Virtual Network Embedding with Opportunistic Resource Sharing

Sheng Zhang, *Student Member, IEEE,* Zhuzhong Qian, *Member, IEEE,* Jie Wu, *Fellow, IEEE,* Sanglu Lu, *Member, IEEE,* and Leah Epstein

Abstract—*Network virtualization* has emerged as a promising approach to overcome the ossification of the Internet. A major challenge in network virtualization is the so-called *virtual network embedding* problem, which deals with the efficient embedding of virtual networks with resource constraints into a shared substrate network. A number of heuristics have been proposed to cope with the NP-hardness of this problem; however, all of the existing proposals reserve fixed resources throughout the entire lifetime of a virtual network. In this paper, we re-examine this problem with the position that time-varying resource requirements of virtual networks should be taken into consideration, and we present an *opportunistic resource sharing*-based mapping framework, ORS, where substrate resources are opportunistically shared among multiple virtual networks. We formulate the time slot assignment as an optimization problem, then we prove the decision version of the problem to be NP-hard in the strong sense. Observing the resemblance between our problem and the bin packing problem, we adopt the core idea of first-fit, and propose two practical solutions: *first-fit by collision probability* (CFF) and *first-fit by expectation of indicators' sum* (EFF). Simulation results show that that ORS provides a more efficient utilization of substrate resources than two state-of-the-art fixed-resource embedding schemes.

Index Terms—Virtual network embedding, opportunistic resource sharing, NP-hard, 3-partition, bin packing.

1 Introduction

The Internet has been extremely successful in supporting global commerce, communication, and defense [1], [2]. However, the multi-provider nature of the Internet and end-to-end design of Internet Protocol (IP) are now creating hurdles for the further evolution of the Internet. *Network virtualization* has been proposed recently as a promising approach to overcome the current ossification of the Internet [2], [3], [4], and it has been investigated in several projects, including CABO [3], PlanetLab [5], and VINI [6].

In a network virtualization environment, an *infrastructure provider* (InP) maintains a *physical/substrate network* (SN), which is composed of substrate nodes and links; a *service provider* (SP) leases physical resources (e.g., CPU, bandwidth, memory space) from InPs and creates customized *virtual networks* (VNs) to provide value-added services (e.g., video conferencing, VoIP, content distribution) for end users. Network virtualization has some desirable properties. First, the separation of the control and data tiers makes the network core programmable and flexible [7]. Second, physical resources can be used more efficiently, and thus high energy efficiency can be achieved.

The fundamental challenge that network virtualization faces is how to embed multiple virtual networks with resource constraints into a substrate network, so as to efficiently utilize substrate resources. Known as the *Virtual Network Embedding* (VNE) problem, it is proven to be NP-complete by reducing the *multiway separator problem* to this problem [8]; therefore, a number of heuristics [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19] have been proposed.

Unfortunately, all of the prior proposals reserve fixed resources throughout the entire lifetime of a virtual network, which wastes the precious substrate resources. First, SPs potentially target users all over the world, so it is extremely difficult to predict the workload before they are ready to serve end users. As the resource requirement of a VN at a particular time is generally proportional to the workload at that time, to cope with a peak workload on demand, service providers often over-purchase substrate resources, which may lead to a considerable waste of resources for a normal workload. Second, the resource requirements of many applications experience significant changes over time [20]. Given these two factors, provisioning fixed resources for virtual networks throughout their lifetimes is clearly wasteful.

In this paper, we exploit this key observation, and propose a novel model that reflects the time-varying resource requirement of a VN. More specifically, we model the resource requirement of a VN as the combination of a basic sub-requirement, which exists throughout the lifetime of the VN, and a variable sub-requirement, which occurs with a probability. Based on this model, this paper designs an *Opportunistic Resource Sharing*-based embedding framework, *ORS* [21], which in general consists of two components, i.e., the macro-level node-to-node/link-to-path embedding, and the micro-level time slot assignment. In the macro-level embedding, we adopt a traditional greedy strategy (e.g., [13]) to derive the mapping results of virtual nodes to substrate nodes, and virtual links to substrate paths.

In the micro-level time slot assignment, we focus on the scenario in a *single* substrate link. The results can adapt naturally to the other substrate links and nodes (details are in Section 5). Suppose that the substrate link is based on *time division multiplexing*, where time is partitioned into multiple

- *S. Zhang, Z.Z. Qian, and S.L. Lu are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China. Z.Z. Qian is the corresponding author. E-mail: zhangsheng@dislab.nju.edu.cn, {qzz,sanglu}@nju.edu.cn.*
- *J. Wu is with the Computer and Information Sciences Department, Temple University, Philadelphia, PA 19122, USA. E-mail: jiewu@temple.edu.*
- *L. Epstein is with the Department of Mathematics, University of Haifa, Haifa 31905, Israel. E-mail: lea@math.haifa.ac.il.*

frames of equal length, and each frame is further divided into time slots of equal length. The number of time slots in a frame depends on the physical bandwidth of this substrate link. Several virtual links are embedded in this substrate link; then, the problem becomes how to map the bandwidth requirement of virtual links to the physical time slots. For the basic bandwidth sub-requirement from a virtual link, which exists throughout the lifetime of the respective VN, we have no choice but to allocate the corresponding required slots to it. For the variable bandwidth sub-requirement, we propose to opportunistically share time slots among multiple virtual links to improve resource utilization. However, collisions accompany sharing. To break the tradeoff between utilization and collision, we use a collision probability threshold to represent the "volume" of a time slot and formulate the time slot assignment as an optimization problem. We prove the decision-version problem to be NP-hard by reducing the 3-partition problem [22] to it. An integer linear programming-based (ILP) optimal solution is also provided. Due to the similarities between this problem and the bin packing problem [23], we then propose two practical first-fit-based solutions from different perspectives: *first-fit by collision probability* (CFF) and *first-fit by expectation of indicators' sum* (EFF).

Through extensive simulations, we demonstrate that, in the long run, ORS accepts more virtual network requests and provides a more efficient utilization of substrate resources than two state-of-the-art fixed-resource embedding schemes. The contributions are summarized as follows:

1) To the best of our knowledge, this is the first attempt that considers virtual network embedding in the context of opportunistic resource sharing at the level of the entire network. To provide efficient resource utilization, which is of great benefit to both InPs and SPs, an embedding framework, ORS, is designed; its effectiveness is confirmed by extensive simulations.
2) We propose a novel model that reflects the time-varying properties of the resource requirement of a VN, based on which we formulate the micro-level time slot assignment problem as an optimization problem. We first prove the decision version of this problem to be NP-hard in the strong sense, then propose an ILP-based optimal solution and two practical algorithms.
3) We conduct extensive theoretical analysis and simulation studies to verify the performance of ORS.

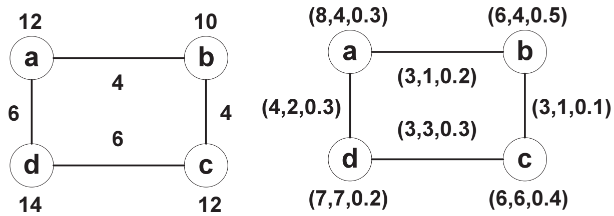
We now continue by proposing the resource requirement model in Section 2 before we introduce the VNE problem in Section 3. We then provide the overview of ORS in Section 4, describe the details of ORS in Section 5, and conduct performance evaluations in Section 6. Before concluding the paper in Section 8, we survey related work in Section 7.

2 Virtual Network Request with Time-Varying Resource Requirement

In this section, we first present the traditional virtual network request model, and then we introduce a model that captures the time-varying properties of virtual network resource requirements.

