Multi-Path Link Embedding for Survivability in Virtual Networks

Md Mashrur Alam Khan, Nashid Shahriar, Reaz Ahmed, and Raouf Boutaba, *Fellow, IEEE*

***Abstract*—Internet applications are deployed on the same network infrastructure, yet they have diverse performance and functional requirements. The Internet was not originally designed to support the diversity of current applications. Network virtualization enables heterogeneous applications and network architectures to coexist without interference on the same infrastructure. Embedding a virtual network (VN) into a physical network is a fundamental problem in network virtualization. A VN embedding that aims to survive physical (e.g., link) failures is known as the survivable VN embedding (SVNE). A key challenge in the SVNE problem is to ensure VN survivability with minimal resource redundancy. To address this challenge, we propose survivability in multi-path link embedding (SiMPLE). By exploiting path diversity in the physical network, SiMPLE provides guaranteed VN survivability against single link failure while incurring minimal resource redundancy. In case of multiple arbitrary link failures, SiMPLE provides maximal survivability to the VNs. We formulate this problem as an integer linear program and implement it using GNU linear programming kit. We propose a greedy proactive approach to solve larger instances of the problem in case of single link failures. In presence of more than one link failures, we propose a greedy reactive algorithm as an extension to the previous one, which opportunistically recovers the lost bandwidth in the VNs. Simulation results show that SiMPLE outperforms full backup and shared backup schemes for SVNE, and produces near-optimal results.**

***Index Terms*—Survivable virtual network embedding, fault tolerance, path splitting.**

I. Introduction

THE INTERNET has to support a wide range of applications having diverse performance and functional requirements. For example, Audio/Video streaming and large data transfers require dedicated bandwidth and bounded delay, online banking requires security guarantees, while Web browsing and email applications are satisfied with best-effort delivery. Currently, most of these applications are deployed on the same network infrastructure, and usually rely on the best-effort Internet's communication model without guarantees. Network Virtualization (NV) [1], [2] has been propounded as a promising solution for enabling heterogeneous applications and network architectures to coexist on the same physical infrastructure (or substrate network). NV involves two entities: Infrastructure Providers (InPs) and Service Providers (SPs). An InP owns and maintains the substrate, e.g., data centers. An SP, in contrast, requests network slices from one or more InP(s), and offers customized services to end users without significant investment in deploying and managing the substrate. An InP manages a network slice as a Virtual Network (VN), and embeds the VN to the Substrate Network (SN) with proper isolation and guaranteed Quality of Service (QoS). In some cases, a third entity, called Virtual Network Operator (VNO), acts as a broker between an SP and multiple InPs. A VNO accepts VN request from different SPs and installs, manages, and operates the VNs. In this way, NV enables multiple SPs to coexist on the same substrate without interference, and satisfies diverse application needs.

Efficient mapping of VNs onto an SN is known as the VN embedding (VNE) problem [3]. In its simplest form, the VNE problem is to map virtual nodes and links of a VN request onto substrate nodes and paths (sequence of physical links), respectively, while satisfying physical resource constraints. The VNE problem is $\mathcal{NP}$-hard and has been studied extensively in [4]–[7]. However, one important aspect of the problem that has received less attention is VN survivability. Finding a VN Embedding that can survive arbitrary substrate node or link failures is known as the Survivable Virtual Network Embedding (SVNE) problem [8]. A failure in the SN may cause multiple VNs to fail, which may significantly degrade service performance and availability. In many applications, a service outage can incur high penalty in terms of revenue and customer satisfaction. For example, online businesses in North America lost 26.5 billion in revenue due to service downtime in 2010 [9]. Hence, VN survivability is crucial for both InPs and SPs.

Survivability has been thoroughly investigated in non-virtualized networks in the past [10]–[13]. However, these solutions focus on ensuring that all nodes in the VN are connected by a path. In contrary, the focus in SVNE is to preserve both connectivity and the demanded bandwidth in VN. Hence, existing solutions are not directly applicable to SVNE. Survivability of VNs is usually achieved through allocation of redundant (i.e., backup) resources, which introduces additional challenges to the VNE problem. First, the failure characteristics and repair time are unpredictable [14], [15]. Reserving the full demand of a virtual link as backup is expensive, since backup resources remain idle when there are no failures [8]. To minimize resource wastage, shared backup schemes have been proposed in [16]. However, they do not guarantee the

Manuscript received December 10, 2015; revised March 23, 2016; accepted April 14, 2016. Date of publication April 27, 2016; date of current version June 8, 2016. The associate editor coordinating the review of this paper and approving it for publication was A. Pras.

The authors are with the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: mmalamkh@uwaterloo.ca; nshahria@uwaterloo.ca; r5ahmed@uwaterloo.ca; rboutaba@uwaterloo.ca).

Digital Object Identifier 10.1109/TNSM.2016.2558598

full requested bandwidth of a virtual link during failure. As such, it is challenging to determine the minimum redundancy level for guaranteed survivability. Second, primary and backup resources need to be disjoint in the SN. Embedding each virtual link into multiple disjoint paths mitigates the impact of failures [17], [18]. Although effective, this approach incurs path splitting overhead including packet redirection, increased routing table size, and packet reordering. In general, it is difficult to find the optimal trade-off between VN survivability, redundancy level, and path splitting overhead.

To address these challenges, we propose SiMPLE for ensuring **S**urvivability **i**n **M**ulti-**P**ath **L**ink **E**mbedding. SiMPLE presents a multi-path link embedding strategy by exploiting the path diversity in the SN. Studies in [14] and [15] have shown that link failures are more frequent than node failures, and node failures can be modeled as multiple link failures [19]. Hence, SiMPLE focuses on survivability against arbitrary substrate link failures. The major contributions of this paper can be summarized as follows:

- **Key Concept.** We propose a novel concept to ensure high survivability against multiple link failures while reserving only a fraction of the virtual link's demand as backup. To the best of our knowledge, SiMPLE is the only approach that provides provable survivability guarantee in presence of a single link failure without allocating full bandwidth of the virtual link's demand as backup.
- **Optimization Model.** The design goal of SiMPLE is to find a trade-off between maximizing survivability and minimizing redundant resources and path splitting overhead, which has not been considered in the previous studies. We formulate this joint optimization problem as an Integer Linear Program (ILP) to achieve this trade-off.
- **Proactive Approach.** We implement the ILP model in GLPK to find optimal solutions for small scale networks. For larger instances of the problem, we propose a greedy algorithm that produces near-optimal solutions. We demonstrate SiMPLE's effectiveness through extensive simulations and comparison with full backup and shared backup schemes for SVNE. Simulation results show that SiMPLE provides better survivability, requires lesser backup bandwidth, and generates more profit.
- **Reactive Approach.** While the proactive approach guarantees survivability against a single substrate link failure, it may not offer full protection for multiple substrate link failures. To mitigate the impact of such scenarios, we present a reactive approach in SiMPLE. This approach recovers the lost bandwidth in each virtual link affected by physical link failures. Simulation results show that, compared to the proactive approach, it improves the results by a significant margin in both minimizing failed VNs and obtaining higher profits for the InP.

This paper extends the results presented in our earlier paper [20] in several ways. First, we identify a failure case in our previous approach, and theoretically prove that it can occur with a significant probability in moderate size networks. Second, we include a reactive survivability mechanism to solve this problem. Third, while the previous paper was mostly focused on data center networks, we also focus on ISP networks in this paper. Fourth, we evaluate our proposed solutions in further details with additional experimental results. Fifth, we present a more detailed survey of virtual network embedding and survivability in this paper.

The rest of the paper is organized as follows. Section II provides necessary background. Section III presents the main concept and ILP model for SiMPLE. Section IV presents two greedy solutions – one proactive and one reactive – for link embedding in SVNE. Section V presents our evaluation results, and Section VI discusses related literature. Finally, Section VII concludes the paper with an outline of possible future research directions.

II. Background

In this section, we present the VNE problem and the existing mechanisms for ensuring survivability in VNE process.

A. Virtual Network Embedding

To describe the VNE problem, we model the SN and the VN as weighted graphs $G^S(N^S, E^S)$ and $G^V(N^V, E^V)$, respectively. Here, N^S and E^S denote the sets of the Substrate Nodes (SNodes) and Substrate Links (SLinks), respectively, while N^V and E^V denote the sets of Virtual Nodes (VNodes) and Virtual Links (VLinks), respectively. Each SNode $n^s \in N^S$ has a CPU capacity, $c(n^s)$, and each SLink $e^s \in E^S$ has a bandwidth capacity, $b(e^s)$. Similarly, the CPU demand of a VNode $n^v \in N^V$ and bandwidth demand of a VLink $e^v \in E^V$ are denoted by $c(n^v)$ and $b(e^v)$, respectively. The residual CPU and bandwidth resources at n^s and e^s are represented by $r(n^s)$ and $r(e^s)$, respectively. Generally, the VNE problem can be divided into two stages:

1) Node Embedding: Each VNode $n^v \in N^V$ from a VN request is mapped to a single SNode by a node mapping function: $\xi_N : N^V \rightarrow N^S$, subject to CPU capacity constraints: $\forall n^v \in N^V : c(n^v) \leq r(\xi_N(n^v))$.

2) Link Embedding: Each VLink $e^v \in E^V$ is mapped to a substrate path $p^{e^v} \in P^{e^v}$ between ingress SNode, $\xi_N(e^v_s)$ and egress SNode, $\xi_N(e^v_d)$, where e^v_s and e^v_d denote the source and destination VNodes of e^v, respectively. The link mapping function is $\xi_E : E^V \rightarrow P^{e^v}$, subject to bandwidth capacity constraint: $\forall e^v \in E^V \wedge \forall p^{e^v} \in P^{e^v} : b(e^v) \leq r(p^{e^v})$, where $r(p^{e^v}) = \min_{e^s \in p^{e^v}} r(e^s)$.

Solving the VNE problem is $\mathcal{NP}$-hard, as it is related to the multi-way separator problem [6]. Even with a given VNode mapping, the problem of optimally allocating the VLinks to substrate paths reduces to the unsplittable flow problem [21], and is thus $\mathcal{NP}$-hard as well. Fig. 1 depicts the mapping of the two VN requests, G^{V1} and G^{V2} (on the left) on an SN, G^S (on the right). Here, the SNodes and VNodes are labeled with letters inside the corresponding node. Node mapping for G^{V1} is $\xi^1_N(a) = D$, $\xi^1_N(b) = A$, $\xi^1_N(c) = F$, and link mapping is $\xi^1_E(ab) = DBA$, $\xi^1_E(ac) = DEF$, $\xi^1_E(bc) = ACF$; while G^{V2} has node mapping $\xi^2_N(e) = D$, $\xi^2_N(d) = G$, $\xi^2_N(f) = F$, and link mapping $\xi^2_E(ed) = DG$, $\xi^2_E(ef) = DEF$.

