SiMPLE: Survivability in Multi-Path Link Embedding

Md Mashrur Alam Khan, Nashid Shahriar, Reaz Ahmed, Raouf Boutaba
David R Cheriton School of Computer Science, University of Waterloo, ON, Canada
{mmalamkh | nshahria | r5ahmed | rboutaba}@uwaterloo.ca

***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 can enable 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 Virtual Network Embedding (SVNE). A key challenge in the SVNE problem is to ensure VN survivability with minimal resource redundancy. To address this challenge, we propose SiMPLE. By exploiting path diversity in the physical network, SiMPLE provides guaranteed VN survivability against single link failure. In addition, SiMPLE produces highly survivable VN embeddings in presence of multiple link failures while incurring very low resource redundancy. We provide an ILP formulation for this problem and implement it using GLPK. We also propose a greedy algorithm to solve larger instances of the problem. Simulation results show that our solution 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 requires 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) [9], [10] 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 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 [13]. 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 the literature [6], [11], [22], [35]. 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 [27]. 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 [1]. Hence, VN survivability is crucial for both InPs and SPs.

Survivability has been thoroughly investigated in non-virtualized networks in the past [5], [19], [20], [29]. However, these solutions focus on ensuring only network connectivity during failures, whereas in SVNE the focus is to preserve both connectivity and bandwidth by using mutually exclusive substrate resources. 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], [23]. Reserving the full demand of a virtual link as backup is expensive, since backup resources remain idle when there are no failures [27]. To minimize resource wastage, shared backup schemes have been proposed in [15]. However, they do not guarantee the 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 [24], [33]. 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 [23] have shown that link failures are more frequent than node failures, and node failures can be modeled as multiple link failures [28]. 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.
- **Algorithms.** 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.

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 a greedy algorithm for link embedding in SVNE. Section V presents our evaluation results, and Section VI discusses the 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.

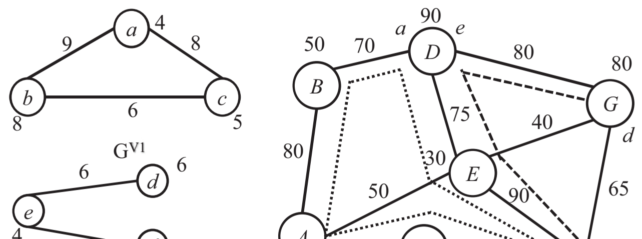


Fig. 1: Embedding VN requests onto a substrate

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: $\forall n^v \in N^V : c(n^v) \leq r(\xi_N(n^v))$.

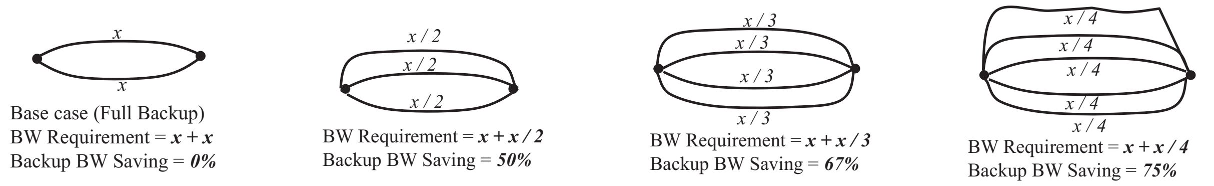
2) Link Embedding: Each VLink $e^v \in E^V$ is mapped to a substrate path $p^{e^v} \in P^{e^v}$ between ingress SNode, $\xi_N(e^v_s)$ and egress SNode, $\xi_N(e^v_d)$, where e^v_s and e^v_d denote the source and destination VNodes of e^v, respectively. The link mapping function is $\xi_E : E^V \rightarrow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

Solving the VNE problem is $\mathcal{NP}$-hard, as it can be reduced to the multi-way separator problem [22]. Even with a given VNode mapping, the problem of optimally allocating the VLinks to substrate paths reduces to the unsplittable flow problem [18], and is thus $\mathcal{NP}$-hard as well. Fig. 1 depicts the embedding 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 $\xi^2_E(ed) = DG$, $\xi^2_E(ef) = DEF$.

B. Survivable Virtual Network Embedding

An SLink may not operate properly all the time due to various reasons such as fiber cut, maintenance, mis-configuration, and so on [14], [23]. 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 [16], 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.



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

Fig. 2: The SiMPLE Embedding Concept

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 [33] 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 [24]. 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 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 considered.

