE I
NOTATIONS IN THIS PAPER

Notation	Meaning
G^s	Substrate network
N^s	Set of nodes
E^s	Set of links
C^s	Set of CPU capacity attributes
B^s	Set of bandwidth capacity attributes
RC^s	Set of residual CPU resources
RB^s	Set of residual bandwidth resources
P^s	Loop-free paths set
G^v_i	The i^{th} virtual network
N^v_i	Set of nodes of G^v_i
E^v_i	Set of links of G^v_i
C^v_i	Set of CPU constraints in G^v_i
B^v_i	Set of bandwidth constraints in G^v_i
bwl_i	A basic sub-workload percentage in G^v_i
pwl_i	Probability of a variable sub-workload in G^v_i

C. Objective

Virtual network mapping is done by InPs. From InPs' standpoint, a natural objective is to increase revenue and to decrease cost. However, an SP is only willing to pay fee proportional to his requested resources, that is, the revenue, $\mathcal{R}(G^v_i)$, of embedding a virtual network, G^v_i, can be defined as (following the definitions in previous works [8, 10]):

$$\mathcal{R}(G^v_i) = \omega_c \cdot \sum_{n^v \in N^v} C^v_i(n^v) + \omega_b \cdot \sum_{e^v \in E^v} B^v_i(e^v) \qquad (1)$$

where ω_c and ω_b are the weights for CPU and bandwidth, respectively. We learn from this definition that, given a VN request, the revenue is fixed. Therefore, in order to get more revenue, the InP should try to accept more VN requests but meet their resource constraints at the same time.

To this end, virtual networks should be properly and efficiently deployed on top of a substrate network. Due to the NP-completeness of the *VNE* problem [5], many heuristic algorithms have been proposed. However, in this paper we re-visit this problem from two novel aspects: opportunistic resource sharing and topology-aware node mapping.

IV. Framework Overview

We present a high level overview of our framework, *ORSTA*, as illustrated in Algorithm 1, in this section and the details of *ORSTA* in the next section.

Initially, we compute the amount of residual resources for each substrate node/link (lines 3-4 of Alg. 1), i.e., the available CPU/bandwidth capacity. Note that, when we perform opportunistic resource sharing, the estimation of residual resources becomes complicated; we will explain our method in Section V-B. Afterwards, inspired by the PageRank algorithm [21] which is used by Google to assign a numerical rank to every web page, we devise a similar ranking algorithm based on Markov chain [22] to compute the rank of each substrate node, denoted by *MCRank* (line 5 of Alg. 1), which is illustrated

in Section V-C. This *MCRank* takes both residual resources and topology into consideration in an effort to improve the deployment of virtual networks.

When a VN request arrives, we adopt the following simple greedy heuristic. In the node mapping phase (lines 7-10 of Alg. 1), all virtual nodes are sorted in a decreasing order of their CPU constraints and are placed in a queue; then, we map each virtual node in the sorted queue to the unused substrate node with the highest *MCRank*. In the link mapping phase (lines 11-13 of Alg. 1), we map each virtual link to the k-shortest path between its end hosts (for increasing k) to minimize the length of the substrate paths that virtual links are mapped to.

Then, considering workload fluctuation, we employ opportunistic resource sharing to efficiently utilize substrate resources at the expense of collisions (line 14 of Alg. 1). We formulate this opportunistic resource sharing-based local time slot assignment problem as an optimization problem and devise an online approximation algorithm, which is inspired by the bin packing problem [23]. The details are presented in section V-A.

When a virtual network request is successfully embedded, the framework *ORSTA* updates $RC^s(n^s)$, $RB^s(e^s)$ and $MCRank(n^s)$, and waits for another virtual network request.

Algorithm 1 The Opportunistic Resource Sharing and Topology-Aware Mapping Framework (*ORSTA*)

1: Initialization Phase
2: **while** true **do**
3: $\forall n^s \in N^s$, update $RC^s(n^s)$
4: $\forall e^s \in E^s$, update $RB^s(e^s)$
5: Run $MCRank(\gamma)$
6: Wait until a VN request, say G_i^v, arrives
7: $Q^s \leftarrow$ sorted virtual nodes in G_i^v according to C_i^v
8: **for** $i = 0$ to $Q^s.length$ **do**
9: Map $Q^s[i]$ to the unused substrate node with the highest *MCRank*
10: **end for**
11: **for all** $e^s \in E^s$ **do**
12: Map e^s to the k-shortest path for increasing k
13: **end for**
14: $\forall n^s \in N^s$ and $\forall e^s \in E^s$, run $FFA(p_{th})$
15: **end while**

V. Detailed Algorithms

A. Opportunistic Resource Sharing-based Local Time Slot Assignment

1) Preliminaries: in *ORSTA*, the virtual network embedding is completed after line 13 of Alg. 1 in a macroscopical sense, i.e., node-to-node and link-to-path matching is accomplished. However, at a single node/link level, we also need to deal with opportunistic resource sharing-based local time slots assignment. Note that both CPU and bandwidth can be represented as time slots, so we focus on assigning time slots in a substrate link among multiple virtual links. The results can be applied to a substrate node without any major changes.