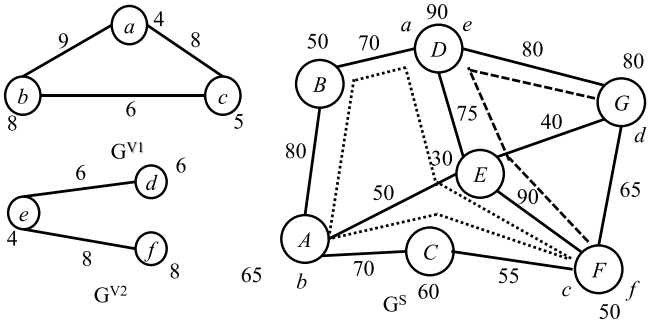


Fig. 1. Embedding VN requests onto a substrate.

B. Survivable Virtual Network Embedding

An SLink may not operate properly all the time due to various reasons such as fiber cut, maintenance, misconfiguration, and so on [14], [15]. To see the impact of such failure, let us consider a failure in SLink *DE* in Fig. 1. It causes the VLinks *ac* and *ef* to fail. In general, survivability against SLink failures can be achieved by either of the following ways:

1) Allocating Backup Resources: To survive against SLink failures, backup resource can be allocated in two ways [22], namely, SLink protection and path protection. In SLink protection, a primary path p^{e^v} is associated to each VLink, and each SLink $e^s \in p^{e^v}$ is protected by a detour. Upon an SLink failure, traffic on that SLink is locally rerouted through its detour. In Fig. 1, SLink *DE* can be associated with two detours *DGE* and *DBAE* for the VLinks *ac* and *ef*, respectively. In case of the path protection, each end-to-end primary path p^{e^v} is protected by an SLink disjoint backup path from source to destination. The source activates the backup path when it is notified about the failure of an SLink along path p^{e^v}. In Fig. 1, the bandwidth demanded by VLinks *ac* and *ef* can be reserved in the backup paths *DGF* and *DBACF*, respectively, which are SLink disjoint to the primary path *DEF*. Hence, redundant bandwidth has to be allocated in SN for each backup path. However, multiple backup paths can share the same backup bandwidth to minimize redundancy.

2) Multipath Embedding: Path splitting is a routing strategy to allow a single data stream to be split across multiple paths. Various path splitting techniques, such as, Equal-cost Multipath (ECMP) and Multipath TCP (MPTCP), have been used in the IP and TCP layers, respectively. Authors in [18] introduced path splitting in VNE to embed a VLink over multiple substrate paths. Multipath embedding mitigates the impact of failure by switching the affected traffic on the failed SLink to alternate paths [17]. In the worst case, it can salvage a fraction of the VLink's bandwidth during an SLink failure. In Fig. 1, we can embed VLinks *ac* and *ef* onto multiple paths such as $\xi_E^1(ac) = \{DEF, DGF\}$ and $\xi_E^2(ef) = \{DEF, DBACF\}$. Here, upon the failure of the SLink *DE*, both VLinks *ac* and *ef* can partially survive through the alternate paths *DGF* and *DBACF*, respectively.

III. SVNE Through Path Splitting

In this section, we first describe the main concept of SiMPLE. Then we present its ILP formulation.

A. The SiMPLE Embedding Concept

The main concept of SiMPLE embedding consists of two parts. The first part is a proactive approach, which embeds each VN as they arrive. The second part is a reactive recovery mechanism, which works after the arrival of each SLink failure, and recovers each affected VN.

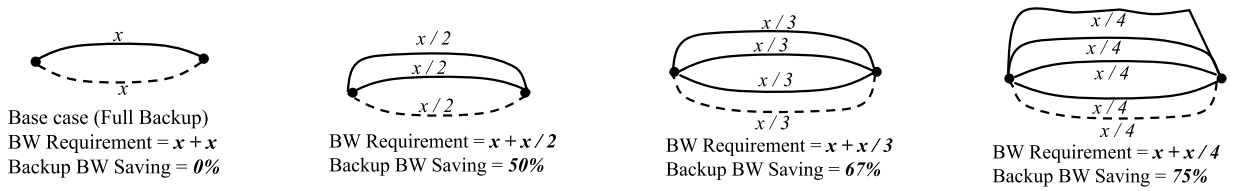
1) Proactive Approach: The proactive embedding concept of SiMPLE is illustrated in Fig. 2. A basic proactive approach for SVNE, the Full Backup Scheme (FBS), is illustrated in Fig. 2a. In this case, a VLink with demand x is embedded onto two disjoint paths with sufficient residual capacity. One of the paths acts as the primary (denoted with solid line), whereas the other is reserved for the backup (dashed line). When an SLink in the primary path fails, the backup path serves the VLink traffic. When the failed SLink recovers, the primary path starts serving the VLink again. However, such technique provisions twice the demand of each VLink. As a result, the number of accepted VNs and SLink utilization decreases significantly.

SiMPLE operates according to Fig. 2b – Fig. 2d. In Fig. 2b, the VLink is split into three disjoint substrate paths, and $x/2$ bandwidth is allocated to each of them. In this case, two paths are used to carry the primary flow, whereas the third path is used as backup. Since these paths are disjoint, at most one of them can be affected by a single SLink failure. If an SLink fails, the two unaffected paths deliver the requested bandwidth x. Note that only half of the requested bandwidth is allocated in the backup path, or, in other words, 50% backup bandwidth is saved in contrast to FBS. We can extend this idea to a higher number of splits, say k. Fig. 2c and Fig. 2d present the VLink embedding scenario for $k = 4$ and 5, respectively. As highlighted in these figures, 67% and 75% backup bandwidth is saved in these two cases, respectively. In addition, the splitting of each VLink into multiple substrate paths improves the possibility of VN request acceptance; even if the full requested bandwidth is not available in any of the SLinks, a VLink can be embedded by splitting the required bandwidth over multiple paths. In other words, it utilizes the links more efficiently than FBS, and increases the number of accepted VNs. However, increasing the number of splits introduces additional overhead, which must be taken into consideration.

There is a trade-off between the number of splits, and VNE overhead. Indeed, each path splitting has a cost in terms of routing entry updates, source and destination buffers, and additional SLink delays. We formulate these costs mathematically in Section III-B. If we increase the number of splits too much, these costs may result into infeasible VN embeddings. For an SN with small SLink to SNode ratio, SiMPLE will perform similar to FBS.

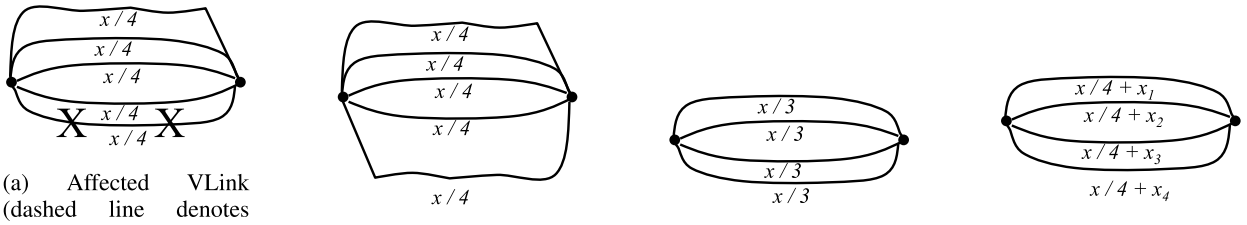
Theorem 1: SiMPLE proactive embedding guarantees to preserve the full demand of every embedded VLink in case of a single SLink failure.

Proof: We prove this Theorem by contradiction. Assume that a VLink $\tilde{e}^v \in E^V$ is not supported with its full demand. According to the SiMPLE working principle, at least two paths $p_1^{\tilde{e}^v}$ and $p_2^{\tilde{e}^v}$ in $\tilde{e}^v$ are impacted by a single SLink failure. By definition, $p_1^{\tilde{e}^v}$ and $p_2^{\tilde{e}^v}$ are disjoint,



(a) Two splits (b) Three splits (c) Four splits (d) Five splits

Fig. 2. Proactive Embedding in SiMPLE.



(a) Affected VLink (dashed line denotes failed path) (b) Provisioning new path (c) Fixed allocation (d) Variable allocation

Fig. 3. Reactive Failure Recovery in SiMPLE.

(i.e., they have no common SLink), and this leads to a contradiction. ■

Theorem 2: While embedding VNs with same characteristics (e.g., size, demand, and arrival rate), SiMPLE outperforms FBS in the number of accepted VNs by a factor of $2\left(\frac{k-1}{k}\right)$, where k is the average number of splits in embedding a VLink.

Proof: Assume that FBS and SiMPLE are evaluated for time T, and accepted a series of n VNs. Each VN has v VLinks, and each VLink demands x bandwidth. At time T, substrate bandwidth consumptions of FBS and SiMPLE are given by $\mathcal{B}_F^T = 2n\bar{l}vx$ and $\mathcal{B}_S^T = kn\bar{l}v\frac{x}{k-1}$, respectively, where $\bar{l}$ is the average length of the substrate paths used in embedding. The additional substrate bandwidth used in FBS is given by $\mathcal{B}_F^T - \mathcal{B}_S^T = n\bar{l}vx\frac{k-2}{k-1}$. Before SiMPLE consumes bandwidth $\mathcal{B}_F^T$, an additional $\hat{n}$ VNs with same characteristics would occupy $\hat{\mathcal{B}}_S^T$ bandwidth, where $\hat{\mathcal{B}}_S^T = k\hat{n}\bar{l}v\frac{x}{k-1}$. Since $\hat{\mathcal{B}}_S^T = \mathcal{B}_F^T - \mathcal{B}_S^T$, we obtain, $\hat{n} = n\frac{k-2}{k}$. Hence, the number of accepted VNs in SiMPLE is $n + \hat{n} = 2\frac{k-1}{k}n$. Note that this theorem presents the upper bound for the number of accepted VNs. Hence, we consider VNs with similar characteristics. ■

2) Reactive Approach: The proactive approach described in the previous paragraph guarantees survivability of each VLink in case of a single SLink failure. However, this approach can be vulnerable in presence of multiple SLink failures, especially when they arrive within a very short time window. This is because multiple failures can affect different paths in a single VLink.

Theorem 3: In SiMPLE Proactive Embedding, if multiple SLink failures are present in a moderate size substrate network, then they affect the same VLink with a probability between $2 - 16\%$.

Proof: For simplicity, assume that each VN share the same characteristics (e.g., size, demand, and arrival rate). In a saturated SN, the expected number of VLinks embedded onto an SLink is α. The first SLink failure will affect α VLinks, each of which have $(k-1)l$ critical SLinks, where k and l are the average number of splits and average path length, respectively.