2.1 Traditional Virtual Network Request Model

The main substrate resources that we consider in this paper are CPU and bandwidth, which is the typical case in almost



(a) Traditional virtual network request (b) VN request with time-varying resource requirement

Fig. 1. Each node or link is associated with a fixed resource requirement in the traditional VN request, while, in our model, the resource requirement of each node or link is expressed in a tuple $< b, v, p >$.

all of the related literature so far. However, our framework can naturally adapt to the scenario where a node has multiple types of resources. We will give remarks on the adaptation in Section 5 when needed.

For the purpose of unifying 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 equal time slots. In doing so, both CPU and bandwidth requirements can be expressed in time slots.

A traditional virtual network request is denoted by a weighted undirected graph, $G^v = (N^v, E^v)$, where N^v and E^v are the sets of virtual nodes and links, respectively. Each virtual node $n^v \in N^v$ is associated with a CPU requirement $C(n^v)$ in time slots, and each virtual link $e^v = (n_i^v, n_j^v) \in E^v$ is associated with a bandwidth requirement $B(e^v)$ in time slots. Fig. 1(a) shows an example, where the corresponding resource requirement of each node or link is written next to the respective node or link that represents it.

2.2 The Time-Varying Resource Requirement Model

SPs can hardly predict the number of end users of the applications deployed in their virtual networks; to guarantee the quality-of-service of a peak workload, SPs always over-purchase substrate resources. Besides, the resource requirements of many applications experience significant changes over time. Therefore, provisioning fixed resources for VNs throughout their lifetimes is clearly wasteful. To avoid such wasteful situations, we need to model the time-varying resource requirement of a VN in the first place.

By using profiling experimentations, one can potentially derive some complicated functions, e.g., high-order polynomials, to capture the time-varying resource requirement in a very precise way [20]. However, such smooth functions may increase the representation and communication burden of SPs, as well as complicate the resource provisioning in SNs. To strike a balance between modeling precision and implementation difficulties, and to initiate a tractable study as a first step, this paper resorts to a probability-based model, and leaves exploring other tradeoffs as future work.

In our model, the time-varying resource requirement of a virtual node or link is composed of a basic sub-requirement, which exists throughout the lifetime of the respective VN, and a variable sub-requirement, which occurs with a probability. Based on this resource requirement mode, we replace the $C(n^v)$ and $B(e^v)$ in the traditional representation with tuples $< b(n^v), v(n^v), p(n^v) >$ and $< b(e^v), v(e^v), p(e^v) >$, respectively, where $b(n^v)$ (resp.

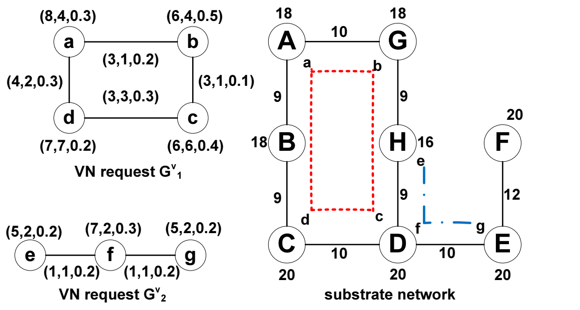


Fig. 2. An example of virtual network embedding

$b(e^v)$) denotes the number of time slots in the basic sub-requirement, and $v(n^v)$ (resp. $v(e^v)$) denotes the number of time slots in the variable sub-requirement, which occurs with probability $p(n^v)$. Take virtual node a in Fig. 1(b) for example, since $< b(a), v(a), p(a) >=< 8, 4, 0.3 >$, we then know that virtual node a needs 8 slots with a probability of 0.7, and needs 12 slots with a probability of 0.3.

Overall, we admit that many challenges remain, e.g., how does an SP choose suitable $< b, v, p >$ tuples to best reflect the time-varying resource requirement of his/her VN. However, the thesis of this paper is the notion of opportunistic resource sharing, and what it brings to InPs and SPs. We hope that this simplified model can provide some insights on the design of future VNE algorithms.

3 The virtual network embedding problem

A substrate network is modeled as a weighted undirected graph, $G^s = (N^s, E^s)$, where N^s and E^s are the sets of substrate nodes and links, respectively. Similarly, each substrate node $n^s \in N^s$ is associated with a CPU capacity $C(n^s)$ in time slots, and each substrate link $e^s = (n_i^s, n_j^s) \in E^s$ is associated with a bandwidth capacity $B(e^s)$ in time slots. The set of loop-free paths from n_i^s to n_j^s is denoted as $P^s(n_i^s, n_j^s)$. The residual resources of n^s and e^s are denoted as $RC^s(n^s)$ and $RB^s(e^s)$, respectively. The computation of $RC^s(n^s)$ and $RB^s(e^s)$ in the context of opportunistic resource sharing is not trivial, as we shall discuss shortly in Section 5.6. The right side of Fig. 2 shows a substrate network, where the corresponding resource capacity of each substrate node or link is written next to the respective node or link that represents it.

The embedding of a VN G_i^v is defined as mapping $\mathcal{M}$ from G_i^v to a subset of G^s, such that the resource requirement of G_i^v is satisfied and the resource capacities in G^s are not violated. It can be further decomposed into two components: 1) *node mapping* $\mathcal{M}_n : N_i^v \to N^s$, which maps different virtual nodes to different substrate nodes; and 2) *link mapping* $\mathcal{M}_l : E_i^v \to P^s$, which maps a virtual link to a substrate loop-free path.

In Fig. 2, the node mapping for G_1^v is $\{a \to A, b \to G, c \to D, d \to C\}$, and the link mapping is $\{(ab) \to \{AG\}, (bc) \to \{GH, HD\}, (cd) \to \{DC\}, (da) \to \{CB, BA\}\}$; the node mapping for G_2^v is $\{e \to H, f \to D, g \to E\}$, and the link mapping is $\{(ef) \to \{HD\}, (fg) \to \{DE\}\}$.

Our main interest is to propose an embedding framework for InPs to cope with a sequence of VN requests that arrive and depart over time. Upon the arrival of request G_i^v, an InP must decide to either accept or reject it. Here, we assume that VN requests arrive one by one, and batch processing is not the focus of this paper. From the standpoint of an InP, the objective is to maximize its revenue through efficiently utilizing its substrate resources. Following prior research [12], [13], the revenue, $\mathcal{R}(G_i^v)$, of embedding G_i^v can be defined as:

$$\mathcal{R}(G_i^v) = [\omega_c \sum_{n^v \in N^v} (b(n^v) + v(n^v)) + \omega_b \sum_{e^v \in E^v} (b(e^v) + v(e^v))] T_i^v$$

where ω_c and ω_b are the weights, providing the flexibility to trade off between the costs of two kinds of resources; T_i^v is the lifetime of G_i^v. Note that the length of substrate paths that virtual links are mapped to does not affect the revenue, since an SP is only willing to pay a rent to the InP that is proportional to the amount of requested resources. To maximize the revenue, VN requests should be intelligently deployed on top of an SN. This paper re-visits this problem from the perspective of opportunistic resource sharing.

4 The Overview of Our Framework

In this section, we present an overview of our framework, ORS. The details are introduced in Sections 5.

ORS generally consists of two components, as shown in Alg. 1. The macro-level node-to-node/link-to-path embedding component adopts a traditional greedy strategy in [13] to derive the mapping of virtual nodes to substrate nodes, and virtual links to substrate paths. In this component, we first place virtual nodes in queue Q with decreasing $(b(Q[i]) + p(Q[i])v(Q[i]))$, which is the expected number of time slots required by a virtual node $Q[i]$; then, we map each virtual node from the head to the end of Q to the unused substrate node with the most residual resource. If the residual resource of a substrate node is less than the expected number of time slots required by the corresponding virtual node, the VN request is rejected. This kind of "maximum-first" embedding fashion is beneficial to future requests that may require some scarce or bottleneck resources. We then map each virtual link to the shortest path [24] with sufficient bandwidth between its end hosts, to minimize the span. We note that, when the VN request contains multiple edges between a pair of nodes, we turn to find the k shortest paths [25] to reduce the sum of the lengths of multiple substrate paths that these edges are mapped to.