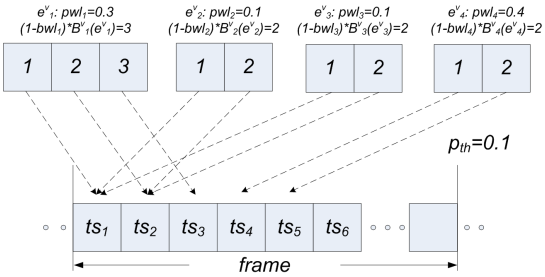

Fig. 2. Time slot assignment in a virtual link

Opportunistic resource sharing for virtual network mapping is proposed in [18] for the first time. In that paper, we studied opportunistic bandwidth sharing in a single physical link and devised two heuristic algorithms from different perspectives. Here, we employ the results from our previous work and improve them by developing an online approximation algorithm.

As we assumed, both the network traffic and the CPU busy time are proportional to the workload; for a virtual link, e_i^v, from virtual network G_i^v, the traffic is composed of basic sub-traffic, which equals $bwl_i \cdot B_i^v(e^v)$, and variable sub-traffic, which equals $(1 - bwl_i) \cdot B_i^v(e^v)$ and happens with probability pwl_i. For basic sub-traffic, the InP has no choice but to allocate the required number of time slots, thus we will only consider variable sub-traffic in the subsequent discussion.

To conserve time slots for upcoming requests, we prefer that each time slot can be assigned to as much variable sub-traffic as possible. However, when more than one variable sub-traffic occurs at the same time slot, a *collision* happens. To break the tradeoff between utilization and collision, we have the following optimization problem.

Problem 1: Given a probability threshold, p_{th}, and a set of n variable sub-traffic from e_i^v, $i = 1, 2, \ldots, n$, each requires $(1 - bwl_i) \cdot B_i^v(e_i^v)$ time slots with probability pwl_i. Find an assignment of time slots for the variable sub-traffic to minimize the number of time slots used, such that: 1) for each variable sub-traffic flow from e_i^v, the number of time slots assigned to it is at least $(1-bwl_i) \cdot B_i^v(e_i^v)$; 2) the collision probability at each time slot is no more than p_{th}.

The collision probability can be obtained as follows. Let X_i indicate whether variable sub-traffic in e_i^v occurs, i.e., $Pr[X_i = 1] = pwl_i$. For time slot k, let D_k denote the set of variable sub-traffic that it is assigned to. Then, the probability of a collision happening at slot k, denoted by $Pr_{collision}(D_k)$, is:

$$\begin{aligned} Pr_{collision}(D_k) &= Pr[\sum_{i \in D_k} X_i \geq 1] \\ &= 1 - \prod_{i \in D_k}(1 - pwl_i) - \sum_{i \in D_k}(pwl_i \cdot \prod_{j \in D_k, j \neq i}(1 - pwl_j)) \end{aligned} \tag{2}$$

Fig. 2 shows a feasible assignment. ts_1 can be assigned to three sub-traffic flows from e_1^v, e_2^v, and e_3^v because they collide with a probability 0.064 (by Eq. (2)), which is less then $p_{th} = 0.1$. ts_3 can not be assigned to e_1^v and e_4^v simultaneously because the collision probability is $0.3 \cdot 0.4 = 0.12 > p_{th}$.

2) A first-fit-based online approximation algorithm: the design of our algorithm is based on the following observation: when each variable sub-traffic requires one single time slot, i.e., $(1 - bwl_i) \cdot B_i^v(e_i^v) = 1$ for all i, Problem 1 is very similar to *bin packing*[1]. However, the occupied size in each bin is the sum of the sizes of all of the packed items in bin packing, whereas in Problem 1, the collision probability in a time slot, as shown in Eq. 2, is neither linear nor multiplicative.

First-fit [23] is a heuristic algorithm with an approximation factor of 2 for bin packing. In first-fit, items are considered in an arbitrary order, and for each item, first-fit attempts to place the item in the first bin that can accommodate the item. If not, the item is placed into a new bin. First-fit can be executed online and has a low time complexity.

Their resemblance sheds light on the design of *FFA*, which is based on the core idea of first-fit. The *FFA* Algorithm is presented in Algorithm 2.