TABLE I
APPROXIMATE VALUES FOR THEOREM 3

Term	Value
Number of Splits, k	$4-5$
Average Path length, l	$4-5$
Average Number of VLink per SLink, α	$3-4$
Number of Substrate Links, L	500

In one extreme, these $\alpha(k-1)l$ can be shared among $(k-1)l$ physical links, and in other, they may be totally disjoint. The probability of a second failure in one of these SLinks can be between $\frac{(k-1)l}{L}$ and $\frac{\alpha(k-1)l}{L}$, where L is the total number of SLinks. We conducted experiments to determine approximate values for the above terms, which are given in Table I. By substituting these values, we estimate that this probability lies within $2 - 16\%$. ■

To handle the impact of multiple failures, we propose a reactive approach in SiMPLE. This approach, as presented in Fig. 3, works for each VLink that are affected by an SLink failure. In essence, this approach considers each possibility to recover an affected VLink (a VLink with a failed path, see Fig. 3a), and selects the most feasible one. The first possibility, as shown in Fig. 3b, is to provision another link-disjoint path for the affected VLink. The second and third possibilities, as shown in Fig. 3b and 3c respectively, considers increasing the lost bandwidth by a fixed or variable amount among the other working paths. Among these possible alternatives, SiMPLE considers a set of certain criteria (e.g., amount of physical resources used, load balancing) to evaluate their goodness, or *cost*. A detailed description of the cost function can be found in Section III-B. The embedding contributing to the lowest cost is chosen by SiMPLE.

Note that, in the worst case, none of these possibilities will work. This implies that another link disjoint path does not exist, and the existing paths do not have enough residuals to support the lost bandwidth. In this case, the VLink will be served with less than 100% of its demand. Nonetheless, we argue that this case can only arise when a highly congested

network is subject to a massive number of failures in a very short time, which is often less frequent. As a basis of the argument, we present the study in [15], where it is evident that at most 5% of the SLink failures occur in presence of another SLink failure.

B. ILP Formulation

We use the following notations to represent different aspects of embedding. The set P^{e^v} represents a set of disjoint paths $\{p_1^{e^v}, p_2^{e^v}, \ldots, p_k^{e^v}\}$ in SN where e^v is embedded. Note that the number of paths in P^{e^v} will be equal to the number of splits for $e^v \in E^V$, i.e., $|P^{e^v}| = k^{e^v}$. Two boolean variables are defined as follows.

$$X(n^v, n^s) = \begin{cases} 1, & \text{if } n^v \text{ is embedded to } n^s \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$Y(p_i^{e^v}, e^s) = \begin{cases} 1, & \text{if the path } p_i^{e^v} \text{ contains } e^s \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

We formulate SiMPLE as an ILP Model, since it involves integer (binary) variables as well as linear constraints. In this model, we optimize both the number of splits and the set of substrate paths for each VLink of a VN such that the overall embedding cost is minimized. Afterwards, the corresponding VLinks are mapped to optimal sets of paths. The VN embedding cost has the following three components.

1) Split and Join Cost: The first cost is the split and join cost at the source and destination SNodes for a VLink e^v. In SiMPLE, we assume that the SN supports path splitting, and this assumption relies on the substrate switches. This is because each data stream is *split* at the ingress switch, and subsequently *joined* at the egress switch.[1] Let $d_1(n^s, k)$ and $d_2(n^s, k)$ be the splitting and joining costs into k branches at $n^s \in N^S$. The total split and join cost at n^s is denoted by $D(n^s, k) = d_1(n^s, k) + d_2(n^s, k)$. We can represent the total split and join cost as follows in (3)

$$\ddot{I}\left(e^v, P^{e^v}, k^{e^v}\right) = \left(D\left(\xi_N\left(e_s^v\right), k^{e^v}\right) + D\left(\xi_N(e_d^v), k^{e^v}\right)\right). \quad (3)$$

2) Switching Cost: The second cost is the packet switching cost, and it is presented in (4) as $\ddot{S}(e^v, p_i^{e^v})$. This cost is associated with each mapped path of e^v due to forwarding the fragmented data stream between the source and destination SNodes. For such a path $p_i^{e^v} \in P^{e^v}$, all intermediate SNodes forward each flow to the next appropriate SNode.[1] The switching cost at $n^s \in N^S$ is denoted by $\beta(n^s)$

$$\ddot{S}(e^v, p_i^{e^v}) = \sum_{n^s \in p_i^{e^v}} \left(\frac{c(n^s)}{r(n^s)} \beta(n^s)\right). \quad (4)$$

3) SLink Cost: The third and final cost component, SLink cost, is given by $\ddot{L}(e^v, p_i^{e^v})$ in (5). This cost represents the sum of allocated substrate bandwidth cost and accumulated delays along the SLinks on $p_i^{e^v}$. This cost is also defined for each mapped path $p_i^{e^v} \in P^{e^v}$ for e^v. In (5), the term w^E represents the relative weight of the SLink delay ($\delta(e^s)$) (in time units) compared to the allocated bandwidth cost (in bandwidth units). In today's data center networks, the link delay is usually very small. For this reason, we suggest that w^E should take a fractional value less than one.

$$\ddot{L}\left(e^v, p_i^{e^v}, k^{e^v}\right) = \sum_{e^s \in p^{e^v}} \left(\frac{b(e^s)}{r(e^s)} \frac{b(e^v)}{k^{e^v} - 1} + w^E \delta\left(e^s\right)\right) \quad (5)$$

A goal in our ILP model is to ensure proper load balancing across SNodes and SLinks. To this end, each SNode and SLink is associated with a non-linear weight function that produces low values for under-utilized SNodes and SLinks, while weight function value increases rapidly as an SNode's or SLink's utilization approaches saturation. The fractions $\frac{c(n^s)}{r(n^s)}$ and $\frac{b(e^s)}{r(e^s)}$ give higher privilege to less loaded SNodes and SLinks, respectively, over the saturated ones. Therefore, in (4) and (5), these two fractions are chosen as the *load balancing factors* for SNodes and SLinks, respectively. The possible alternates, e.g., $(1 - \frac{r(n^s)}{c(n^s)})$ and $(1 - \frac{r(e^s)}{b(e^s)})$, have a linear relation between utilization and demand, and so cannot be used for our purpose.

Now we introduce the SiMPLE objective function. The goal is to minimize the cost presented in (6). In this equation, $\ddot{I}$ and $\ddot{S}$ have units in MIPS (for split, join, switching costs involving CPU resources), whereas $\ddot{L}$ has Mbps unit. To unify these different units, we multiply the split, join, and switching costs with a weight, w^N. Furthermore, in comparison with the bandwidth resources, the CPU resources are cheaper and more available. Therefore, we propose that w^N should be a fraction. In this process, we prioritize bandwidth in the cost function above other resources.

`SiMPLE_ILP`:

$$minimize \left[\sum_{e^v \in E^V} \left(\begin{array}{r} \ddot{I}(e^v, P^{e^v}, k^{e^v}) w^N \\ + \sum_{p_i^{e^v} \in P^{e^v}} \left(\begin{array}{l} \ddot{S}(e^v, p_i^{e^v}) w^N + \\ \ddot{L}(e^v, p_i^{e^v}, k^{e^v}) \end{array} \right) \end{array} \right)\right] \quad (6)$$

The constraints for `SiMPLE_ILP` are presented in (7) - (12). SNode and SLink capacity constraints are presented in (7) and (8), whereas VNode demand constraint is given by (9). Constraint (10) ensures that a VNode is mapped to exactly one SNode. Path disjointness constraint is presented in (11). Constraint (12) ensures that a total of k^{e^v} paths are found.

$$\forall n^s \in N^S : \sum_{n^v \in N^V} c\left(n^v\right) \times X\left(n^v, n^s\right) \leq c\left(n^s\right) \quad (7)$$

$$\forall e^s \in E^S : \sum_{e^v \in E^V} \frac{b(e^v)}{k^{e^v} - 1} \times Y\left(p^{e^v}, e^s\right) \leq b\left(e^s\right) \quad (8)$$

$$\forall e^v \in E^V : Y\left(P^{e^v}, e^s\right) \times \frac{b(e^v)}{k^{e^v} - 1} \leq r\left(e^s\right) \quad (9)$$

$$\forall n^v \in N^V : \sum_{n^s \in N^S} X\left(n^v, n^s\right) = 1 \quad (10)$$

$$\forall e^v \in E^V : \sum_{p_i^{e^v} \in P^{e^v}} Y\left(p_i^{e^v}, e^s\right) \leq 1 \quad (11)$$

$$\forall e^v \in E^V : \sum_{p_i^{e^v} \in P^{e^v}} \sum_{e^s \in p_i^{e^v}} \frac{1}{|p|} \times Y\left(p_i^{e^v}, e^s\right) = k^{e^v} \quad (12)$$

[1] Without loss of generality and for simplifying the formulation, we do not place any cap on the number of splits, joins, or switchings per SNode.

Algorithm 1 SiMPLE Proactive Allocation, `SiMPLE-PR`

1: **function** `SiMPLE-PR`(G^S, G^V, ξ_N)
2: **for all** $e^v \in E^V$ **do**
3: $\forall k \in \{2,3,4,5\} : P^k \leftarrow \phi \wedge Cost(P^k) \leftarrow \infty$
4: **for** $k \in \{2,3,4,5\}$ **do**
5: $\mathbb{E}^S \leftarrow E^S$
6: **for** $j \leftarrow 1, k$ **do**
7: $Q \leftarrow$ `Dijkstra` $(N^S, \mathbb{E}^S, \xi_N(e_s^v), \xi_N(e_d^v), \frac{b(e^v)}{k-1})$
8: $P^k \leftarrow P^k \cup Q$
9: $\mathbb{E}^S \leftarrow \mathbb{E}^S - P^k$
10: **end for**
11: **end for**
12: $P^* \leftarrow \min(P^2, P^3, P^4, P^5)$
13: **if** $Cost(P^*) = \infty$ **then**
14: **return** ϕ
15: **end if**
16: $\xi_E(e^v) \leftarrow P^*$
17: **end for**
18: $\forall e^s \in E^S \cap P^* :$ `update` $r(e^s)$
19: **return** ξ_E
20: **end function**

Algorithm 2 SiMPLE Reactive Recovery, `SiMPLE-RE`

1: **function** `SiMPLE-RE`(G^S, e^v, ξ_N)
2: $P^{e^v} \leftarrow P^{e^v} - \{p_f^{e^v}\}$
3: $P^1 \leftarrow$ `FindNewPath`(G^S, e^v)
4: $P^2 \leftarrow \forall p^{e^v} \in (P^{e^v} - p_f^{e^v}) :$ `alloc` $(p^{e^v}, \frac{b(e^v)}{k^{e^v}-1})$
5: $P^3 \leftarrow \forall p^{e^v} \in (P^{e^v} - p_f^{e^v}) :$ `add` $(p^{e^v}, b(p_f^{e^v}) \cdot \frac{r(p^{e^v})}{\sum r(p^{e^v})})$
6: $P^* \leftarrow \min(P^1, P^2, P^3)$
7: $\xi_E(e^v) \leftarrow P^*$
8: $\forall e^s \in E^S \cap P^* :$ `update` $r(e^s)$
9: **return** ξ_E
10: **end function**
11: **function** `FindNewPath`(G^S, e^v)
12: $\mathbb{E}^S \leftarrow E^S - P_{new}^{e^v}$
13: $P^{e^v} \leftarrow P^{e^v} \cup$ Dijkstra $(N^S, \mathbb{E}^S, \xi_N(e_s^v), \xi_N(e_d^v))$
14: **return** P^{e^v}
15: **end function**

IV. Proposed Solutions

The ILP model presented in Section III can find optimal solution for small instances of the multi-path embedding problem, but it will not scale with SN and VN size. In this section, we propose two scalable greedy algorithms for each part in SiMPLE, as introduced in Section III-A. The first one is a survivable multi-path proactive embedding approach, and the second one is a recovery mechanism that operates after each SLink failure. Both solutions operate under the assumption that the node mapping has already been done, possibly using one of the greedy approaches (e.g., First Fit or Best Fit approach [23]).