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 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, (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 ν VLinks, and each VLink demands x bandwidth. At time T, substrate bandwidth consumptions of FBS and SiMPLE are given by $\mathcal{B}_F^T = 2n\bar{l}\nu x$ and $\mathcal{B}_S^T = kn\bar{l}\nu\frac{x}{k-1}$, 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 $\mathcal{B}_F^T - \mathcal{B}_S^T = n\bar{l}\nu x\frac{k-2}{k-1}$. 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 $\hat{\mathcal{B}}_S^T = k\hat{n}\bar{l}\nu\frac{x}{k-1}$. 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. ■

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.

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

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

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).

$$\ddot{I}(e^v, P^{e^v}, k^{e^v}) = \Big(D(\xi_N(e_s^v), k^{e^v}) + D(\xi_N(e_d^v), k^{e^v}) \Big) \tag{3}$$

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 $\beta(n^s)$.

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

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.

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

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. Alternatively, it can be interesting to distribute bandwidth demand proportional to the residual SLink capacity. However, this may result into higher backup bandwidth requirement, since the maximum bandwidth in the alternative paths should be allocated to the backup path.

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` :

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

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.

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

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

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

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

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

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

IV. Proposed Greedy Solution

The ILP model presented in Section III can find optimal solutions for small instances of the multi-path embedding problem, but it will not scale with SN and VN size. In

[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 Greedy Algorithm, `SiMPLE-GR`

```
function SiMPLE-GR($G^S, G^V, \xi_N$)
    for all $e^v \in E^V$ do
        $\forall k \in \{2,3,4,5\} : P^k \leftarrow \phi \wedge Cost(P^k) \leftarrow \infty$
        for $k \in \{2,3,4,5\}$ do
            $\mathbb{E}^S \leftarrow E^S$
            for $j \leftarrow 1, k$ do
                $Q \leftarrow$ Dijkstra $(N^S, \mathbb{E}^S, \xi_N(e^v_s), \xi_N(e^v_d), \frac{b(e^v)}{k-1})$
                $P^k \leftarrow P^k \cup Q$
                $\mathbb{E}^S \leftarrow \mathbb{E}^S - P^k$
            end for
        end for
        $P^* \leftarrow \min(P^2, P^3, P^4, P^5)$
        if $Cost(P^*) = \infty$ then
            return $\phi$
        end if
        $\xi_E(e^v) \leftarrow P^*$
    end for
    $\forall e^s \in E^S \cap P^*$ : update $r(e^s)$
    return $\xi_E$
end function
```

this section, we propose a scalable greedy algorithm named `SiMPLE-GR` to solve this problem. `SiMPLE-GR` assumes that the node mapping has already been done, possibly using one of the greedy approaches (e.g., First Fit [17]). `SiMPLE-GR` 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-GR`, as presented in Algorithm 1, is an SN G^S, a VN G^V, and its node mapping function, ξ_N. In `SiMPLE-GR`, we split each VLink into no more than five paths. This is because, we experimentally found that 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. For space constraints, we skip the details of this experiment. `SiMPLE-GR` iteratively works on each VLink of a newly arrived G^V. The set P^k (initially empty) denotes the set of candidate paths selected for split k, where $2 \leq k \leq 5$ and $k \in N$. At each iteration of k, `SiMPLE-GR` runs the Dijkstra's weighted shortest path algorithm to select a candidate path with the sufficient residuals $(b(e^v) \mathbin{/} (k-1))$ between the source and destination SNodes of the corresponding VLink. This path is added to P^k. To maintain the disjointness constraint, the SLinks of the path are temporarily removed from G^S. After the end of this loop, the discarded SLinks are restored, and the set of paths with the minimal cost, P^*, is calculated. If no such set is found (i.e., cost of P^* is ∞), `SiMPLE-GR` finds no feasible mapping for this VLink (and hence G^V) and returns ϕ. Otherwise, it updates the link mapping function ξ_E, and moves on to process the next VLink. If the mapping of all the VLinks are found in this process, `SiMPLE-GR` returns ξ_E. In this case, G^V is embedded onto G^S, and the residual capacities in the corresponding SLinks are updated.

TABLE I: Evaluation Environment

Characteristics	Small Scale	Large Scale
Fat-tree Arity	10	20
Number of SNodes	125	500
SNode Capacity	$[50, 150]$	$[10, 50]$
SNode Switching Cost	$[2, 7]$	$[2, 7]$
Number of SLinks	500	4000
SLink Capacity	$[70, 80]$	$[70, 80]$
SLink Delay	$[3, 15]$	$[3, 15]$
Split Cost	10 per split	10 per split
Join 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	α% 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$\{7000\}$

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 [3] to assess the behavior of SiMPLE in data center networks. To demonstrate `SiMPLE-GR` 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-GR` and the optimal solution, `SiMPLE-OP`. 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 [31]. However, we evaluate only `SiMPLE-GR` in large scale experiments.