Algorithm 2 $FFA(p_{th})$

1: Wait until variable sub-traffic from e_i^v arrives
2: $counter \leftarrow 0$
3: **While**$(counter < (1 - bwl_i) \cdot B_i^v(e_i^v))$
4: Let $pos \leftarrow 0$
5: **While**$(Collision(ts_{pos}, e_i^v) > p_{th})$
6: $pos \leftarrow pos + 1$
7: Assign ts_{pos} to e_i^v
8: $counter \leftarrow counter + 1$

3) Incremental Calculation: $Collision(ts_{pos}, e_i^v)$ is a function that returns the probability of a collision at ts_{pos} if ts_{pos} is assigned to e_i^v. To calculate the collision probability efficiently, we adopt the following incremental approach. Let D_k be the set of variable sub-traffic that ts_k is currently assigned to. Also let:

$$A(D_k) = \prod_{i \in D_k} (1 - pwl_i)$$

$$B(D_k) = \sum_{i \in D_k} (pwl_i \cdot \prod_{j \in D_k,\ j \neq i} (1 - pwl_j))$$

then, the collision probability $Pr_{collision}(D_k) = 1 - A(D_k) - B(D_k)$. When slot k is to be assigned to a new sub-traffic from e_h^v, then:

$$\begin{aligned} A(D_k \cup \{e_h^v\}) &= A(D_k)(1 - pwl_h) \\ B(D_k \cup \{e_h^v\}) &= B(D_k)(1 - pwl_h) + A(D_k) \cdot pwl_h \end{aligned} \tag{3}$$

This can be used to calculate the new collision probability.

4) Approximation Ratio: we denote by S_{ffa} the solution generated by *FFA*, and by S_{opt}, the optimal solution. Abusing the notation a bit, we will also use S_{ffa} and S_{opt} to denote the number of time slots needed by these two solutions, respectively, if no confusion can be caused. Let,

$$p_{min} = min_{1 \le i \le n} pwl_i,\ d_{min} = min_{1 \le i \le n}((1 - bwl_i) \cdot B_i^v(e_i^v))$$
$$p_{max} = max_{1 \le i \le n} pwl_i,\ d_{max} = max_{1 \le i \le n}((1 - bwl_i) \cdot B_i^v(e_i^v))$$

[1]Bin packing [23]: given n items with sizes $s_1, s_2, ..., s_n \in (0, 1]$, find a packing method in unit-sized bins that minimizes the number of bins used.

Case I: we use a particular variable sub-traffic flow, which requires d_{min} time slots and occurs with probability p_{min}, to replace all of the variable sub-traffic. Then, the maximal allowable number of sub-traffic, vol_I, in a substrate slot is determined by:

$$1 - (1 - p_{min})^{vol_I} - vol_I \cdot p_{min} \cdot (1 - p_{min})^{vol_I - 1} = p_{th}$$

Hence, in this case, the number of time slots required by all of the n sub-traffic is:

$$S_I = \frac{n}{vol_I} \cdot d_{min}$$

Case II: we use another particular variable sub-traffic flow, which requires d_{max} time slots and occurs with probability p_{max}, to replace all of the variable sub-traffic. Then, the maximal allowable number of sub-traffic, vol_{II}, in a substrate slot is determined by:

$$1 - (1 - p_{max})^{vol_{II}} - vol_{II} \cdot p_{max} \cdot (1 - p_{max})^{vol_{II} - 1} = p_{th}$$

Similarly, the number of time slots required by all of the n sub-traffic in this case is:

$$S_{II} = \frac{n}{vol_{II}} \cdot d_{max}$$

***Theorem** 1:* $S_{ffa} \le (d_{max} \cdot vol_I)/(d_{min} \cdot vol_{II}) \cdot S_{opt}$

Proof: it is straightforward to see that:

$$0 < S_I \le S_{opt} \le S_{ffa} \le S_{II}$$

then:

$$\frac{S_{ffa}}{S_{opt}} \le \frac{S_{II}}{S_I} = \frac{d_{max} \cdot vol_I}{d_{min} \cdot vol_{II}}$$

The theorem follows immediately. ■

B. Residual Resource Estimation

The amount of the residual resource in a substrate node/link is an important metric in *VNE* process. In a traditional scenario, where opportunistic resource sharing is not considered, the residual CPU, $RC^s(n^s)$, of a substrate node, n^s, and residual bandwidth, $RB^s(e^s)$, of a substrate link, e^s, are defined as:

$$RC^s(n^s) = C^s(n^s) - \sum_{\forall n^v} f_c(n^v, n^s)$$
$$RB^s(e^s) = B^s(e^s) - \sum_{\forall e^v} f_b(e^v, e^s)$$

where $f_c(n^v, n^s)$ denotes the CPU resources of n^s allocated to n^v, and $f_b(e^v, e^s)$ denotes the bandwidth resources of e^s allocated to e^v.

However, with opportunistic resource sharing, the residual resource becomes complicated. Fig. 3 shows a snapshot of the time slot allocation in a substrate link. Intuitively, the residual resource consists of the right part and the available room in the middle part which is allocated to variable sub-traffic. The former is easy to compute while the latter is not so obvious.