A. Proactive Solution

The proactive solution, `SiMPLE-PR`, embeds each VLink of a VN request as it arrives. The algorithm iteratively computes a set of disjoint paths for each VLink, and returns the result of embedding, or ϕ if none exists.

The input to `SiMPLE-PR`, as presented in Algorithm 1, is an SN G^S, a VN G^V, and its node mapping function, ξ_N. In `SiMPLE-PR`, we split each VLink into no more than five paths. This is because, as illustrated experimentally in Section V-H, a higher number of splits will cause a very high splitting, joining, routing and delay overheads, which will eventually make the embedding expensive and infeasible. `SiMPLE-PR` iteratively works on each VLink of a newly arrived G^V (Lines 2 – 19). The set P^k (initially empty) denotes the set of candidate paths selected for split k, where $2 \leq k \leq 5$ and $k \in N$ (Line 3). At each iteration of k (Lines 4 – 11), `SiMPLE-PR` runs the Dijkstra's weighted shortest path algorithm to select a candidate path with the sufficient residuals ($b(e^v)$ / $(k-1)$) between the source and destination SNodes of the corresponding VLink (Line 7). This path is added to P^k (Line 8). To maintain the disjointness constraint, the SLinks of the path are temporarily removed from G^S (Line 9). After the end of this loop, the discarded SLinks are restored (Line 5), and the set of paths with the minimal cost, P^*, is calculated (Line 12). If no such set is found (i.e., cost of P^* is ∞), `SiMPLE-PR` finds no feasible mapping for this VLink (and hence G^V) and returns ϕ (Lines 13 – 15). Otherwise, it updates the link mapping function ξ_E (Line 16), and moves on to process the next VLink. If the mapping of all the VLinks are found in this process, `SiMPLE-PR` returns ξ_E (Line 19). In this case, G^V is embedded onto G^S, and the residual capacities in the corresponding SLinks are updated (Line 18).

Theorem 4: The running time of `SiMPLE-PR` is $O(|E^V| \times (|N^S| \cdot log|N^S| + |E^S|))$.

Proof: As mentioned in the previous paragraph, `SiMPLE-PR` iteratively embeds each VLink of a VN. For each VLink, it runs Dijkstra's algorithm $O(1)$ times. Since, for i splits, Dijkstra's algorithm is invoked i times, where $2 \leq i \leq 5$. In total, Dijkstra's algorithm is called at most $2+3+4+5 = 14$ times. The running time of Dijkstra's algorithm is $O(|N^S| \cdot log|N^S| + |E^S|)$. The later operations, e.g., temporarily discarding the SLinks in the selected path, take $O(|E^S|)$ running time. At the end of the iteration, updating the residual capacities of the SLinks in the optimal path P^* also takes $O(|E^S|)$ running time. Therefore, a single VLink is embedded in a running time $O((|N^S| \cdot log|N^S| + |E^S|))$. In total, `SiMPLE-GR` embeds a VN in a running time of $O(|E^V| \times (|N^S| \cdot log|N^S| + |E^S|))$. ■

B. Reactive Solution

The reactive recovery mechanism, `SiMPLE-RE`, performs on each SLink failure as they arrive. As briefly discussed in Section III-A2, `SiMPLE-RE` recovers a failed path in an affected VLink by exploring different recovery strategies. In the end, it returns the recovery embedding with lowest cost, which minimizes the physical resource consumption while considering load balancing.

`SiMPLE-RE` is briefly presented in Algorithm 2. The input to this algorithm is the substrate network, G^S, and the affected

virtual link, e^v, and the node mapping function, ξ_N. At first, this algorithm discards the failed path, $p_f^{e^v}$, from the existing paths in e^v, P^{e^v} (Line 2). Then, it calls the `FindNewPath` method (Line 3), which finds another link-disjoint path, adds it to P^{e^v} (Lines $11-14$), and returns this set into P^1 (Line 3). Next, `SiMPLE-RE` allocates the fixed amount of bandwidth in the existing paths, and stores the result in P^2 (Line 4). Afterwards, `SiMPLE-RE` recovers the lost bandwidth, $b(p_f^{e^v})$, among the existing paths, P^{e^v}. For this purpose, it splits $b(p_f^{e^v})$ in proportion to residuals of the paths in P^{e^v}, adds this split bandwidth to these paths, and saves this set into P^3 (Line 5). After that, `SiMPLE-RE` chooses the embedding with lowest cost, P^*, among P^1, P^2 and P^3 (Line 6). In the end, the new embedding of e^v is changed to P^* (Line 7), the residuals of the substrate network is updated accordingly (Line 8), and the resultant embedding is returned (Line 9).

Theorem 5: The running time of `SiMPLE-RE` is $O(|N^S|log|N^S| + |E^S|)$.

Proof: Note that the steps presented in Lines $3-5$ in `SiMPLE-RE` are independent of each other, and can be parallelized. Provisioning a new path, as presented in the `FindNewPath` method (Lines $11-14$) invokes Dijkstra's shortest path algorithm, and this can be run in $O(|N^S|log|N^S|+|E^S|)$ time. The fixed and variable bandwidth allocations, as presented in Lines $3-4$, are linear operations in the number of physical links, $|E^S|$. Line 6 performs finding the minima among three possible alternatives embeddings, and this is a constant-time operation. Lines $7-8$ are also linear in $|E^S|$, since they involve updating the virtual links and substrate links, respectively. As a result, the running time of `SiMPLE-RE` is determined by the `FindNewPath` method call in Line 3, which is $O(|N^S|log|N^S| + |E^S|)$. ■

V. Performance Evaluation

A. Simulation Setup

We consider the online version of the SVNE problem, where each VN request is embedded as it arrives. We use the Fat tree topology [24] and a Synthetic topology to assess the behavior of SiMPLE in data center networks and ISP networks, respectively. The Synthetic topology connects each SNode at a low probability (≤ 0.1), which represents an arbitrary ISP network. To demonstrate the SiMPLE scalability, we present the results on VN embedding performance at small scale, and VN survivability at large scale. In small scale experiments, we evaluate both `SiMPLE-PR` and the optimal solution, `SiMPLE-OP`, where the later is an implementation of the ILP model presented in Section III-B using GLPK. This ILP model finds an optimal embedding for all VLinks of a VN request. To reduce the solution space, the GLPK implementation considers the first 200-shortest loop-less paths between a pair of SNodes, computed using Yen's Algorithm [25]. However, for the large instances, GLPK exceeds memory limits and is unable to find any solution. Also note that we evaluate `SiMPLE-RE` only at the survivability experiments, since its major focus is failure recovery.

For all experiments, VN requests are generated by varying their size randomly. We use Poisson process to model VN arrival and SLink failure events. The VN lifetime is modeled using a Geometric distribution. It is worth noting that, our simulation setup, choice of simulation parameters, VN and failure arrival distributions, and VN lifetime distributions summarized in Table II are similar to the previous works [8], [26]. In Table II, $[x_{min}, x_{max}]$ denotes a uniform distribution between x_{min} and x_{max}. $Pois\{p\}$ and $Geo\{g\}$ stand for the Poisson and Geometric distributions with mean p and g, respectively. For our experiments, we use random node mapping, which is less informed and thus makes the VLink embedding more challenging than the systematic node mapping approaches (e.g., First Fit [23]). However, the basic constraints in VNE ((7), (9), (10)) were satisfied by this random node mapping algorithm. We run our experiments on a workstation with AMD III+ FX-6100 X6 processor, 16GB DDR3 Non-ECC SDRAM, and Windows 7 64 bit operating system.

TABLE II
Evaluation Environment

Characteristics	Small Scale	Large Scale
Number of SNodes	125 [2] 100 [4]	500 [3] 500 [4]
SNode Capacity	$[50, 150]$	$[10, 50]$
SNode Switching Cost	$[2, 7]$	$[2, 7]$
Number of SLinks	500 [2] 574 [4]	4000 [3] 4029 [4]
SLink Capacity	$[70, 80]$	$[70, 80]$
SLink Delay	$[3, 15]$	$[3, 15]$
Splitting Cost	10 per split	10 per split
Joining Cost	10 per join	10 per join
VNodes per VN	$[2, 6]$	$[2, 10]$
VNode Capacity	$[5, 20]$	$[5, 20]$
VLink Conn. Prob.	0.5	0.5
VLink Demand	$\alpha\%$ of $[70, 80]$	$[10, 20]$
Total Number of VNs	300	300
Total Simulation Time	15000	15000
VN Arrival Rate, λ_V	`Pois`$\{0.05\}$	`Pois`$\{0.05\}$
VN Lifeime	`Geo`$\{1000\}$	`Geo`$\{1000\}$
Failure Arrival Rate, λ_F	N/A	`Pois` $\{0.05 \times \gamma\}$
Failure Repair Time	N/A	`Geo`$\{7000\}$

We run our experiments under different levels of workload, α, defined as the percentage of the average VLink demand to the average SLink capacity. To observe the impact of different workloads, α is varied from 10% to 60%. Furthermore, since our focus is to mitigate SLink failures, we measure SiMPLE's ability to survive different failure levels, expressed as γ – the ratio of the failure rate to the VN arrival rate. In large scale experiments, we stress the SN with a lot of failures, even at a rate higher than the VN arrival rate. For this reason, γ is varied from 1 to 6. In addition, the mean time to repair (MTTR) is significantly higher than the mean VN lifetime (Table II) to magnify the impact of failures.

B. Baseline Algorithms

We compare SiMPLE to two proactive approaches, *Full Backup Scheme (FBS)* and *Shared Backup Scheme (SBS)*.

[2] 10-ary Fat tree topology.
[3] 20-ary Fat tree topology.
[4] Synthetic topology.

1) FBS: In FBS, the full demand of each VLink is mapped to two disjoint substrate paths, which are computed using Dijkstra's weighted shortest path algorithm. The shorter of these two paths act as primary, whereas the other path is reserved as backup.

2) SBS: The primary and backup path allocations in SBS [16] are similar to that in FBS. However, in contrast to FBS, multiple VLinks can share the same resources for their backup flows. When a failure occurs, the affected VLinks try to recover their full demand from the backup path. When multiple VLinks try to use the same backup link simultaneously, fair sharing policy is adopted.

C. Terminology

We use the following terms to analyze failure impacts.

1) Path Failure: A *path failure* event is defined as the failure of one (or, more) SLink(s) belonging to a specific path. At this state, the corresponding path cannot carry the flow from the source to the destination SNode.

2) Affected VLink: A VLink is affected by a SLink failure if and only if one (or, more) of its substrate paths fail(s). An affected VLink may still retain its full demand depending on the severity of failure. For example, both FBS and SiMPLE retain their full demand in presence of a single SLink failure.