In the micro-level component, we run CFF or EFF in each of the substrate nodes and links that are involved in the mapping of G^v to deal with time slot allocations, then we update residual resources of them. The details of this component are introduced in Section 5. It is worth elaborating on that lines 7 and 11 of Alg. 1 only provide early-reject conditions; even when the node mapping $\mathcal{M}_n$ passes the checking condition in line 7, and the link mapping $\mathcal{M}_l$ passes checking condition of line 11, it is still possible that the resource requirement of G^v could not be guaranteed in the micro-level time slot assignment.

While the "maximum-first" strategy of the macro-level component largely comes from [13], the main contributions of this paper lie in the micro-level component. We conclude this section by presenting the time complexity of ORS. In macro-level embedding, the sorting and mapping of virtual nodes takes $O(|N^v| log(|N^v|) + |N^v|)$ time, and finding the k shortest paths takes $O(|E^s| + |N^s| log(|N^s|) + k)$ [25]; since we need to execute the k shortest paths algorithm at most $|N^s|^2$ times, this component takes $O((|N^v| log(|N^v|) + |N^v|) + |N^s|^2(|E^s| + |N^s| log(|N^s|) + k)) = O(|N^s|^4)$ time

Algorithm 1 The ORS embedding framework

Wait until a VN request G^v arrives
Macro-level node-to-node/link-to-path mapping:
for all $n^s \in N^s$ **do** $unused(n^s) \leftarrow 1$ **end for**
$Q \leftarrow$ sorted N^v with decreasing $(b(Q[i])+p(Q[i])v(Q[i]))$
for $i = 1$ to $Q.length$ **do**
  $\mathcal{M}_n(Q[i]) \leftarrow argmax(RC^s(n^s) \cdot unused(n^s))$
  if $RC^s(\mathcal{M}_n(Q[i])) < (b(Q[i]) + p(Q[i])v(Q[i]))$
  then reject G^v and **return**
end for
for all $e^v = (n^v, m^v) \in E^v$ **do**
  $P^{s\prime} \leftarrow \{path|RB^s(path) \geq (p(e^v)v(e^v) + b(e^v)), path \in P^s(\mathcal{M}_n(n^v), \mathcal{M}_n(m^v))\}$
  if $P^{s\prime} == \emptyset$ **then** reject G^v and **return**
  $\mathcal{M}_l(e^v) \leftarrow argmin(hop(path))$
  (the shortest path [24] or the k shortest paths [25])
end for
Micro-level time slot assignment:
for all $n^v \in N^v$ **do**
  if $false == CFF(v(n^v), p(n^v))$ (or EFF)
  then reject G^v and **return**
  update $RC^s(\mathcal{M}_n(n^v))$
end for
for all $e^v \in E^v$ **do**
  for all $e^s \in \mathcal{M}_l(e^v)$ **do**
    if $false == CFF(v(e^v), p(e^v))$ (or EFF)
    then reject G^v and **return**
    update $RB^s(e^s)$
  end for
end for

in all. Here we have simplified the summations by using $|E^s| = O(|N^s|^2)$. Based on the results in Section 5.7, the micro-level component takes $O(F|N^s|^2)$, therefore, the overall time complexity of ORS is $O(|N^s|^4 + F|N^s|^2)$.

5 Micro-level Time Slot Assignment—An Opportunistic Resource Sharing View

In this section, we will first provide a formal description of the time slot assignment problem and its hardness result. Then, we present an ILP-based optimal solution and two practical first-fit-based solutions. We also show how to estimate residual resources of substrate nodes and links. Finally, we will give a brief summary of this section.

5.1 Problem Formulation

Since both CPU and bandwidth requirements can be expressed as time slots, this section only takes the time slot assignment in a substrate link for illustration. The solutions can be applied to substrate nodes without any changes.

Consider the following scenario, where a set of n virtual links from different VNs are embedded across a substrate link. For simplicity, the resource requirements from different VN requests are assumed to be independent of each other. This seems to be reasonable, since VNs are operated by different SPs and offer different services to different users. For the basic sub-requirements that exist throughout the lifetime of the respective VN request, we must allocate the required number of dedicated time slots for them; however, for the variable sub-requirements, since they occur with a probability that is less than 1, sharing may be a viable choice to conserve substrate resources for future VN requests. Therefore, we will only consider how to assign substrate slots to variable sub-requirements in the rest of this section.

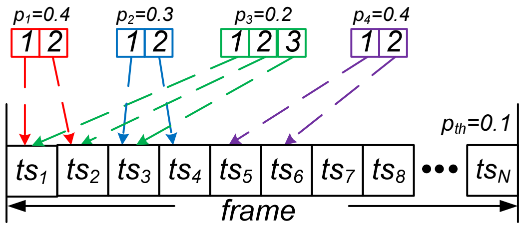


Fig. 3. The time slot assignment problem. The probability threshold serves as the “volume” of a substrate time slot.

We propose to assign one substrate slot to multiple units of variable sub-requirements. However, collisions may happen, i.e., multiple units of sub-requirements occur simultaneously. To strike a tradeoff between utilization and collision, we use a collision threshold p_{th} to represent the “volume” of a substrate time slot.

Denote by D_j the set of variable sub-requirements that substrate slot ts_j is assigned to; let X_i indicate whether the i-th variable sub-requirement occurs, i.e., $Pr[X_i = 1] = p_i$. Then, the probability of a collision happening at slot ts_j, denoted by $Pr(D_j)$, is:

$$Pr(D_j) = Pr[\sum_{i \in D_j} X_i \geq 1] = 1 - \prod_{i \in D_j}(1 - p_i) - \sum_{i \in D_j}(p_i \prod_{k \in D_j, k \neq i}(1 - p_k)) \quad (1)$$

We have the following optimization problem.

Problem 1: (**The time slot assignment problem, TSA**) Given a set of n virtual links from different VNs, the variable sub-requirement of the i-th virtual link is v_i time slots, each of which is needed with probability p_i. Find an assignment of substrate time slots to the sub-requirements to minimize the number of slots used, such that: 1) for the variable sub-requirement of the i-th virtual link, the number of time slots assigned to it is at least v_i; and 2) the collision probability at each substrate time slot is no more than a given collision threshold p_{th}.

For example, Fig. 3 shows a feasible assignment. ts_1 can be assigned to two variable sub-requirements because they collide with a probability 0.08, which is less than $p_{th} = 0.1$; however, ts_4 can not be assigned to the 2-th and 4-th sub-requirements simultaneously, because the collision probability 0.12 is larger than p_{th}.

For the hardness of the TSA problem, we have the following theorem. Please refer to the supplemental material for the detailed proofs of all the theorems in this paper.

Theorem 1: TSA is NP-hard in the strong sense

5.2 An ILP-based Optimal Solution

Inspired by the *cutting stock* problem[1], we can formulate the TSA problem by means of ILP. Denote a set of variable sub-requirements whose collision probability is no more than p_{th} as a *pattern*. Denote the number of all possible patterns as m. For each possible pattern j, let x_j represent the times

1. Cutting stock problem [26]: Given a number of rolls of paper of fixed width waiting to be cut, yet different customers want different numbers of rolls of various-sized widths, find a cutting method to minimize the waste.

that pattern j appears in a feasible assignment. Thus, the TSA problem can be formulated as:

$$\begin{aligned} \min \ & \sum_{j=1}^{m} x_j \\ \text{s.t.} & \sum_{j=1}^{m} (a_{ji} x_j) \geq v_i, \forall i \in \{1,2,...,n\} \\ & x_j, \ \text{nonnegative integer}, \forall j \in \{1,2,...,m\} \end{aligned} \tag{2}$$

where a_{ji} indicates whether pattern j contains the i-th sub-requirement. Ideally, Equ. (2) can be optimally solved using intelligent exhaustive search approaches, such as backtracking and branch-and-bound [24]. However, it is not practical. First, the number of possible patterns can be exponentially large, the construction of which costs exponential time; Second, the intelligent exhaustive search approach usually consists of a systematic enumeration of all candidate solutions, which is also difficult to apply in practice. This motivates us to design practical solutions, which are introduced in the next two subsections.