Suppose that the capacity of a substrate link, e^s, is $B^s(e^s)$, and there are L time slots in a frame. There are b slots allocated to basic sub-traffic and another d slots allocated to variable

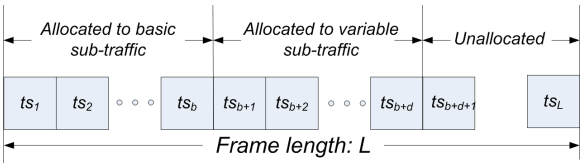

Fig. 3. A snapshot of time slot allocation in a substrate link

sub-traffic. Therefore, $RB^s(e^s)$ is $(L-b-d)/L \cdot B^s(e^s)$ plus the available room in the middle d slots.

A simple and practical measurement of residual room, rb_k, in a single time slot, ts_k $(b < k < b+d+1)$, is defined as the probability of a variable sub-traffic flow that would cause the collision probability to be higher than p_{th} if ts_k is assigned to it. Hence, by Eq. (3), we have:

$$1 - A(D_k)(1 - rb_k) - (B(D_k)(1 - rb_k) + A(D_k) \cdot rb_k) = p_{th}$$

That is:

$$rb_k = \frac{A(D_k) + B(D_k) + p_{th} - 1}{B(D_k)} = \frac{p_{th} - Pr_{collision}(D_k)}{B(D_k)} \quad (4)$$

Thereby, the available room in the middle d slots that are allocated to variable sub-traffic is:

$$\frac{B^s(e^s)}{L} \cdot \sum_{k=b+1}^{b+d} rb_k$$

In summary:

$$\begin{aligned} RB^s(e^s) &= \frac{L-b-d}{L} \cdot B^s(e^s) + \frac{B^s(e^s)}{L} \cdot \sum_{k=b+1}^{b+d} rb_k \\ &= (L - b - d + \sum_{k=b+1}^{b+d} \frac{p_{th} - Pr_{collision}(D_k)}{B(D_k)}) \frac{B^s(e^s)}{L} \end{aligned} \quad (5)$$

$RC^s(n^s)$ can be analyzed in a similar way.

C. Markov Chain-based Node Ranking (MCRank)

1) Preliminaries: PageRank [21] nicely reflects the popularity and quality of web pages. In PageRank, the rank of a page measures the "importance" of that page. A page has a higher rank if it is pointed to by more highly-ranked pages. The more pages that one page points to, the less its influence on their rankings is. This intuitive idea can be formalized as follows. The World-Wide-Web is treated as a directed graph, with web pages as vertices and hyper-links as directed edges. The rank of a page p, denoted by $rank(p)$, is supposed to satisfy:

$$rank(p) = \sum_{g:(g,p)\in E} \frac{rank(g)}{d_+(g)} \quad (6)$$

where $d_+(g)$ means the number of edges going out of page g. Note that the sum is over edges going into p.

Inspired by PageRank, we consider using a similar method to measure the topology importance of a substrate node. To understand the topology importance, suppose that there are two substrate nodes with the same residual resources, but the resources of their neighbors (or neighbors of neighbors) vary greatly. In order to heuristically decrease the length of virtual links (substrate paths), we prefer the substrate node with more resources in its vicinity. Therefore, we propose in the following, a Markov chain-based node ranking method, *MCRank*.

2) MCRank: we define the resource, $R(n^s)$, of a substrate node, n^s, as the product of its residual CPU resource and a half of the collective bandwidth resources of its outgoing links:

$$R(n^s) = RC^s(n^s) \cdot \sum_{e^s \in NeighE(n^s)} \frac{1}{2} RB^s(e^s)$$

where $NeighE(n^s)$ denotes the set of links that are adjacent to n^s. The coefficient $1/2$ can be seen as the bandwidth of a link is equally shared between its endpoints, though it is not in fact. We then normalize $R^s(n^s)$.

$$NormR(n^s) = \frac{R^s(n^s)}{\sum_{m^s \in N^s} R^s(m^s)} \quad (7)$$

Intuitively, if the $MCRank(n^s)$ of a substrate node, n^s, is high, the possible reasons are: (i) the node itself has ample resources; (ii) its neighbors have abundant resources. Hence, we define the $MCRank(n^s)$ as:

$$\begin{aligned} MCRank(n^s) &= w_1 \cdot MCRank(n^s) \\ &+ w_2 \cdot \sum_{m^s \in NeighN(n^s)} \frac{NormR(n^s)}{\sum_{h^s \in NeighN(m^s)} NormR(h^s)} \cdot MCRank(m^s) \end{aligned} \quad (8)$$

where w_1 and w_2 are the corresponding weights. $NormR(n^s)/\sum_{h^s \in NeighN(m^s)} NormR(h^s)$ can be seen as the impact of m^s on n^s. Let P be a matrix with rows and columns corresponding to substrate nodes, and $\forall n^s, m^s \in N^s$:

$$P(m^s, n^s) = \begin{cases} w_1 & \text{if } m^s = n^s \\ w_2 \cdot \frac{NormR(n^s)}{\sum_{h^s \in NeighN(m^s)} NormR(h^s)} & \text{if } (m^s, n^s) \in E^s \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