3) Failed VLink and Failed VN: A VLink is failed if and only if all of its mapped substrate paths fail (i.e., when it meets 0% of its demand). A VN fails if and only if one (or, more) of its VLinks fail(s).

D. Performance Metrics

Unless otherwise mentioned, the symbols used in this Section have their usual meanings as described in Section III-B.

1) Profit, Ψ: We first define the revenue, $\Pi(G^V)$, for a VN as $\Pi(G^V) = c_1 \sum_{e^v \in E^V} b(e^v) + c_2 \sum_{n^v \in N^V} c(n^v)$. Here, c_1 and c_2 are application-specific constants that represent the relative importance of bandwidth and CPU. The profit of G^V is defined by $\Psi(G^V) = T(G^V) \times \big(\Pi(G^V) - Cost(G^V)\big)$. Here, $T(G^V)$ is the lifetime of G^V, and $Cost(G^V)$ represents the total substrate cost for G^V, as represented in (6). The overall profit is given by, $\Psi = \sum_{\forall G^V} \Psi(G^V)$.

2) Acceptance Ratio, $\mathbb{AR}$: It is the ratio of the number of accepted VNs in the system ($|\mathbb{Z}^{\mathbb{A}}|$) to the total number of arrived VN requests ($|\mathbb{Z}^{\mathbb{T}}|$). Formally, $\mathbb{AR} = |\mathbb{Z}^{\mathbb{A}}|/|\mathbb{Z}^{\mathbb{T}}|$, where $\mathbb{Z}^{\mathbb{A}} \subseteq \mathbb{Z}^{\mathbb{T}}$.

3) Average Fraction of Backup Bandwidth, $\hat{\mathbb{B}}$: For e^v, $\mathbb{B}^{e^v}$ is the ratio of its backup bandwidth allocation to its total bandwidth allocation, i.e., $\mathbb{B}^{e^v} = |p_b^{e^v}| / \sum_{p_i^{e^v} \in P^{e^v}} |p_i^{e^v}|$. Here, $|p_b^{e^v}|$ is the bandwidth consumption for backup path $p_b^{e^v}$. The average fraction of backup bandwidth is, $\hat{\mathbb{B}} = \underset{\forall e^v \in E^V}{Avg} (\mathbb{B}^{e^v})$.

4) Average Splitting Overhead, $\hat{\mathbb{S}}$: The average splitting overhead is given by the average of the total split, join, and switch cost for all VLinks, i.e., $\hat{\mathbb{S}} = \underset{e^v \in E^V}{Avg} (\ddot{I}(e^v, P^{e^v}, k^{e^v}) + \sum_{p_i^{e^v} \in P^{e^v}} \ddot{S}(e^v, p_i^{e^v}))$.

5) Average Fraction of Survived Bandwidth, $\hat{\mathbb{F}}$: Let $\tilde{E}^V \subseteq E^V$ denote the set of affected VLinks. For an *affected VLink* $\tilde{e}^v \in \tilde{E}^V$, $\mathbb{F}^{\tilde{e}^v}$ represents the ratio of the available bandwidth to its total demand. The average fraction of survived bandwidth ($\hat{\mathbb{F}}$) of the affected VLinks is given by, $\hat{\mathbb{F}} = \underset{\forall \tilde{e}^v \in \tilde{E}^V}{Avg} \big(\mathbb{F}^{\tilde{e}^v}\big)$.

6) Probability of Simultaneous VN Failures, Prob(ρ^i): Let ρ^i denote the event of i simultaneous VN failures, and τ_i be the duration of time for ρ_i. $Prob(\rho^i)$ is denoted as the ratio of its lifetime τ_i to total simulation time τ, i.e., $Prob(\rho^i) = \tau_i/\tau$.

7) Nine Availability: The availability of a system is often represented by the number of nines in its uptime probability; e.g., 1 or 2 nines imply that the probability of the system being available is 0.9 or 0.99, respectively [27]. We compute the nine availability of a failed VN, G^V, as $(-\log_{10} \omega(G^V))$, where $\omega(G^V)$ is the ratio of time G^V is in failed state to its lifetime.

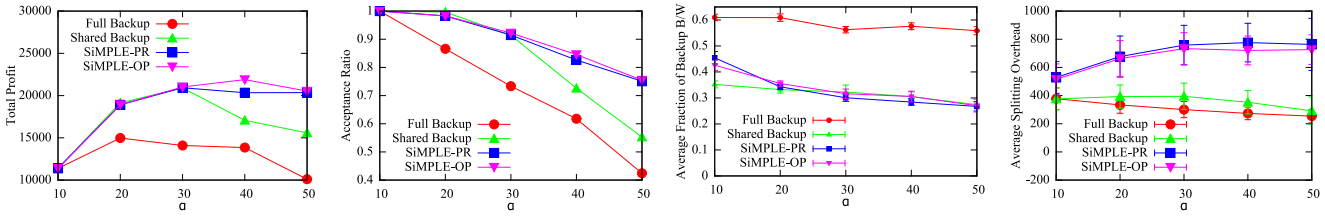
E. Performance Evaluation Results

We evaluate the VN embedding performances in all four schemes for Fat tree and Synthetic topologies.

1) Profit: In terms of Profit, `SiMPLE-PR` outperforms both FBS and SBS approaches, and is very close to the optimal result (`SiMPLE-OP`). Fig. 4a and Fig. 5a show the profits for different load (or, α) in the Fat tree and Synthetic topologies, respectively. As shown in these two figures, all approaches achieve similar profits for small load ($\alpha \leq 20$). However, at increased loads, the profits decrease for FBS and SBS. `SiMPLE-PR` achieves approximately $100 - 300\%$ and $50 - 120\%$ more profit than FBS and SBS, respectively.

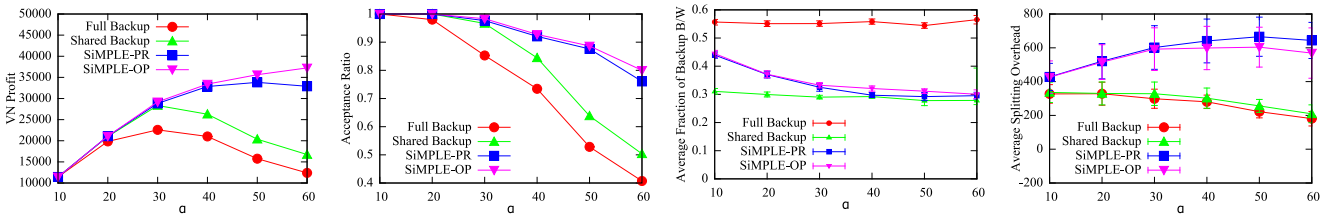
2) Acceptance Ratio: Results for the $\mathbb{AR}$ at different α are given in Fig. 4b and Fig. 5b, respectively. According to these results, `SiMPLE-PR` performs as good as FBS and SBS for small loads ($\alpha \leq 20$). However, at larger loads, $\mathbb{AR}$ of `SiMPLE-PR` exceeds the baseline approaches by roughly 20–100%, and lies very close to `SiMPLE-OP`.

3) Overhead: The overhead of the considered approaches are evaluated from two perspectives – backup bandwidth allocation and splitting overhead. `SiMPLE-PR` uses a very small fraction of the total allocated bandwidth resource as backup. Fig. 4c and Fig. 5c show $\hat{\mathbb{B}}$ with 95% confidence intervals for different α for Fat tree and Synthetic topologies, respectively. These two figures show that FBS uses more than half of its resources for backup, regardless of α and substrate topology. On the contrary, $\hat{\mathbb{B}}$ is relatively smaller for both `SiMPLE-PR` and `SiMPLE-OP`. The value $\hat{\mathbb{B}}$ for SBS is always small for all α, since SBS allows sharing the same backup resource between multiple VLinks, and thus does not guarantee survivability unlike SiMPLE. However, for heavier loads, SiMPLE uses approximately $40-50\%$ less backup bandwidth than FBS, and performs very close to SBS. The splitting overhead, $\hat{\mathbb{S}}$, of these approaches are shown with 95% confidence intervals in Fig. 4d and Fig. 5d. According to these results, $\hat{\mathbb{S}}$ in `SiMPLE-PR` or `SiMPLE-OP` is roughly two to three times higher than that in FBS or SBS. But this increase in splitting overhead comes with the benefits of survivability guarantee and reduced backup overhead. Moreover, with the built-in path splitting capability, modern switches are expected to mitigate this impact.



(a) Profit, Ψ vs. α (b) Acceptance Ratio vs. α (c) Backup B/W, $\hat{\mathbb{B}}$ vs. α (d) Avg. Split Overhead vs. α

Fig. 4. Performance Analysis for Fat tree topology.



(a) Profit, Ψ vs. α (b) Acceptance Ratio vs. α (c) Backup B/W, $\hat{\mathbb{B}}$ vs. α (d) Avg. Split Overhead vs. α

Fig. 5. Performance Analysis for Synthetic topology.

TABLE III
AVERAGE EXECUTION TIME (SEC)

Topology	SiMPLE-OP	SiMPLE-PR
Fat tree	61.72	0.95
Synthetic	51.04	0.96

4) Execution Time: The average execution time for embedding a VN request in SiMPLE-OP and SiMPLE-PR are presented in Table III, which shows that SiMPLE-PR is $50-60$ times faster than SiMPLE-OP. A significant portion of the execution time of SiMPLE-OP is consumed by GLPK in finding an optimal solution. However, the performance of SiMPLE-PR (Fig. 4 and Fig. 5) lies very close to SiMPLE-OP in all cases. For large scale instances, GLPK exceeds memory limits and cannot find a solution. Please note that these times are highly dependent on simulation environment, and different results could be obtained on a different machine.

5) Discussion:

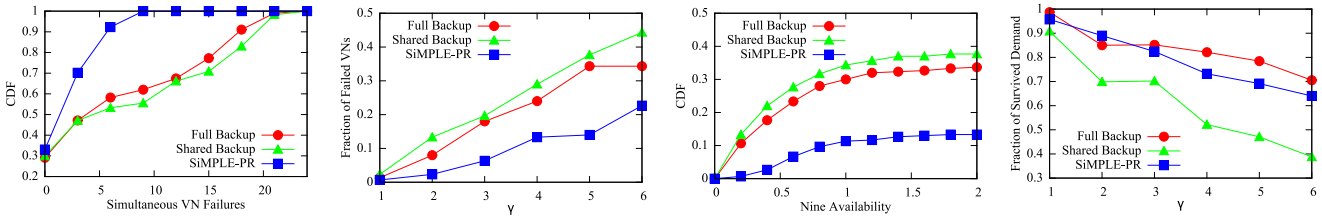
a) Profit vs. $\mathbb{AR}$: From Figures 4a and 5a, we observe that both FBS and SBS suffer from decreasing profit with increasing α. This reduction in profit is due to the lower $\mathbb{AR}$, as presented in Figures 4b and 5b. To embed the VLinks, SiMPLE relies on *path splitting*. Since SiMPLE spreads the VLink demand across multiple paths, it utilizes the substrate resources more efficiently, and achieves a higher $\mathbb{AR}$. On the contrary, FBS and SBS do not rely on path splitting, and fail to achieve satisfactory $\mathbb{AR}$ due to resource fragmentation. SBS utilizes resources more efficiently than FBS because of backup resource sharing, and achieves slightly better performance.

b) Profit vs. overhead: In addition to providing a higher profit as shown in Figures 4a and 5a, SiMPLE requires a lower fraction of backup bandwidth. This behavior is depicted in Figures 4c and 5c. SBS has the lowest backup bandwidth over all workloads α, which is mostly due to the backup resources fair sharing policy. On the contrary, because it is often not cost effective to split smaller demands, SiMPLE has a slightly higher backup bandwidth requirement than SBS at lower α. However, with increasing α, the number of splits at each VLink increases. Therefore, in SiMPLE, $\hat{\mathbb{B}}$ decreases, and becomes similar to SBS. At the same time, path splitting allows SiMPLE to achieve a higher profit than FBS and SBS. However, path splitting brings additional overhead (presented in Fig. 4d and Fig. 5d) to SiMPLE. Nonetheless, this overhead is compensated by larger profit, better acceptance ratio, and lower backup bandwidth requirement.