5.3 First-Fit by Collision Probability

In the *bin packing* problem [23], we are given n items with sizes s_1, s_2, ..., $s_n \in (0,1]$, and the objective is to find a packing method in unit-sized bins that minimizes the number of bins used. We observe that, when each variable sub-requirement requires only one time slot, i.e., $v_i = 1$ for all $1 \leq i \leq n$, TSA is similar to bin packing, except that the size of multiple items is the sum of them in bin packing; the collision probability of multiple sub-requirements is neither linear nor multiplicative, as shown in Equ. (1).

The first-fit algorithm [23] is a greedy approximation algorithm of factor 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 this is not possible, the item is placed into a new bin. First-fit can be executed online, and has a low time complexity.

Algorithm 2 First-Fit by Collision Probability (CFF)

Input: v_i and p_i
$cnt \leftarrow 0$, $index \leftarrow 0$
while $cnt < v_i$ **do**
  while $getCollistionPro(D_{index}, p_i) > p_{th}$ **do**
    $index \leftarrow index + 1$
    if $index > N$ **return false**
  end while
  $D_{index} \leftarrow D_{index} \cup \{i\}$
  $cnt \leftarrow cnt + 1$, $index \leftarrow index + 1$
  if $index > N$ **return false**
end while
return true

The resemblance between the two problems inspires us to adopt the core idea of first-fit and design the "First-Fit by Collision Probability" (CFF) algorithm, shown in Alg. 2. In the algorithm, N is the total number of substrate time slots, and D_j is the set of sub-requirements that the j-th substrate time slot is assigned to; the fuction $getCollisionPro(D_{index}, p_i)$ returns the collision probability of sub-requirements $D_{index} \cup \{i\}$ and can be implemented in a incremental manner. Let:

$$A(D_j) = \prod_{h \in D_j} (1 - p_h)$$
$$B(D_j) = \sum_{h \in D_j} (p_h \prod_{k \in D_j, \ k \neq h} (1 - p_k))$$

then the collision probability in Equ. (1) can be rewritten as $Pr(D_j) = 1 - A(D_j) - B(D_j)$. We have:

$$\begin{aligned} A(D_j \cup \{i\}) &= A(D_j)(1 - p_i) \\ B(D_j \cup \{i\}) &= B(D_j)(1 - p_i) + A(D_j) p_i \end{aligned} \tag{3}$$

Let us look at the performance guarantee of CFF. Denote by S_{cff} the assignment results from CFF, and by S_{opt} the results from the optimal solution. Abusing the notation a bit, we also use S_{cff} and S_{opt} to denote the number of substrate slots used in these results, respectively, if no confusion can be caused. Let:

$$p_{min} = min_{1 \leq i \leq n} p_i, v_{min} = min_{1 \leq i \leq n} v_i$$
$$p_{max} = max_{1 \leq i \leq n} p_i, v_{max} = max_{1 \leq i \leq n} v_i$$

We then have the following theorem.

***Theorem** 2:* $S_{cff} \leq S_{opt}(v_{max} \cdot vol_1)/(v_{min} \cdot vol_2)$, where vol_I and vol_{II} are the roots of equations:

$$1 - (1 - p_{min})^{vol_1} - vol_1 \cdot p_{min} \cdot (1 - p_{min})^{vol_1 - 1} = p_{th}$$
$$1 - (1 - p_{max})^{vol_2} - vol_2 \cdot p_{max} \cdot (1 - p_{max})^{vol_2 - 1} = p_{th}$$

5.4 First-Fit by Expectation of Indicators' Sum

In Alg. 2, the $getCollisionPro$ function is invoked whenever we want to see whether a substrate slot can accommodate a unit of variable sub-requirement, and it still costs five additions and three multiplications, even when using incremental calculation. Recall that the number of substrate nodes and links may be very large; if we could reduce the time complexity of $getCollisionPro$ a little, then the total benefit would be great.

Denote X_i as the indicator of the i-th variable sub-requirement. Our motivational question is, for a given p_{th}, does a corresponding value exist such that, if the sum of the indicators of a set of variable sub-requirements is less than that value, then we can definitely know that the collision probability of them is less than p_{th}? Fortunately, based on *Chernoff bound* [27], we prove the following theorem.

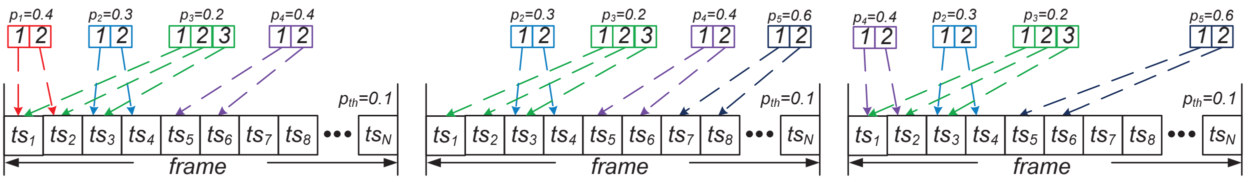
***Theorem** 3:* If $E[\sum_{i \in D_j} X_i] \leq \mu_{th}$, then $Pr[D_j] \leq p_{th}$, where $\mu_{th} e^{1 - \mu_{th}} = p_{th}$, and e is the exponential constant.

Given the value of p_{th}, we have to solve a transcendental equation $p_{th} = \mu_{th} e^{1 - \mu_{th}}$ to get the corresponding μ_{th}. In our implementation, we resort to numerical methods. We notice that the curve of $p_{th} = \mu_{th} e^{1 - \mu_{th}}$ is similar to a parabola; therefore, polynomial interpolation is used to approximately calculate μ_{th}. Given three points, (0.1,0.245), (0.5,0.824), and (0.9,0.994), we get:

$$p_{th} \approx -1.27812 {\mu_{th}}^2 + 2.21437 \mu_{th} + 0.0363438$$

With the help of this theorem, the original determination of whether a substrate slot can accommodate a unit of variable sub-requirement turns into evaluating whether the expectation of the sum of the sub-requirements' indicators is less than μ_{th}. We then modify the TSA problem a little and get the following problem.

***Problem** 2:* (**The Expectation-based time slot assignment problem, ETSA**) Given a set of n virtual links from different



(a) The original assignment (b) Without rearrangement. (c) With rearrangement

Fig. 4. Fragmentation of time slots due to the dynamics of virtual networks. (a) shows the original assignment; after some time, the first virtual link leaves and the fifth virtual link comes; (b) and (c) show the scenarios without and with rearrangement, respectively. We see that the rearrangement reduces the number of slots used by 2.

	$p=0.1$			$p=0.2$		
n	$E[Y]$	$Pr[coll]$	p_{th}	$E[Y]$	$Pr[coll]$	p_{th}
1	0.1	0	0.245	0.2	0	0.445
2	0.2	0.01	0.445	0.4	0.04	0.729
3	0.3	0.028	0.604	0.6	0.104	0.895
4	0.4	0.052	0.729	0.8	0.181	0.977
5	0.5	0.081	0.824			
9	0.9	0.225	0.994			

Fig. 5. Due to the linearity of expectation, the mutual independence is ignored in EFF, leading to a relaxation gap.