Then, Eq. (8) can be expressed in the following matrix form:

$$MCRank \cdot P = MCRank \quad (10)$$

Let $w_1 + w_2 = 1$; it is easy to verify that P is stochastic, i.e.:

$$\sum_{n^s \in N^s} P(m^s, n^s) = P(m^s, m^s) + \sum_{n^s \in NeighN(m^s)} P(m^s, n^s) = 1$$

Hence, *MCRank* is actually a stationary distribution of the Markov chain with transition matrix P. The following theorem gives the existence of a stationary distribution.

***Theorem** 2:* the Markov chain, determined by P, has a stationary distribution.

Proof: it is sufficient to prove that: (i) *MCRank* has finite states, as there are a finite number of substrate nodes; (ii) the substrate network is strongly connected, so the Markov chain is irreducible; (iii) the substrate node iteslf has an impact on its *MCRank*, i.e., the chain is a lazy random walk, so it is aperiodic. The theorem follows immediately. ■

Based on [21], we give our algorithm to calculate the stationary distribution, i.e., *MCRank*, in Algorithm 3, where γ serves as a threshold to control iterations.

Algorithm 3 *MCRank*(γ)

1: $MCRank \leftarrow NormR$, $i \leftarrow 0$
2: **repeat**
3: $\quad MCRank_{i+1} \leftarrow MCRank_i \cdot P$
4: $\quad \delta \leftarrow \|MCRank_{i+1} - MCRank_i\|_1$
5: $\quad i \leftarrow i + 1$
6: **until** $\delta < \gamma$

TABLE II
PARAMETERS

$E[C^v]$	Expectation of CPU requirement on a virtual node
$E[B^v]$	Expectation of bandwidth requirement on a virtual link
$E[bwl]$	Expectation of a basic sub-workload percentage
$E[pwl]$	Expectation of a variable sub-workload occurring probability

VI. Experimental Evaluation

In this section, we first describe our evaluation environment, then present main evaluation results.

A. Evaluation Environment

As network virtualization is still an open field, our evaluations follow settings similar to [7, 8, 10–12, 20]. We use ANSNET and ARPANET as the substrate network topologies. Both the CPU capacity at substrate nodes and bandwidth capacity at substrate links are generated uniformly from [50,100]. The arrivals of VN requests are modeled as a Poisson process with an average rate of 5 requests per minute. For each virtual network, the number of nodes is determined by a uniform distribution between 2 and 10; each pair of virtual nodes is connected with probability 0.5. The lifetime of each virtual network is assumed to be exponentially distributed with an average of 10 minutes. The collision probability threshold is set to 0.1 throughout our evaluation. We summarize the parameters that we vary in our evaluations in Table II.

Comparing our framework with previous research on virtual network mapping is difficult because it is the first attempt that considers virtual network mapping in the context of opportunistic resource sharing at the entire network level. Therefore, our evaluation focuses primarily on quantifying the benefits of opportunistic resource sharing and topology-aware node ranking. The notations that we use to refer to different algorithms are enumerated in Table III. In the following subsection, we present the average results after running over ANSNET 100 times. (Results over ARPANET are similar and omitted due to space limitation.)

B. Evaluation Results

We first present the comparison results between $ORSTA$, TA, ORS, and *Greedy* (refer to Table III). Figs 4, 5, and 6 show the acceptance ratio over time, CDF (*cumulative distribution function*) of node utilization ratios, and link utilization ratios, respectively. In these experiments, both $E[C^v]$ and $E[B^v]$ are set to be 10; $E[bwl] = 0.5$; $E[pwl] = 0.15$, and $w_1 = 0.7$. In Fig. 4, the acceptance ratio of $ORSTA$ is higher than all of the other algorithms, which indicates that opportunistic resource sharing and topology-aware node ranking indeed improves the deployment of virtual networks and further enables the substrate network to accept more VN requests. We also notice that the acceptance ratio of these four algorithms is averagely less than 0.4, which is a little low. The main reason behind this may be because links in the substrate network (ANSNET has 32 nodes and 58 links, ARPANET has 20 nodes and 32 links) are sparse, while each pair of nodes in a virtual network is connected with probability 0.5. This leads to the topology becoming the dominating factor in virtual network embedding.

Fig. 4 also shows that the acceptance ratio of TA is much higher than ORS and *Greedy*, while ORS achieves nearly the same acceptance ratio as *Greedy*. This implies that distinguishing between substrate nodes is helpful in virtual network mapping while opportunistic resource sharing is not significantly effective when there is no topology-aware node ranking. In Figs 5 and 6, the node/link utilization ratio means the ratio of the allocated resources to the capacity of a substrate node/link. We observe similar results where $ORSTA$ performs better than TA, ORS, and *Greedy*.