F. Survivability Evaluation Results for SiMPLE-PR

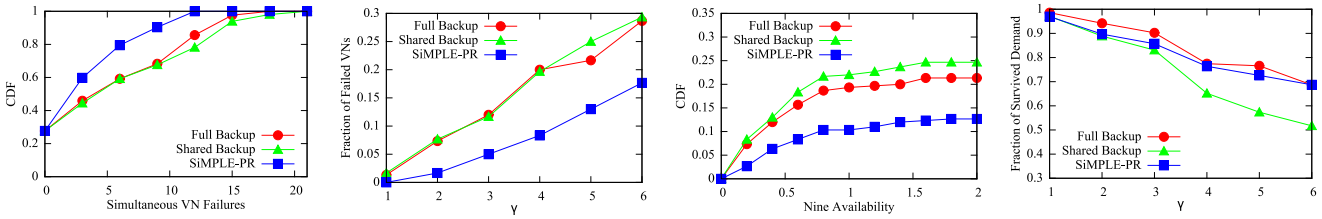
We conducted experiments to evaluate survivability of SiMPLE-PR, FBS, and SBS in the event of failures for Fat tree and Synthetic topologies.

1) Impact of Failures: The impact of failures is evaluated from two perspectives. First, we present the Cumulative Distribution Function (CDF) for $Prob(\rho^i)$ – the probability of i simultaneous VN failures, for $i = 0, 1, 2, \ldots, i_{max}$, where i_{max} denotes the maximum number of simultaneous VN failures. For $\gamma = 5$, the CDF for Fat tree and Synthetic topologies are shown in Fig. 6a and Fig. 7a, respectively. Second, we measure the fraction of failed VNs to total accepted VNs in SN. For different γ, these results are shown in Fig. 6b and Fig. 7b for Fat tree and Synthetic topologies, respectively. These figures show that, both simultaneous and total VN failures are less likely to occur in SiMPLE-PR. In contrast, these quantities are higher in FBS, and the highest in SBS. For larger γ, the number of failed VNs is approximately $50-100\%$ higher in FBS and SBS than that in SiMPLE-PR. These results reveal that SiMPLE-PR provides the best resilience to failures, whereas SBS performs the worst among these schemes.



(a) CDF of $Prob(p^i)$, $\gamma = 5$ (b) Failed VNs vs. γ (c) Nine availability CDF, $\gamma = 5$ (d) Survived B/W, $\hat{\mathbb{F}}$ vs. γ

Fig. 6. Survivability Analysis for Fat tree topology.



(a) CDF of $Prob(p^i)$, $\gamma = 5$ (b) Failed VNs vs. γ (c) Nine availability CDF, $\gamma = 5$ (d) Survived B/W, $\hat{\mathbb{F}}$ vs. γ

Fig. 7. Survivability Analysis for Synthetic topology.

2) Availability: The CDF of nine availability of the failed VNs for $\gamma = 5$ are depicted in Figures 6c and 7c, for Fat tree and Synthetic topologies, respectively. We see that a small fraction of VNs have low nine availability in `SiMPLE-PR`. In contrast, this fraction is much higher in case of FBS and SBS. Therefore, compared to these two schemes, `SiMPLE-PR` provides high availability to a higher number of embedded VNs. For example, the number of VNs with only 68% or less availability (0.5 nines) in FBS and SBS are roughly twice (Synthetic topology) or four times (Fat tree topology) than that in `SiMPLE-PR`.

3) Failure Tolerance: To evaluate the failure tolerance of each of the considered approaches, we measure the average fraction of survived bandwidth for affected VLinks, $\hat{\mathbb{F}}$. Fig. 6(d) and Fig. 7(d) present the changes in $\hat{\mathbb{F}}$ for different values of γ for Fat tree and Synthetic topologies, respectively. In these figures, we see that the $\hat{\mathbb{F}}$ obtained in `SiMPLE-PR` is within $5-10\%$ of that in FBS for all values of γ. However, $\hat{\mathbb{F}}$ provided by SBS is lower than the other two schemes. For larger values of γ, $\hat{\mathbb{F}}$ obtained in SBS is approximately $50-70\%$ less than that in `SiMPLE-PR`, which demonstrates a poor performance of SBS in presence of frequent failures.

4) Discussion:

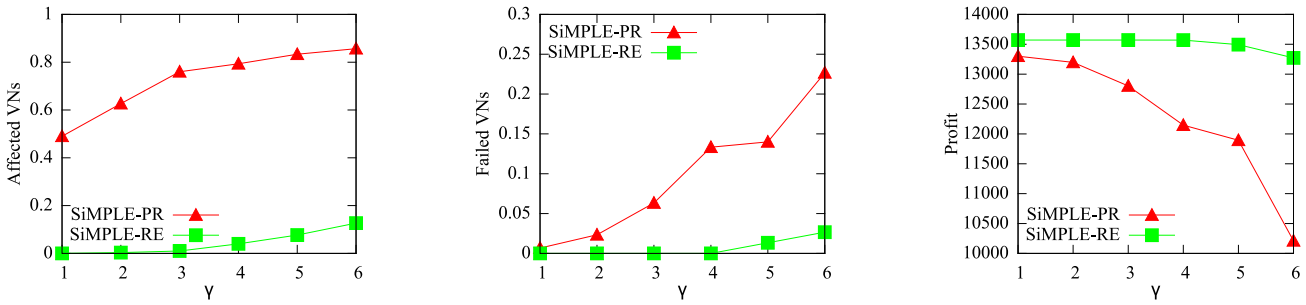
a) Impact of failures vs. availability: We see that SiMPLE outperforms FBS and SBS in both minimizing failure impact (Fig. 6a, Fig. 6b, Fig. 7a, Fig. 7b) and achieving better availability (Fig. 6c, Fig. 7c). The superiority of SiMPLE is achieved due to embedding VLinks over multiple disjoint paths. Since SiMPLE associates more SLinks to each VLink e^v, the minimum number of SLink failures required for e^v to fail also increases. In contrast, the number of associated SLinks to e^v in FBS and SBS are lower, because they do not embed VLinks into multiple paths. Hence, SLink failures are more likely to cause VLink (or, VN) failures in these two approaches. For SBS, the SLinks in the backup path of a VLink e_1^v may already be used by another VLink e_2^v suffering from SLink failure, which makes e_1^v more vulnerable. For these reasons, SBS suffers from failures more than FBS, while SiMPLE outperforms both of these approaches. Again, the number of associated SLinks to each VLink e^v is higher in SiMPLE than that in FBS and SBS. Hence, repairing an SLink is more likely to restore one of the failed paths associated to e^v in SiMPLE. This will make e^v operational by salvaging a fraction of its demand through the restored path. In contrast, the probability of restoring one of the failed paths in FBS or SBS is lower than SiMPLE. Therefore, SiMPLE achieves higher availability than these two strategies.

b) Impact of failures vs. fault tolerance: The correlation between low impact of failures (Fig. 6a, Fig. 6b, Fig. 7a, Fig. 7b) and high fault tolerance (Fig. 6d, Fig. 7d) in SiMPLE is also part of its main concept, i.e., path splitting with minimal backup. In previous paragraph, we have seen how path splitting increases survivability by associating multiple SLinks to each VLink. In addition, we notice that the number of operational SLinks in an affected VLink e^v is also high. These fully operational SLinks facilitate e^v to retain its full demand (for a single SLink failure), or a high fraction of it (for multiple SLink failures). In SiMPLE, $\hat{\mathbb{F}}$ is very close to that of FBS. However, FBS needs dedicated backup path with full demand unlike SiMPLE. The backup path sharing in SBS makes it vulnerable to multiple and frequent failures. Therefore, for the affected VLinks, SiMPLE performs identically to FBS, and outperforms SBS.

G. Survivability Evaluation Results for `SiMPLE-RE`

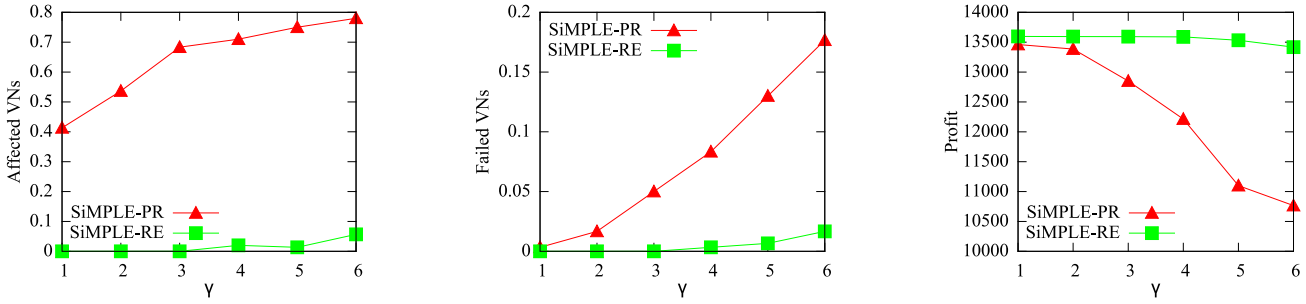
In this section, we present the simulation results on survivability experiments for `SiMPLE-RE`. For Fat tree and Synthetic topologies, these results are shown in Fig. 8 and Fig. 9, respectively.

1) Impact of Failures: The impact of failures are investigated from two perspectives. First, we measure the fraction of affected VNs to total VNs in the network, for



(a) Affected VNs vs. γ (b) Failed VNs vs. γ (c) Increase in Profit vs. γ

Fig. 8. Recovery Analysis for Fat tree topology.