VNs, the variable sub-requirement of the i-th virtual link is v_i time slots, each of which is needed with probability p_i. Find an assignment of substrate time slots to the sub-requirements to minimize the number of slots used, such that: 1) for the variable sub-requirement of the i-th virtual link, the number of time slots assigned to it is at least v_i; and 2) the expectation of the sum of the indicators of a set of variable sub-requirements that a substrate slot is assigned to is no more than a given expectation threshold μ_{th}.

***Theorem** 4:* The ETSA problem is NP-complete.

We replace the condition in line 4 of Alg. 2 with $p_i + \sum_{k\in D_{index}} p_k > \mu_{th}$, and name the new algorithm "First-Fit by Expectation of Indicators' Sum" (EFF). In doing so, the checking condition in line 4 is reduced to one addition operation, suggesting that EFF may run faster than CFF.

It turns out that using an expectation threshold decreases the number of variable sub-requirements that a substrate slot can be assigned to, however, this relaxation gap is a bit more subtle than it might initially appear. To motivate it, we start with the following illuminating example.

Consider a substrate slot that is assigned to n variable sub-requirements from different virtual links, each occurring with the same probability p; then, the collision probability $Pr[coll]$ is $1-(1-p)^n - np(1-p)^{n-1}$ and the expectation of the sum of indicators $E[Y]$ is np. For each $E[Y]$, we obtain a value of p_{th} by Theorem 3. Fig. 5 shows the relaxation gap. For instance, when $n=2$ and $p=0.1$, we have $E[Y]=2\times 0.1=0.2$, $Pr[coll]=1-(1-0.1)^2-2\times 0.1\times(1-0.1)=0.01$, $p_{th}=E[Y]e^{1-E[Y]}=0.445$, indicating, if we use $\mu_{th}=0.2$ as the expectation threshold, then the collision probability is guaranteed to be no more than 0.445. However, the collision probability of these two sub-requirements is 0.01, which is much smaller than 0.445.

The main reason behind this phenomenon is that, mutual independence is ignored in the EFF algorithm due to the linearity of expectation. To make up the relaxation gap, we replace μ_{th} by $\lambda\mu_{th}$ in EFF, i.e., $p_i + \sum_{k\in D_{index}} p_k > \lambda\mu_{th}$. Here, the parameter λ is used to control the relaxation, and its empirical value will be investigated in our simulations.

5.5 Rearrangement

Due to the dynamics of virtual network requests, the substrate resources may become fragmented, i.e., some shared time slots are not in full use. In this subsection, we propose to use rearrangement to avoid resource fragmentation and improve resource utilization.

We start with an illustrating example, shown in Fig. 4. Fig. 4(a) shows a snapshot of the time slot assignment in a substrate link. Note that only the shared time slots are shown in the figure, since the dedicated time slots are in full use all the time. After some time, the first virtual link along with its variable sub-requirement leaves, and the fifth virtual link along with its variable sub-requirement arrives. According to the first-fit-based algorithms, we first check whether ts_1 can accommodate a unit of sub-requirement from the fifth virtual link, and it cannot, since $p_3p_5=0.12 > p_{th}$. We then check the following slots, and finally reach the assignment shown in Fig. 4(b), where 8 slots are used.

However, if we rearrange the time slot assignment when the first virtual link leaves, we could assign ts_1 and ts_2 to the variable sub-requirements from the fourth virtual link. In doing so, slots ts_5 and ts_6 would be assigned to the newly arrived virtual link. The final assignment is shown in Fig. 4(c), where we can see that the rearrangement reduces the number of slots used by 2.

This example motivates us to propose the rearrangement protocol as follows. On a virtual network request's leave, or at intervals set by an InP, the following operations are performed in every substrate node and link: for decreasing j from N to 1, the sub-requirements in D_j are reassigned by using CFF or EFF. The loop ends upon an encounter with a substrate slot, which is just assigned to a new sub-requirement by this rearrangement protocol.

In a sense, rearrangement "compresses" the assignment so that it takes up less time slots, which is beneficial to future VN requests, and improves substrate resources utilization. It is worth noticing that, after the rearrangement is performed, the residual resources of substrate nodes and links change. To capture this change, the residual resource estimation should be executed. We can see that the rearrangement incurs some computational overhead; therefore, our protocol allows InPs to achieve a tradeoff between resource utilization and computational overhead by tuning the trigger intervals.

5.6 Estimating Residual Resource

This subsection presents how we estimate the residual resources of each substrate node and link in the context of opportunistic resource sharing.

Residual resources are traditionally defined as follows: $RC^s(n^s) = C^s(n^s) - \sum_{\forall n^v} f_c(n^v, n^s)$ and $RB^s(e^s) =$

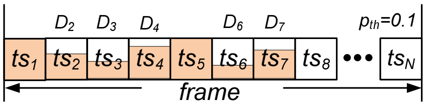


Fig. 6. A snapshot of time slot allocation in a substrate link. ts_1 and ts_5 are assigned to some basic sub-requirements; each of ts_2, ts_3, ts_4, ts_6, and ts_7 is assigned to a set of variable sub-requirements, denoted as D_i, respectively; the other slots are unused.

$B^s(e^s) - \sum_{\forall e^v} f_b(e^v, e^s)$, where $f_c(n^v, n^s)$ denotes the amount of the CPU resources in n^s that are allocated to n^v, and $f_b(e^v, e^s)$ denotes the amount of the bandwidth resources in e^s that are allocated to e^v. Since both CPU and bandwidth are expressed in time slots, this subsection focuses on $RB^s(e^s)$; $RC^s(n^s)$ can be analyzed similarly.

However, when we apply opportunistic resource sharing to the resource allocation in substrate networks, some substrate time slots are shared among multiple virtual networks, then it is non-trivial to calculate the amount of residual resources in a substrate node or link. Fig. 6 shows a time slot allocation snapshot in a substrate link. We see that ts_1 and ts_5 are assigned to some basic sub-requirements; each of ts_2, ts_3, ts_4, ts_6, and ts_7 is assigned to a set of variable sub-requirements, denoted as D_i; the other slots are unused. The residual resource should include the unused slots and the residual "room" in the shared slots. We then propose a reasonable method to properly measure the latter.

For a substrate node or link that has N time slots, where $N = C^s(n^s)$ if it is a substrate node n^s, or $N = B^s(e^s)$ if it is a substrate link e^s, denote the set of slots that are assigned to basic sub-requirements as S_b; denote the set of slots that are assigned to variable sub-requirements as S_v; denote the rest as S_u. For example, in Fig. 6, $S_b = \{1, 5\}$, $S_v = \{2, 3, 4, 6, 7\}$, and $S_u = \{1, 2, 3, ..., N\} \setminus (S_b \cup S_v)$.

The residual room rr_k in the k-th slot which belongs to S_v is defined as a probability that satisfies the following condition: if we assign ts_k to a new variable sub-requirement, which occurs with this probability, then the collision probability would be equal to p_{th}. This definition is intuitively reasonable, as it indicates the maximum probability of a variable sub-requirement that we can assign ts_k to.

When $|D_k| = 1$ and $D_k = \{h\}$, $rr_k = p_{th}/p_h$; when $|D_k| > 1$, according to Equ. (3), we have:

$$1 - A(D_k)(1 - rr_k) - (B(D_k)(1 - rr_k) + A(D_k)rr_k) = p_{th}$$

After solving it, we get:

$$rr_k = \frac{A(D_k) + B(D_k) + p_{th} - 1}{B(D_k)} = \frac{p_{th} - Pr(D_k)}{B(D_k)} \tag{4}$$

Thereby, the residual resource of this substrate link is:

$$RB^s(e^s) = |S_u| + \sum\nolimits_{k \in S_v} min\{rr_k, 1\} \tag{5}$$

Take ts_1 in Fig. 2 for example, $Pr(\{1, 3\}) = 0.08$, $B(\{1, 3\}) = 0.44$; thus, the residual room in ts_1 is $rr_1 = (p_{th} - Pr(\{1, 3\}))/B(\{1, 3\}) \approx 0.045$.