We then try to evaluate the effects of w_1, $E[C^v]$, $E[B^v]$, $E[bwl]$, and $E[pwl]$, respectively. Fig. 7 shows the comparison between $ORSTA$s with different w_1 settings. We see that $ORSTA(w_1 = 0.5)$ achieves the highest acceptance ratio while $ORSTA(w_1 = 0.3)$ gets the lowest. This is anticipated because in *MCRank*, "$w_1 = 0.5$" indicates the rank of a substrate node is half determined by itself; "$w_1 = 0.3$" causes the rank of a substrate node to be greatly dominated by the amount of residual resources of other nodes. However, the difference is not considerably significant, which means the stationary distribution of our Markov chain nicely represents the importance of substrate nodes, irrespective of whether w_1 is large or small.

In Fig. 8, we show the results of $E[C^v]$ and $E[B^v]$. We note that in the case when $E[C^v]$ and $E[B^v]$ are small, the acceptance ratio is high. However, with $E[C^v]$ and $E[B^v]$ increasing, the substrate network resources become scarce, which causes more and more VN requests to be rejected. In this figure, $ORSTA(E[C^v] = E[B^v] = 15)$ achieves almost the same acceptance ratio as $ORSTA(E[C^v] = E[B^v] = 20)$. A reasonable explanation is that, $E[C^v] = E[B^v] = 15$ is sufficiently large compared to the average amount of CPU/bandwidth capacities of substrate nodes/links, i.e., 75.

Fig. 9 illustrates the effects of $E[bwl]$ and $E[pwl]$. Remember that *bwl* is the percentage of a basic sub-workload in the overall workload, and *pwl* is the probability of a variable sub-workload happening. In this figure, $ORSTA(E[bwl] = 0.30, E[pwl] = 0.15)$ presents the best performance, and $ORSTA(E[bwl] = 0.50, E[pwl] = 0.05)$ achieves the second best. This is to say, *bwl* plays a more important role than *pwl*.

In summary, our proposed framework, $ORSTA$, increases the acceptance ratio through opportunistic resource sharing, which enables the substrate network to support more VN requests and topology-aware node ranking, which treats substrate nodes differently to shorten the length of substrate paths that virtual links are mapped to.

TABLE III
ALGORITHMS IN COMPARISON

Notation	Description
ORSTA	The entire framework, including opportunistic resource sharing, topology-aware ranking, and greedy node/link mapping
ORS	A partial framework, including opportunistic resource sharing, and greedy node/link mapping
TA	A partial framework, including topology-awareness, and greedy node/link mapping
Greedy	The traditional greedy embedding algorithm, including greedy node/link mapping