(a) Affected VNs vs. γ (b) Failed VNs vs. γ (c) Increase in Profit vs. γ

Fig. 9. Recovery Analysis for Synthetic topology.

different values of γ. These fractions for SiMPLE-PR and SiMPLE-RE are illustrated in Fig. 8a (for Fat tree topology) and in Fig. 9a (for Synthetic topology). These graphs demonstrate that, when SiMPLE-RE is adopted, the total number of affected VNs are always decreased by a significant amount. For smaller values of γ, SiMPLE-RE successfully recovers all affected VNs in the substrate. The decrease in the affected VNs is less when γ is large. However, SiMPLE-RE still decreases the number of affected VNs by a factor of 85% (for Fat tree topology) or 90% (for Synthetic topology).

Second, we measure the fraction of failed VNs to total VNs in the substrate. For Fat tree and synthetic topologies, these numbers are shown in Fig. 8b and in Fig. 9b, respectively. These figures state that, for smaller values of γ, the number of failed VNs are decreased completely, for both Fat tree and Synthetic topologies. For larger values of γ, this decreasing factor is still close to 90%.

2) Profit: The profit for both SiMPLE-PR and SiMPLE-RE are shown for Fat tree and Synthetic topologies, in Fig. 8c and in Fig. 9c, respectively. From these two figures, we see that SiMPLE-RE provides a higher profit than SiMPLE-PR, for all values of γ. Especially, for higher values of γ, this increase in profit can reach as much as 35%.

3) Discussion: In comparison with SiMPLE-PR, SiMPLE-RE reduces both the number of affected and failed VNs, as well as increases the profit. For small values of γ, the impact of failures is relatively low, and SiMPLE-RE recovers each failure successfully. For this reason, it decreases the number of affected VNs and failed VNs by 100%. For higher values of γ, the impact of failures are also higher, and, as discussed in Section III-A2, there can be cases when SiMPLE-RE fails to recover. For this reason, this affected or failed VN decreasing factor is not 100% in these cases.

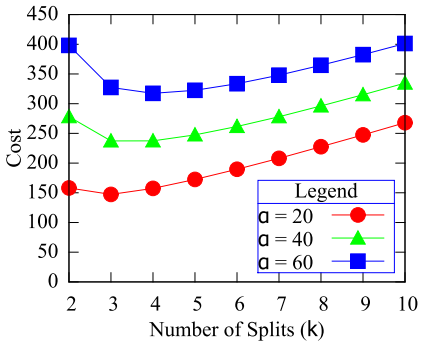


Fig. 10. Impact of different number of splits for one VLink.

Since SiMPLE-RE recovers the affected and failed VNs reactively, it also increases the profit Ψ. This behavior is more prominent for large values of γ. For small γ, the number of affected or failed VNs are also small in SiMPLE-PR, and their recoveries do not increase Ψ more than 15%. However, for large γ, the number of affected or failed VNs are also high. In these cases, even though SiMPLE-RE cannot recover all the affected or failed VNs, it recovers most of them. As a result, the increase in Ψ is also larger.

H. Determining the Maximum Number of Splits

To determine the upper bound on the number of splits for a VLink, we ran an experiment on a 12-ary

Fat tree [24] topology. For this experiment, we embedded only one VN with only one VLink. The number of paths (i.e., splits, k) allocated per VLink was varied from 2 to 10, and the embedding cost (given in (6)) for each set of paths was measured. The experiment was run for different workloads $\alpha = 20$, 40, and 60. The results from this experiment are shown in Fig. 10.

Fig. 10 shows that, for all α, the embedding cost increases with an increasing number of splits. When the number of paths increases beyond 5, this behavior becomes more prominent. For this reason, we restricted the value of k between 2 and 5 for the experiments in Section V.

VI. Related Works

A. VN Embedding

The VN Embedding literature can be classified based on the coordination between its two sub-problems – node embedding and link embedding [22]. The first category, *i.e.,* the uncoordinated VNE, does not take into account the possibility of coordination between these two sub-problems. One possible example of this category is presented in [18], where the link embedding is performed after a greedy node embedding. In the link embedding step, the multi-commodity flow problem formulation is used to find a set of paths between the substrate hosts for each virtual link. However, this lack of coordination might result into embedding adjacent virtual nodes towards distant physical nodes, which increases resource consumption. The second category, *i.e.,* the one-stage coordinated VNE [28], embeds the virtual links at the same time as the virtual nodes. In this category, at first, a greedy algorithm (*e.g.,* PageRank [29]) is used to rank the virtual nodes. Then, a virtual node pair and their intermediate virtual link are embedded. Afterwards, the remaining virtual nodes and virtual links connecting them are embedded one by one. The third category, *i.e.,* the two-stage coordinated VNE, embeds the virtual network into the substrate network into two distinct steps [30]. The first step is to represent a set of preferred substrate nodes for each virtual node as a meta-node, and perform a relaxed Mixed Integer Programming (MIP) to calculate the physical hosts of the virtual nodes in the corresponding meta-nodes. The second step is to find the physical paths corresponding to the virtual links, and this is solved by the multi-commodity flow problem formulation. In the one-stage and two-stage coordinated VNE approaches, the coordination between node mapping and link mapping stages are strong, and the substrate nodes chosen to map the virtual nodes are likely suited to provide virtual link mappings with low embedding cost.

B. Survivability

Survivable Virtual Network Embedding (SVNE) deals with keeping the VNs intact, even after substrate failures (*e.g.,* SNodes or SLinks). SNode failures are very rare and results into multiple SLink failures [14], [15], [19]. Hence, majority of the SVNE literature focuses on SLink failures.

In SVNE literature, protection (also known as, *proactive allocation*) refers to allocating redundant resources per each VLink while embedding, *i.e.,* before any SLink failure occurs. A number of research works contribute to the VLink protection problem. Several research works, including [8] and [31], formulated two separate LP models for VLink embedding, which allocate full demand of each VLink along a primary path and a disjoint backup path. These full backup schemes result into poor bandwidth utilization. Shared backup schemes, on the other hand, allow multiple VLinks to share backup resources allocated to each end-to-end path [26] or SLink [16]. However, primary paths are dedicated to each VN and cannot be shared with other VNs. Since the same backup resources are shared among multiple VLinks, these approaches do not offer bandwidth guarantee, even in presence of a single SLink failure. The authors in [16] propose two shared backup schemes to protect against any potential single SLink failure: Shared On-Demand approach and Shared Pre-Allocation approach. In the first approach, bandwidth resources are allocated to the primary flows and backup flows upon the arrival of each VN request. Backup resources can be reused by other VNs to make room for accepting more incoming VN requests. However, primary flows are dedicated to each VLink and are not allowed to be shared with other VLinks. In the second approach, backup bandwidth for each SLink is pre-allocated during the initial phase, *i.e.,* before any VN request arrives. Since the bandwidth pre-allocation only needs to be done once and not for every VN request, it requires less processing during the VN embedding phase. The disadvantage of these approaches is that they can not guarantee recovery of full bandwidth even in the case of single SLink failure. For example, if multiple VLinks are embedded on the failed SLink, bandwidth recovery of these VLinks can be compromised due to sharing.

The recovery (or, reactive) techniques provide VN survivability without allocating any backup bandwidth at the beginning. In practice, they react after an SLink fails, and start the path restoration mechanism. Lu *et al.* [32] proposes such a reactive mechanism where either a substitute path is searched, or the corresponding VN is remapped after a link failure. In a highly saturated SN, there may not be enough resources left for finding the substitute path or remapping the VN. We discuss two most prominent recovery approaches in SVNE. Rahman *et al.* [8], [33] proposes a three-phase hybrid mechanism where a set of possible backup detours for each SLink is computed before any VN request arrives. Then, node embedding is done for the arriving request with an existing embedding algorithm [5], [7], followed by a multi-commodity based link embedding. Finally, in the event of a SLink failure, a reactive online optimization mechanism reroutes the affected flows along candidate backup detours selected in the first phase. This approach may demand a long convergence time, leaving VNs unavailable during such periods. This may cause data loss. Furthermore, since the substrate resources are not fragmented to serve multiple virtual requests, all proactive and reactive approaches may suffer from improper load balancing and link underutilization.

Network survivability in Optical and Multi-Protocol Label Switched (MPLS) networks is usually considered during the network design [34]. The solutions in these domains, *e.g.,* [11] and [12] assume that traffic demands are known in advance (*i.e.,* offline). In contrast, SVNE is online; it needs

to provide survivability for unpredictable VN request arrivals and demand patterns. Furthermore, SVNE solutions have to ensure the intactness of all VLinks in presence of failures. This restriction is not present in Optical/MPLS networks, where the goal is to ensure connectivity in the network. Hence, these solutions are not suitable for the NV environment.

Due to the importance of providing high service availability in Cloud environments, recently there is a trend towards designing survivable resource allocation schemes for bandwidth constrained data centers [35]–[38]. Xu *et al.* [39] proposes a resource allocation scheme for provisioning virtual data centers with backup virtual machines and links. However, this work do not consider the survivability of physical machines and links. Bodík *et al.* [10] proposes an optimization framework for improving survivability while reducing the total bandwidth consumption. However, this approach does not consider the heterogeneous failure rates of the underlying physical equipment, and mitigates the bandwidth bottleneck only in the core of the data center. Zhang *et al.* [38] proposes a framework for reliable virtual data center embedding in clouds by considering heterogeneous failure rates. SiMPLE differs from these works in its objective of simultaneously optimizing VN survivability, bandwidth usage, and path splitting overhead.

Path splitting, such as Equal-cost Multipath Routing (ECMP) [40] or Multipath TCP (MPTCP) [41], is a routing strategy to allow one data stream to be split across multiple paths. Path splitting enables an increased resource utilization, failure-tolerance, and better QoS. Multi-path routing, such as flowlet-based traffic splitting [42], may have an impact on applications like large data transfers and multimedia streaming. Examples of such impacts include delivery of out of order packets within a flow, and handle the variable amount of delay observed by the packets in different paths. A number of works in both VNE and SVNE literatures use path splitting. The authors in [18] introduced path splitting in VNE to embed a VLink over multiple substrate paths. For this purpose, they use multi commodity flow (MCF) based link embedding. In an MCF based solution, any intermediate SNode between the end hosts can split the flow. However, in SiMPLE, we assume that only the source and destination SNodes can split the flow, hence MCF cannot be used in this context. Oliviera *et al.* [17], [43], [44] proposed embedding strategies that provide opportunistic recovery for each VLink via path splitting. However, these approaches do not guarantee VN survivability in presence of a single SLink failure.

VII. Conclusion

In this paper, we have presented SiMPLE which exploits the substrate network's path splitting capability for survivable embedding of virtual network requests. SiMPLE's design goal is to reconcile the conflicting objectives of achieving maximal survivability and minimizing both redundancy and overhead. Compared to existing approaches, SiMPLE reserves less backup bandwidth, yet guarantees virtual link survivability in presence of a single substrate link failure. In case of multiple link failures, the survived bandwidth of the affected virtual link(s) is better than that of FBS and SBS. Simulation results demonstrated that `SiMPLE-PR` reduces the failure percentage by at least 50% over those two schemes, and provides better availability of VNs. In addition, backup bandwidth overhead in `SiMPLE-PR` is $40-50\%$ less than that of FBS, and lies very close to SBS, as well as to `SiMPLE-OP`. The recovery approach, `SiMPLE-RE`, improves profit generated by the `SiMPLE-PR` by approximately 40%, as well as decreases the number of failed VNs by approximately 90%. Finally, the path splitting overhead incurred by SiMPLE is compensated by guaranteed survivability, increased profit, better acceptance ratio and lower backup bandwidth requirement.