5.7 Remarks and Summary

In summary, this section starts with the formulation and the NP-hard result of the micro-level time slot assignment problem, then provides an ILP-based optimal solution, which is not practical. The similarities between our problem and bin packing further motivates us to propose two first-fit-based heuristics, the performances of which are to be investigated in our extensive simulations. We then design a simple rearrangement protocol to cope with resource fragmentation, and show how to estimate residual resources of substrate nodes and links. We also provide in Section 1 of the supplemental material some intuitive insights on how opportunistic resource sharing can lead to a win-win situation—service providers' costs are lowered, while infrastructure providers' revenues increase, as well.

We note that the adaptation of the micro-level time slot assignment to the scenario where a node has multiple types of resources is trivial, since the algorithms in this section are micro-level, and are executed in every substrate and link. When there are multiple types of resources, the InPs just have to run the algorithms for them individually.

We conclude this section by presenting the time complexity results. Denote the maximum variable sub-requirement among all of the virtual nodes and links from a virtual network as $max(v)$; denote the maximum capacity among all of the substrate nodes and links in a substrate network as $max(max(B), max(C))$. Let $F = max(v) \cdot max(max(B), max(C))$, then, both CFF and EFF have at most $O(F)$ comparisons. The estimation of residual resources takes $O(|N^s|+|E^s|)$ time. The overall time complexity of the micro-level component is $O((|N^s|+|E^s|)(1+F)) = O(F|N^s|^2)$, where $|N^s|$ and $|E^s|$ are the cardinalities of N^s and E^s, respectively.

6 Performance Evaluation

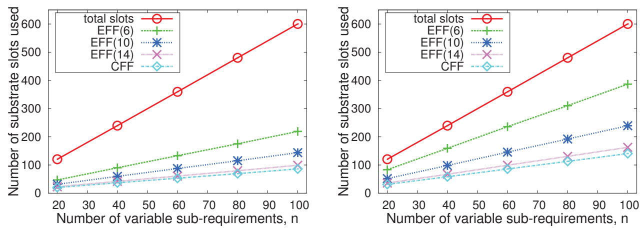
In this section, we first concentrate on the scenario of a single substrate link in an effort to quantify the benefits of opportunistic resource sharing and compare the performances of CFF and EFF. We then compare ORS with two state-of-the-art fixed-resource embedding schemes.

6.1 Single Substrate Link

We first consider a scenario where a single substrate link is shared among multiple virtual links from different virtual network requests. Since we have no choice but to allocate the corresponding required slots for basic sub-requirements, we do not consider the basic sub-requirements in this subsection. The number of variable sub-requirements is n, and the i-th ($1 \le i \le n$) sub-requirement needs v_i slots with probability p_i. In our simulation, v_i is uniformly generated between 2 and v_{max}; p_i is uniformly generated from two intervals, i.e., $(0.05, 0.10)$ and $(0.05, 0.20)$; the collision threshold p_{th} is chosen from $\{0.1, 0.2, 0.3\}$. We try to compare the performances of CFF and EFF, and see the effects of n, v_{max} and p_{th}.

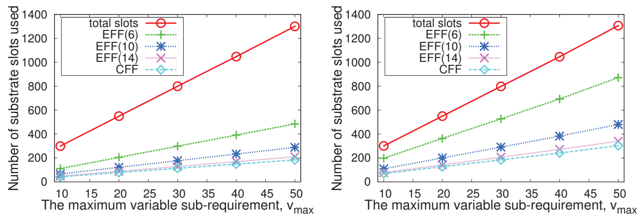
6.2 Results of Single Substrate Link

(1) The impact of n: Fig. 7 shows the corresponding results, where we keep the other parameters fixed, e.g., $p_{th} = 0.1$ and $v_{max} = 10$. We denote EFF with relaxation parameter λ by EFF(λ), and the number of substrate slots that are needed, if opportunistic resource sharing is not adopted, by "total slots." We note that, when n increases from 20 to 100 with an increment of 20, the data points are linear in shape, indicating that the number of substrate slots used grows linearly with n. We also see that, when λ increases,



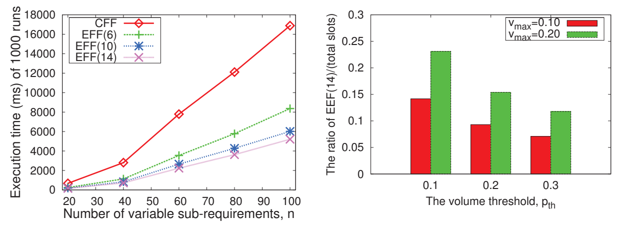
(a) $p_i \in (0.05, 0.10)$ (b) $p_i \in (0.05, 0.20)$

Fig. 7. Comparison of CFF and EFF under varying n while keeping $p_{th} = 0.1$ and $v_{max} = 10$. EFF(x) denotes $\lambda = x$.



(a) $p_i \in (0.05, 0.10)$ (b) $p_i \in (0.05, 0.20)$

Fig. 8. Comparison of CFF and EFF under varying v_{max} while keeping $p_{th} = 0.1$ and $n = 50$.



(a) Comparison of running time. $p_{th} = 0.1$, $v_{max} = 30$ and $p_i \in (0.05, 0.10)$

(b) The ratio of $EFF(14)$ to total slots under different thresholds, where $n = 100$ and $v_{max} = 10$

Fig. 9. Running time comparison and the impact of p_{th}.

the results of EFF(λ) occupy less substrate slots, since a larger λ allows more sub-requirements to be accommodated in a single substrate slot. We also find that EFF(14) achieves almost the same results as CFF; however, when $\lambda > 14$, as we shall explain shortly in Fig. 9, the collision probability would be bigger than the threshold.

(2) The impact of v_{max}: Fig. 8 shows the corresponding results, where we keep the other parameters fixed, e.g., $p_{th} = 0.1$ and $n = 50$. When v_{max} goes up from 10 to 50 with an increment of 10, the substrate slots used also grows linearly with v_{max}. By comparing Fig. 8(a) with Fig. 8(b), we find that, when p_i doubles on average, the number of slots used nearly doubles. The main reason behind this phenomenon is, when p_i increases on average, the number of sub-requirements that a substrate slot can accommodate decreases; however, as the collision probability is neither additive nor multiplicative, the double of p_i does not necessarily lead to a doubling of the number of slots used.

(3) Comparison of running times: Fig. 9(a) demonstrates the comparison results between the running times of CFF and EFF, where $p_{th} = 0.1$, $v_{max} = 30$, and $p_i \in (0.05, 0.10)$. We make two observations. First, EFF generally runs faster than CFF. The main reason behind this phenomenon is, as we mentioned in Section 5.4, EFF replaces the $getCollisionPro$ function, which requires five additions and three multiplications, with just one addition. Second, EFF(λ) runs faster when λ is increasing. The reason is implicit, if somewhat subtle: one substrate slot can accommodate more variable sub-requirements when λ becomes larger, thus, the value of $index$ in EFF(λ) becomes smaller on average.

(4) The impact of p_{th}: Fig. 9(b) shows the ratio of EFF(14) to total slots under different thresholds, while we keep $n = 100$ and $v_{max} = 10$. We note that, for fixed v_{max}, the ratio goes down when the threshold increases. This is reasonable, since the threshold serves as the "volume" of a substrate slot, and a larger threshold allows a substrate slot to accommodate more sub-requirements. For fixed p_{th}, the ratio goes up when v_{max} increases. This is because a larger v_{max} makes the number of sub-requirements that a substrate slot can accommodate decrease, and hence EFF(14) needs more substrate slots.