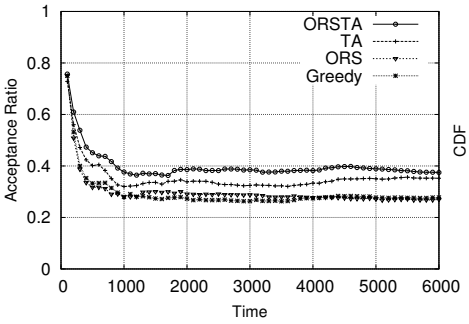

Fig. 4. Acceptance ratio over time

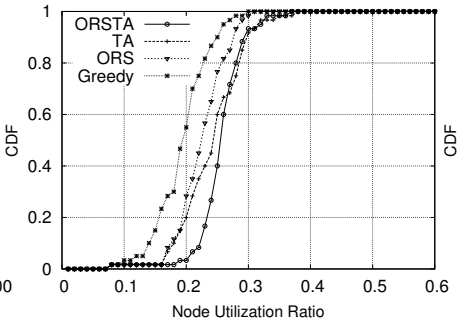

Fig. 5. CDF of node utilization ratios

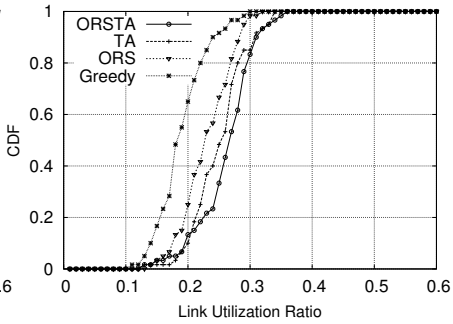

Fig. 6. CDF of link utilization ratios

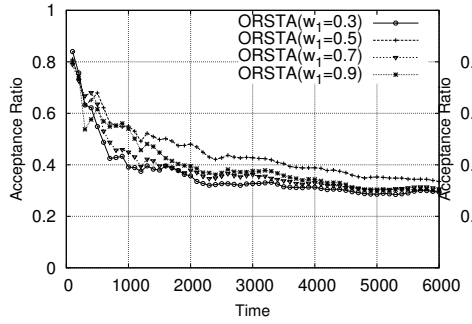

Fig. 7. Effect of w_1

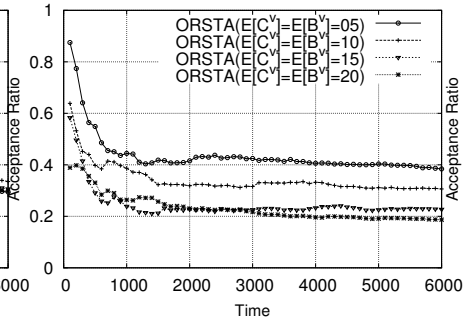

Fig. 8. Effect of $E[C^v]$ and $E[B^v]$

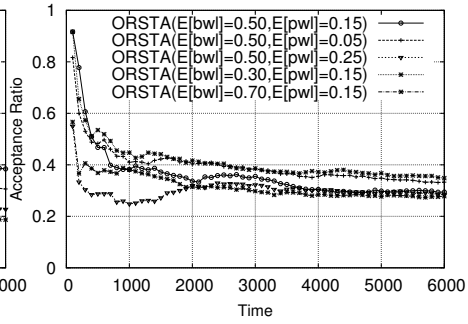

Fig. 9. Effect of $E[bwl]$ and $E[pwl]$

VII. RELATED WORK

A. Network Virtualization

In networking literature, *virtual private networks* (VPNs), *overlay networks*, and network virtualization deal with selecting nodes to construct logical paths. However, the differences are: (i) VPNs only need to determine the locations of logical links while network virtualizaiton simultaneously considers locations of logical nodes and links [7, 8]; (ii) only link constraints are considered in VPNs while both link and node constraints are considered in network virtualization [7, 8]; (iii) overlays are designed in the application layer on top of the network layer [4]. The most relevant work in the overlay networks is [24], where Fan et al. investigated the dynamic topology configuration problem in service overlay networks. They first acquired general properties of the optimal reconfiguration topology by observing small cases, and then used observations as heuristics to design algorithms for large-scale cases of networks.

In network virtualization, Ishibashi et al. [25] envisioned cognitive radio-based virtual wireless networks and tried to make efficient use of residual wasted bandwidth of the primary service providers. Shiomoto et al. [26] developed a network virtualization method to represent the resources in the optical backbone network of the service network. Zhu et al. [27] tried to lower the barrier for deploying wide-area services through introducing the *connectivity* layer, which, in fact, is a special virtual network that provides a necessary geographic footprint, reliability, and performance. He et al. [28] proposed *DaVinci* architecture, where resource allocation is dynamic and used optimization theory to show that adaptive resource allocation can be stable and can maximize the aggregate performance across virtual networks.

B. Virtual Network Embedding

To cope with the NP-hardness of the *VNE* problem, researchers resorted to heuristics to reduce computational time.

Ricci et al. [6] considered only bandwidth constraints and proposed the *Assign* algorithm based on simulated annealing with the assumption that the substrate node can only be used by one virtual node. Lu et al. [9] developed a method for the *VNE* problem in a cost-efficient way and attempted to find the best topology in a family of backbone-star topologies. Zhu et al. [7] assumed that the substrate nodes and links have unlimited CPU and bandwidth resources and focused on load balancing and on-demand assignments. Yu et al. [10] assumed that SN supports path splitting and proposed a two-stage online algorithm: firstly, map the virtual nodes greedily, then deal with link mapping based on the multi-commodity flow problem. Lischka et al. [11] proposed a backtracking algorithm based on subgraph isomorphism detection, but restricted the length of the substrate paths. Chowdhury et al. [8]

proposed a *VNE* algorithm with better coordination between node mapping and link mapping based on linear programming and deterministic/randomized rounding, but added location constraints to simplify the problem. Chowdhury et al. [13] presented a policy-based decentralized inter-domain virtual network embedding framework and also designed a location-aware VN request forwarding mechanism. Zhang et al. [20] focused on flexibility and proposed a simulated annealing-based algorithm to control the tradeoff between result accuracy and the running time of the embedding algorithm. Zhang et al. [18] investigated the opportunistic bandwidth sharing in a single physical link among multiple virtual links from different VNs; however, in this paper, we study opportunistic resource sharing at the entire network level.

C. Topology-Awareness in Virtual Network Mapping

To the best of our knowledge, there are only two research articles [12, 14] that incorporate topology into virtual network embedding. Butt et al. [14] differentiated substrate resources by introducing scaling factors: *Critical Index*, which measures the likelihood of a residual substrate network partition due to its unavailability, and *Popularity Index*, which measures "how many different VNs are affected when a link or a node is unavailable". Cheng et al. [12] formulated a Markov random walk model to compute the topology-aware resource ranking of nodes in a substrate network; however, in this paper, we observe that we only need to consider a walk from a substrate node to itself or one of its neighbors because the impact is recursive in a Markov chain.