As a future extension of this work, we intend to evaluate the performance of SiMPLE through a prototype implementation in an SDN environment [45] for supporting path splitting in the substrate, to be deployed on an SDN testbed, like Distributed OpenFlow Testbed (DOT) [46], [47], or Mininet [48]. We also would like to extend SiMPLE's link embedding concept towards a coordinated node and link mapping strategy. Finally, it would be interesting to extend this work to multi-layer NV environment [49] that could raise further challenges because of the need for cross layer optimization.

References

[1] N. M. M. K. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," *IEEE Commun. Mag.*, vol. 47, no. 7, pp. 20–26, Jul. 2009.

[2] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, no. 5, pp. 862–876, 2010.

[3] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quart. 2013.

[4] N. F. Butt, M. Chowdhury, and R. Boutaba, "Topology-awareness and reoptimization mechanism for virtual network embedding," in *Proc. 9th Int. IFIP Netw. Conf. (NETWORKING)*, Chennai, India, May 2010, pp. 27–39.

[5] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, 2009, pp. 783–791.

[6] J. Lu and J. Turner, "Efficient mapping of virtual networks onto a shared substrate," Dept. Comput. Sci. Eng., Washington Univ., St. Louis, MO, USA, Tech. Rep. WUCSE-2006-35, 2006.

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

[8] M. R. Rahman and R. Boutaba, "SVNE: Survivable virtual network embedding algorithms for network virtualization," *IEEE Trans. Netw. Service Manag.*, vol. 10, no. 2, pp. 105–118, Jun. 2013.

[9] *IT Downtime Costs $26.5 Billion In Lost Revenue*. Accessed on May 24, 2011. [Online]. Available: [Online]. Available: http://www.informationweek.com/it-downtime-costs-$265-billion-in-lost-revenue/d/d-id/1097919

[10] P. Bodík *et al.*, "Surviving failures in bandwidth-constrained datacenters," in *Proc. ACM SIGCOMM Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, Helsinki, Finland, 2012, pp. 431–442.

[11] W. Lau and S. Jha, "Failure-oriented path restoration algorithm for survivable networks," *IEEE Trans. Netw. Service Manag.*, vol. 1, no. 1, pp. 11–20, Apr. 2004.

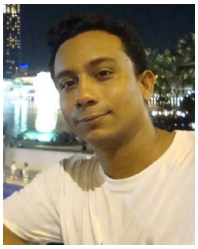
[12] K. Lee, E. Modiano, and H.-W. Lee, "Cross-layer survivability in WDM-based networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 4, pp. 1000–1013, Aug. 2011.

[13] Y. Xiong and L. G. Mason, "Restoration strategies and spare capacity requirements in self-healing ATM networks," *IEEE/ACM Trans. Netw.*, vol. 7, no. 1, pp. 98–110, Feb. 1999.

[14] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 350–361, Aug. 2011.

[15] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of failures in an IP backbone," in *Proc. IEEE INFOCOM*, vol. 4. Hong Kong, Mar. 2004, pp. 2307–2317.
[16] T. Guo, N. Wang, K. Moessner, and R. Tafazolli, "Shared backup network provision for virtual network embedding," in *Proc. IEEE ICC*, Kyoto, Japan, Jun. 2011, pp. 1–5.
[17] R. R. Oliveira *et al.*, "DoS-resilient virtual networks through multipath embedding and opportunistic recovery," in *Proc. SAC*, Coimbra, Portugal, Mar. 2013, pp. 597–602.
[18] 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, Apr. 2008.
[19] A. Todimala and B. Ramamurthy, "A scalable approach for survivable virtual topology routing in optical WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 6, pp. 63–69, Aug. 2007.
[20] M. M. A. Khan, N. Shahriar, R. Ahmed, and R. Boutaba, "SiMPLE: Survivability in multi-path link embedding," in *Proc. IEEE 11th Int. Conf. Netw. Service Manag.*, Barcelona, Spain, 2015, pp. 210–218.
[21] S. G. Kolliopoulos and C. Stein, "Improved approximation algorithms for unsplittable flow problems," in *Proc. IEEE SFCS*, Miami Beach, FL, USA, Oct. 1997, pp. 426–436.
[22] S. Herker, A. Khan, and X. An, "Survey on survivable virtual network embedding problem and solutions," in *Proc. ICNS*, Lisbon, Portugal, 2013, pp. 99–104.
[23] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, "Worst-case performance bounds for simple one-dimensional packing algorithms," *SIAM J. Comput*, vol. 3, no. 4, pp. 299–325, 1974.
[24] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Oct. 2008.
[25] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Manag. Sci.*, vol. 17, no. 11, pp. 712–716, Jul. 1971.
[26] Y. Chen, J. Li, T. Wo, C. Hu, and W. Liu, "Resilient virtual network service provision in network virtualization environments," in *Proc. IEEE ICPADS*, Shanghai, China, Dec 2010, pp. 51–58.
[27] J. R. Douceur, "Is remote host availability governed by a universal law?" *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 3, pp. 25–29, 2003.
[28] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, Apr. 2011.
[29] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the Web," Stanford InfoLab, Tech. Rep. 1999-66, Nov. 1999.
[30] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. INFOCOM*, Rio de Janeiro, Brazil, 2009, pp. 783–791.
[31] Q. Chen, Y. Wan, X. Qiu, W. Li, and A. Xiao, "A survivable virtual network embedding scheme based on load balancing and reconfiguration," in *Proc. IEEE NOMS*, Kraków, Poland, May 2014, pp. 1–7.
[32] B. Lu, T. Huang, X. C. Sun, J.-Y. Chen, and Y.-J. Liu, "Dynamic recovery for survivable virtual network embedding," *J. China Univ. Posts Telecommun.*, vol. 21, no. 3, pp. 77–84, Jun. 2014.
[33] M. R. Rahman, I. Aib, and R. Boutaba, "Survivable virtual network embedding," in *Proc. 9th Int. IFIP Netw. Conf. (NETWORKING)*, Chennai, India, May 2010, pp. 40–52.
[34] D. Medhi, "A unified approach to network survivability for teletraffic networks: Models, algorithms and analysis," *IEEE Trans. Commun.*, vol. 42, no. 234, pp. 534–548, Feb./Mar./Apr. 1994.
[35] S. Agarwal *et al.*, "Volley: Automated data placement for GEO-distributed cloud services," in *Proc. NSDI*, San Jose, CA, USA, 2010, p. 2.
[36] N. Bansal *et al.*, "Towards optimal resource allocation in partial-fault tolerant applications," in *Proc. IEEE 27th Conf. Comput. Commun. INFOCOM*, Phoenix, AZ, USA, 2008, pp. 1319–1327.
[37] W.-L. Yeow, C. Westphal, and U. C. Kozat, "Designing and embedding reliable virtual infrastructures," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 57–64, 2011.
[38] Q. Zhang, M. F. Zhani, M. Jabri, and R. Boutaba, "Venice: Reliable virtual data center embedding in clouds," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 289–297.
[39] J. Xu, J. Tang, K. Kwiat, W. Zhang, and G. Xue, "Survivable virtual infrastructure mapping in virtualized data centers," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Honolulu, HI, USA, 2012, pp. 196–203.
[40] C. E. Hopps. (Nov. 2000). *Analysis of an Equal-Cost Multi-Path Algorithm*. [Online]. Available: https://tools.ietf.org/html/rfc2992
[41] (Nov. 2011). *Multipath TCP—Linux Kernel Implementation*. [Online]. Available: http://multipath-tcp.org
[42] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 51–62, 2007.
[43] R. R. Oliveira *et al.*, "No more backups: Toward efficient embedding of survivable virtual networks," in *Proc. IEEE ICC*, Budapest, Hungary, Jun. 2013, pp. 2128–2132.
[44] R. R. Oliveira *et al.*, "Opportunistic resilience embedding (ORE): Toward cost-efficient resilient virtual networks," *Comput. Netw.*, vol. 89, pp. 59–77, Oct. 2015.
[45] D. Kreutz *et al.* (Oct. 2014). *Software-Defined Networking: A Comprehensive Survey*. [Online]. Available: http://arxiv.org/abs/1406.0440
[46] (Oct. 2014). *Dot: A Distributed Openflow Testbed*. [Online]. Available: http://dothub.org
[47] A. R. Roy, M. F. Bari, M. F. Zhani, R. Ahmed, and R. Boutaba, "Design and management of DOT: A distributed openflow testbed," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, Kraków, Poland, May 2014, pp. 1–9.
[48] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proc. 8th Int. Conf. Emerg. Netw. Experiments Technol.*, Nice, France, 2012, pp. 253–264.
[49] T. Koponen *et al.*, "Network virtualization in multi-tenant datacenters," in *Proc. USENIX NSDI*, Berkeley, CA, USA, 2014, pp. 203–216.

Md Mashrur Alam Khan received the B.Sc. degree in computer science and engineering from the Bangladesh University of Engineering and Technology, in 2012, and the M.Math. degree in computer science from the University of Waterloo, Waterloo, ON, Canada, in 2015. He is currently a Software Engineer with Cisco. His research interests include network virtualization, virtual network embedding, fault tolerance, and Internet of Things.



Nashid Shahriar received the B.Sc. and M.Sc. degrees in computer science and engineering from the Bangladesh University of Engineering and Technology, in 2009 and 2011, respectively. He is currently pursuing the Ph.D. degree with the University of Waterloo, Canada. His research interests include network virtualization, future Internet architectures, and network function management. He was a recipient of the David R. Cheriton Graduate Scholarship at the University of Waterloo.

Reaz Ahmed received the B.Sc. and M.Sc. degrees from the Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh, in 2000 and 2002, respectively, and the Ph.D. degree from the University of Waterloo, in 2007, all in computer science. He is an Associate Professor with the Department of Computer Science and Engineering, BUET. His research interests include future Internet architectures, wide area service discovery, and content sharing peer-to-peer networks with a focus on search flexibility, efficiency, and robustness. He was a recipient of the IEEE Fred W. Ellersick Award in 2008.

Raouf Boutaba received the M.Sc. and Ph.D. degrees in computer science from the University Pierre and Marie Curie, Paris, in 1990 and 1994, respectively. He is currently a Professor of Computer Science with the University of Waterloo, Canada. His research interests include resource and service management in networks and distributed systems. He was a recipient of several best paper awards and other recognitions, such as the Premiers Research Excellence Award, the IEEE ComSoc Hal Sobol Award, the Fred W. Ellersick Award, the Joe LociCero Award, the Dan Stokesbury Award, the Salah Aidarous Award, and the IEEE Canada McNaughton Gold Medal. He is the Founding Editor-in-Chief of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT from 2007 to 2010, and on the editorial boards of other journals. He is a fellow of the Engineering Institute of Canada, and the Canadian Academy of Engineering.