In our simulations, we also find that, when $\lambda > 14$, the collision probability in the embedding results of EFF would be bigger than $p_{th} = 0.1$. In addition, this critical value is about 10 when $p_{th} = 0.2$, and about 8 when $p_{th} = 0.3$. We explain this as follows: if we replace every p_i with $p_{max} = max_{(1 \le i \le n)} p_i$, then the number of sub-requirements that a single substrate slot can accommodate, denoted as y, can be resolved by $1 - (1 - p_{max})^y - p_{max}(1 - p_{max})^{y-1} y = p_{th}$. Then, by double counting the indicators' sum, we get $\lambda \mu_{th} = p_{max} y$. When p_{th} goes up, both y and μ_{th} go up, but λ goes down, indicating that μ_{th} grows faster than y.

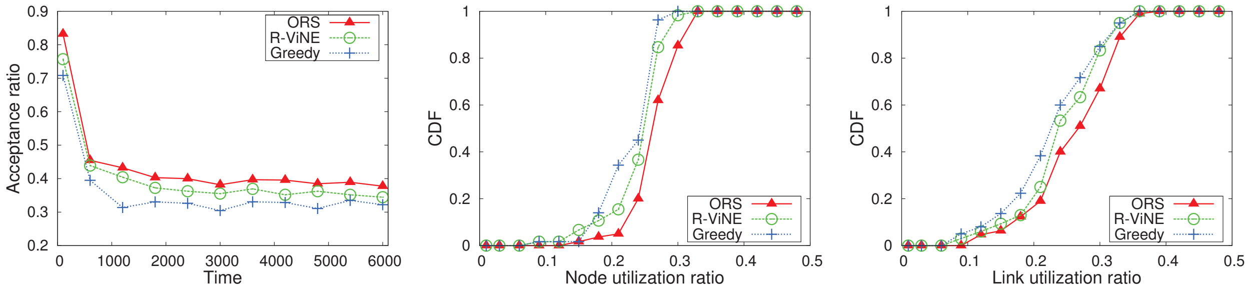
We also conducted simulations with $p_{th} = 0.2$ and $p_{th} = 0.3$. The results are similar to the above, and are therefore omitted due to space limitations. Briefly speaking, both CFF and EFF improve the resource utilization, and EFF is less time-consuming and more flexible than CFF.

6.3 Entire Substrate Network

In this subsection, we consider VNE at the level of the entire network, compare our framework with two state-of-the-art fixed-resource embedding algorithms [12], [13], and investigate the impacts of various parameters.

Our simulation settings follow prior work [12], [13], as network virtualization is still in its infancy. We use ANSNET and ARPANET as the substrate network topologies. Both CPU and bandwidth capacities in substrate networks are generated uniformly from the interval between 50 and 100. For virtual networks, the number of virtual nodes is determined by a uniform distribution between 2 and 10, and each pair of virtual nodes is connected with a probability of 0.5. We also check whether a virtual network is connected; if it is not, we just regenerate it until we get a connected topology. The lifetime of each virtual network is assumed to be exponentially distributed with an average of 10 minutes. The arrivals of VN requests are modeled as a Poisson process with an average rate of five requests per minute. The collision probability threshold is set to 0.1 throughout this evaluating scenario. The results are averaged over 100 times of running. (Results over ARPANET are similar and are omitted due to space limitations.) Our framework ORS is compared with the following two algorithms:

- $R-ViNE$ [12]: coordinated node and link mapping through mixed integer programming formulation and randomized rounding.
- $Greedy$ [13]: greedy node mapping and path splitting.



(a) Acceptance ratio over time (b) CDF of node utilization ratio (c) CDF of link utilization ratio

Fig. 10. Comparison results among ORS, $R-ViNE$ and $Greedy$, where $E[b+v] = 10, E[b/(b+v)] = 0.5, E[p] = 0.15$.

The performance metrics we use for comparison include *acceptance ratio*, which is the ratio of the number of accepted virtual network requests to all requests, *node utilization ratio*, which is the ratio of the amount of the allocated CPU resources to overall CPU resources in the substrate network, and *link utilization ratio*, which is the ratio of the amount of the allocated bandwidth resources to overall bandwidth resources in the substrate network. We are also interested in the impacts of the following parameters:

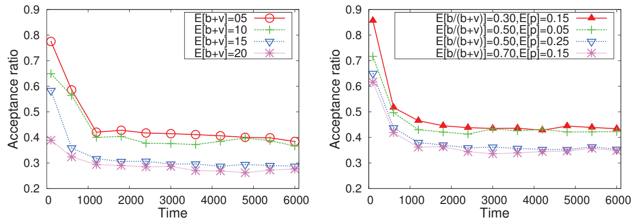
- $E[b+v]$: the average total number of slots required by a virtual node or link;
- $E[b/(b+v)]$: the average percentage of the number of slots in a basic sub-requirement to the total number of slots required by a virtual node or link;
- $E[p]$: the average happening probability of variable sub-requirements of virtual nodes and links.

6.4 Results of Entire Substrate Network

(1) Comparison of acceptance ratios: Figs. 10(a), 10(b), and 10(c) show the comparison of the acceptance ratio over time, *cumulative distribution function* (CDF) of node utilization ratio, and CDF of link utilization ratio, respectively. In these experiments, $E[b+v]$ is 10, $E[b/(b+v)] = 0.5$, and $E[p] = 0.15$. In Fig. 10(a), as a whole, the acceptance ratio of ORS is the highest, and $Greedy$ is the lowest, indicating that opportunistic resource sharing indeed improves the deployment of virtual networks, which further enables the substrate network to accept more VN requests. We notice that the acceptance ratio of three algorithms is about 0.4 on average, which is a little low. The main reason is that 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 a probability of 0.5. Thus, topology becomes the dominating factor in our simulation scenarios.

(2) Comparison of node and link utilization ratios: In Figs. 10(b) and 10(c), the node/link utilization ratios of ORS and $R-ViNE$ are the highest and the second highest, respectively. We notice that the link utilization ratio is a little higher than node utilization ratio in every algorithm, i.e., each CDF curve in Fig. 10(b) is in the left of the corresponding curve in Fig. 10(c), if we can put these two figures together and look at them. This is reasonable, since a virtual link spans over several substrate links, while a virtual node only exists in a substrate node.

(3) The impact of $E[b+v]$: Fig. 11(b) shows the results of the impact of $E[b+v]$. We note that, in the case of a small $E[b+v]$, the acceptance ratio is high. However, with increasing $E[b+v]$, the substrate network resources become scarce,



(a) Impact of C^v and B^v (b) Impact of b and p

Fig. 11. Sensitivity analysis. In (a) $E[b/(b+v)] = 0.5, E[p] = 0.15$; and in (b) $E[b+v] = 10$.

which causes more and more VN requests to be rejected. In this figure, $(E[b+v] = 15)$ achieves almost the same acceptance ratio as $(E[b+v] = 20)$. The main reason behind this phenomenon is that $(E[b+v] = 15)$ is sufficiently large compared to the average capacity of substrate nodes and links, i.e., 75 in our simulation.

(4) The impact of $E[b/(b+v)]$ and $E[p]$: Fig. 11(c) shows the impact of them, where $(E[b/(b+v)] = 0.30, E[p] = 0.15)$ has the best performance, and $(E[b/(b+v)] = 0.50, E[p] = 0.05)$ has the second best, indicating that the basic sub-requirement percentage $b/(b+v)$ plays a more important role than the occurring probability p, which is reasonable, since the basic sub-requirements cannot be shared.