VIII. Conclusions

This paper re-examines the virtual network mapping problem in network virtualization from two novel perspectives, i.e., opportunistic resource sharing and topology-aware node ranking, which are incorporated into our proposed framework, *ORSTA*. *ORSTA* contains three main parts, opportunistic resource sharing-based local time slot assignment, estimation of residual resources and topology-aware node ranking. The effectiveness of our framework is confirmed by extensive simulations. We are currently seeking the proof of the NP-Completeness of Problem 1 and are exploring methods of scheduling packets when a collision occurs in a time slot.

Acknowledgments

We would like to thank Yitong Yin for his *Randomized Algorithms* and *Combinatorics* courses, and the anonymous reviewers for their insightful suggestion.

This work is supported in part by the National NSF of China under Grant No. 61073028 and No. 61021062; Key Project of Jiangsu Research Program under Grant No.BE2010179; Jiangsu Natural Science Foundation under Grant No. BK2011510; the National 973 Basic Research Program of China under Grant No. 2009CB320705 and No. 2011CB302800; US NSF grants ECCS 1128209, CNS 1065444, CCF 1028167, CNS 0948184, and CCF 0830289.

References

[1] J. Turner and D. Taylor, "Diversifying the Internet," in *IEEE GLOBECOM 2005*, vol. 2, 2-2 2005, pp. 6 pp. –760.

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

[3] N. Feamster, L.-X. Gao, and J. Rexford, "How to lease the Internet in your spare time," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 61–64, 2007.

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

[5] D. G. Andersen, "Theoretical approaches to node assignment," Dec. 2002, unpublished Manuscript.

[6] R. Ricci, C. Alfeld, and J. Lepreau, "A solver for the network testbed mapping problem," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 65–81, 2003.

[7] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *IEEE INFOCOM 2006*, april 2006, pp. 1 –12.

[8] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *IEEE INFOCOM 2009*, 19-25 2009, pp. 783 –791.

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

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

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

[12] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 38–47, April 2011.

[13] M. Chowdhury, F. Samuel, and R. Boutaba, "Polyvine: policy-based virtual network embedding across multiple domains," in *ACM SIGCOMM VISA 2010*. New York, NY, USA: ACM, 2010, pp. 49–56.

[14] N. Butt, N. Chowdhury, and R. Boutaba, "Topology-awareness and reoptimization mechanism for virtual network embedding," in *Networking*, 2010, pp. 27–39.

[15] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, and L. Mathy, "Adaptive virtual network provisioning," in *ACM SIGCOMM VISA 2010*. ACM, 2010, pp. 41–48.

[16] *Auckland Data Trace*. http://pma.nlanr.net/traces/long/auck2.html.

[17] Q. Zhao and B. Sadler, "A survey of dynamic spectrum access," *IEEE Signal Processing Magazine,*, vol. 24, no. 3, pp. 79–89, May 2007.

[18] S. Zhang, Z. Qian, B. Tang, J. Wu, and S. Lu, "Opportunistic bandwidth sharing for virtual network mapping," in *IEEE Globecom 2011*, December 2011.

[19] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h.264/avc standard," *IEEE TCSVT*, vol. 17, no. 9, pp. 1103 –1120, sept. 2007.

[20] S. Zhang, Z. Qian, S. Guo, and S. Lu, "FELL: A flexible virtual network embedding algorithm with guaranteed load balancing," in *IEEE ICC 2011*, June 2011.

[21] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Technical Report 1999-66, November 1999.

[22] "Markov chain," *http://en.wikipedia.org/wiki/Markov_chain*.

[23] V. V. Vazirani, *Approximation Algorithms*. Berlin: Springer, 2003.

[24] J. Fan and M. H. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies," in *IEEE INFOCOM 2006*, april 2006, pp. 1 –12.

[25] B. Ishibashi, N. Bouabdallah, and R. Boutaba, "Qos performance analysis of cognitive radio-based virtual wireless networks," in *IEEE INFOCOM 2008*, april 2008, pp. 2423 –2431.

[26] K. Shiomoto, I. Inoue, and E. Oki, "Network virtualization in high-speed huge-bandwidth optical circuit switching network," in *IEEE INFOCOM 2008 Workshops*, april 2008, pp. 1 –6.

[27] Y. Zhu, R. Zhang-Shen, S. Rangarajan, and J. Rexford, "Cabernet: connectivity architecture for better network services," in *ACM CoNEXT 2008*. New York, NY, USA: ACM, 2008, pp. 64:1–64:6.

[28] J. He, R. Zhang-Shen, Y. Li, C.-Y. Lee, J. Rexford, and M. Chiang, "Davinci: dynamically adaptive virtual networks for a customized internet," in *ACM CoNEXT 2008*. ACM, 2008, pp. 15:1–15:12.