In summary, simulations of the single substrate link s-cenario demonstrate that both CFF and EFF improve the resource utilization of substrate networks, and EFF is more flexible and less time-consuming than CFF. In addition, simulations of the entire substrate network show that our framework outperforms two state-of-the-art fixed-resource embedding algorithms, in terms of both acceptance ratio and utilization ratio. Our results also show some insights into the impacts of various parameters.

7 Related Work

For the general network virtualization, cognitive radio-based virtual networks are envisioned in [28]; optical backbone network virtualization is investigated in [29]. Virtulization is used to lower the barrier for deploying wide-area services in [30]. Adaptive resource allocation is introduced to maximize the aggregate performance across multiple virtual networks in [31].

For the virtual network embedding problem, a large number of algorithms have been proposed in the past. These algorithms give good inspiration to the design of ORS. Simulated annealing was introduced to cope with VNE's NP-completeness in [9] and [19]. Embedding with unlimited substrate resources is studied in [11] and [10]. Zhu *et al.* [11]

focused on load balancing and on-demand assignments, and Lu *et al.* [10] attempted to minimize the embedding cost of a single virtual network with a backbone-star topology. Yu *et al.* [13] envisioned path splitting support from substrate networks, and proposed to first map virtual nodes greedily, then handle link mapping based on the multi-commodity flow algorithm. Lischka *et al.* [14] proposed a backtracking algorithm based on subgraph isomorphism detection, but restricted the length of the substrate paths. Chowdhury *et al.* [12] proposed a linear programming and deterministic/randomized rounding-based algorithm with better coordination between node and link mappings, but added location constraints to simplify the problem. Chowdhury *et al.* [16] presented a policy-based decentralized inter-domain virtual network embedding framework, and also designed a location-aware VN request forwarding mechanism. Recently, Bienkowski *et al.* [32] presented a competitive analysis framework for service migration in a mobile network virtualization architecture, where thin clients on mobile devices access services that can be migrated closer to the access points, as to reduce user latency. Even *et al.* [7] proposed a competitive online algorithm for admission control, while assuming the existence of an *oracle* that helps to compute the embedding.

Comparatively, while prior embedding algorithms reserve fixed resources throughout the lifetime of a virtual network, this work rethinks this paradigm and proposes to opportunistically share resources among multiple virtual networks, so as to make efficient use of the precious substrate resources.

8 Conclusions

In this paper, we rethink the virtual network embedding problem from the perspective of opportunistic resource sharing, and we propose an embedding framework that consists of the macro-level node-to-node/link-to-path embedding and the micro-level time slot assignment. Extensive simulations confirm the effectiveness of our framework.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful suggestions. This work was supported in part by NSFC Grants (No. 61073028, No. 61202113, and No. 61021062), Key Project of Jiangsu Research Program Grant (No. BE2010179), Jiangsu NSF Grant (No. BK2011510), 973 Program of China Grants (No. 2009CB320705 and No. 2011CB302800), College graduate research and innovation project of Jiangsu Grant (No. CXZZ12_0055), and 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*.

[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 CCR*, 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] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM CCR*, vol. 33, no. 3, pp. 3–12, 2003.

[6] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI veritas: realistic and controlled network experimentation," *ACM SIGCOMM CCR*, vol. 36, no. 4, pp. 3–14, 2006.

[7] G. Even, M. Medina, G. Schaffrath, and S. Schmid, "Competitive and deterministic embeddings of virtual networks," in *ICDCN 2012*.

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

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

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

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

[12] M. Chowdhury, M. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM TON*, vol. 20, no. 1, pp. 206–219, 2012.

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

[14] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *VISA 2009*.

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

[16] M. Chowdhury, F. Samuel, and R. Boutaba, "Polyvine: policy-based virtual network embedding across multiple domains," in *ACM SIGCOMM VISA 2010*.

[17] N. Butt, N. Chowdhury, and R. Boutaba, "Topology-awareness and reoptimization mechanism for virtual network embedding," in *IFIP Networking 2010*.

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

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

[20] D. Xie, N. Ding, Y. C. Hu, and R. Kompella, "The only constant is change: incorporating time-varying network reservations in data centers," in *ACM SIGCOMM 2012*.

[21] S. Zhang, Z. Qian, J. Wu, and S. Lu, "An opportunistic resource sharing and topology-aware mapping framework for virtual networks," in *IEEE INFOCOM 2012*.

[22] M. Gary and D. Johnson, "Computers and intractability: A guide to the theory of np-completeness," *WH Freeman and Company, New York*, 1979.

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

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press.

[25] D. Eppstein, "Finding the k shortest paths," in *FOCS 1994*.

[26] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Operations Research*, vol. 9, pp. 849–859, 1961.

[27] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Annals of Mathematical Statistics*, vol. 23, no. 4, pp. 493–507, 1952.

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

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

[30] Y. Zhu, R. Zhang-Shen, S. Rangarajan, and J. Rexford, "Cabernet: connectivity architecture for better network services," in *ACM CoNEXT 2008*.

[31] 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*.

[32] M. Bienkowski, A. Feldmann, D. Jurca, W. Kellerer, G. Schaffrath, S. Schmid, and J. Widmer, "Competitive analysis for service migration in vnets," in *ACM VISA 2010*.

Sheng Zhang received his B.S. degree in Computer Science from Nanjing University in 2008. He is currently a PhD candidate in the Department of Computer Science and Technology, Nanjing University, Nanjing, China. He is also a member of the State Key Laboratory for Novel Software Technology. His research interests include delay tolerant networks, future Internet design, and service computing.

Zhuzhong Qian is an associate professor at the Department of Computer Science and Technology, Nanjing University, P. R. China. He received his PhD. Degree in computer science in 2007. Currently, his research interests include cloud computing, distributed systems, and pervasive computing. He is the chief member of several national research projects on cloud computing and pervasive computing. He has published more than 30 research papers in related fields.

Jie Wu is Chair and Laura H. Carnell Professor in the Department of Computer and Information Sciences at Temple University, USA. Prior to joining Temple University, he was a program director at the National Science Foundation and Distinguished Professor at Florida Atlantic University. His research interests include wireless networks, mobile computing, routing protocols, fault-tolerant computing, and interconnection networks. Dr. Wus publications include over 600 papers in scholarly journals, conference proceedings, and books. He has served on several editorial boards, including IEEE Transactions on Computers and Journal of Parallel and Distributed Computing. Dr. Wu was general co-chair for IEEE MASS-2006, IEEE IPDPS-2008, and IEEE DCOSS-2009 and was the program co-chair for IEEE INFOCOMM-2011. He served as general chair for IEEE ICDCS-2013. He was an IEEE Computer Society Distinguished Visitor and the chair for the IEEE Technical Committee on Distributed Processing (TCDP). Currently, Dr. Wu is an ACM Distinguished Speaker and a Fellow of the IEEE.

Sanglu Lu received her B.S., M.S., and Ph.D. degrees from Nanjing University in 1992, 1995, and 1997, respectively, all in computer science. She is currently a professor in the Department of Computer Science and Technology and the State Key Laboratory for Novel Software Technology. Her research interests include distributed computing, wireless networks, and pervasive computing. She has published over 80 papers in referred journals and conferences in the above areas. She is a member of IEEE.

Leah Epstein is an associate professor of mathematics at the University of Haifa, Israel. She received her PhD in computer science from Tel-Aviv University in 1999. Her main areas of research are bin packing and scheduling, paging and caching, graph problems, online algorithms, approximation algorithms, and algorithmic game theory. She has published over 100 journal articles, and a similar number of conference papers, and supervised many graduate students. She is a member of the editorial board of several journals, participated in many conference program committees, and was the program committee chair for the 20th European Symposium on Algorithms (ESA2012), track A.