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 I are similar to the previous works [8], [27]. In Table I, $[x_{min}, x_{max}]$ 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. However, the basic constraints in VNE ((7), (9), (10)) were satisfied by this random node mapping algorithm.

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 I) to magnify the impact of failures.

B. Baseline Algorithms

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

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 [15] 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 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, $\hat{\mathbb{B}} = \underset{\forall e^v \in E^V}{Avg} \left(\mathbb{B}^{e^v}\right)$.

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

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, $\hat{\mathbb{F}} = \underset{\forall \tilde{e}^v \in \tilde{E}^V}{Avg} \left(\mathbb{F}^{\tilde{e}^v}\right)$.

6) Probability of Simultaneous VN Failures, $Prob(\rho^i)$: 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 [12]. We compute the nine availability of a failed VN, G^V, as $\big(-\log_{10} \omega(G^V)\big)$, where $\omega(G^V)$ is the ratio of time G^V is in failed state to its lifetime.

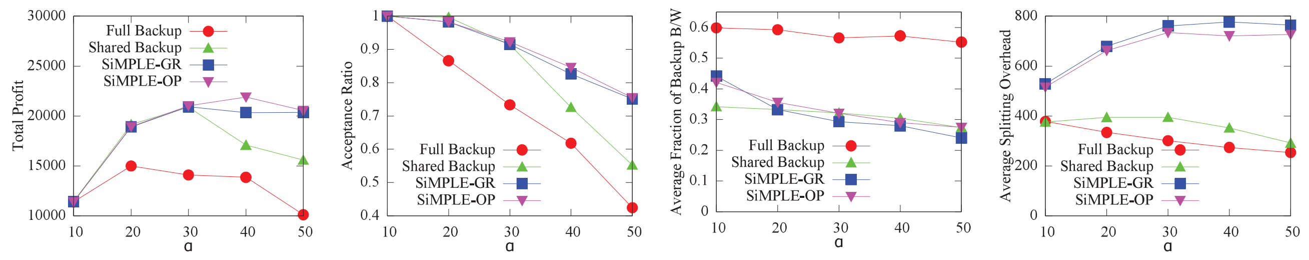
E. Performance Evaluation Results

We evaluate the VN embedding performances in all four schemes as follows.

1) Profit: In terms of Profit, SiMPLE-GR outperforms both FBS and SBS approaches, and is very close to the optimal result (SiMPLE-OP). Fig. 3a shows the profits for different load (or, α). As shown in this figure, all approaches achieve similar profits for small load ($\alpha \leq 10$). However, at increased loads, the profits decrease for FBS and SBS, and SiMPLE-GR achieves approximately 100% and 50% more profit than FBS and SBS, respectively.

2) Acceptance Ratio: Results for the $\mathbb{AR}$ at different α are given in Fig. 3b. According to these results, SiMPLE-GR performs as good as FBS and SBS for small loads ($\alpha \leq 10$). However, at large loads, $\mathbb{AR}$ of SiMPLE-GR exceeds these two approaches by $20-50\%$, and lies close to SiMPLE-OP.

3) Overhead: The overhead of the considered approaches are evaluated from two perspectives – backup bandwidth allocation and splitting overhead. SiMPLE-GR uses a very small fraction of the total allocated bandwidth resource as backup. For different α, the fraction $\hat{\mathbb{B}}$ is shown in Fig. 3c. This figure shows that FBS uses more than half of its resources for backup, regardless of α. On the contrary, $\hat{\mathbb{B}}$ is relatively smaller for both SiMPLE-GR 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. 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 in Fig. 3d. According to these results, $\hat{\mathbb{S}}$ in SiMPLE-GR 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. 3: Performance Analysis

4) Execution Time: The average execution time for embedding a VN request in `SiMPLE-OP` and `SiMPLE-GR` is 61.72 and 0.95 seconds, respectively. This shows that `SiMPLE-GR` 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-GR` (as represented in Fig. 3) lies very close to `SiMPLE-OP` in all cases. For large scale instances, GLPK exceeds memory limits and cannot find a solution.

5) Discussion:

a) Profit vs. $\mathbb{AR}$: From Fig. 3a, 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 Fig. 3b. 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 Fig. 3a, SiMPLE requires a lower fraction of backup bandwidth. This behavior is depicted in Fig. 3c. 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 small 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 extra overhead (see Fig. 3d) to SiMPLE. Nonetheless, this overhead is compensated by larger profit, better acceptance ratio, and lower backup bandwidth requirement.

F. Survivability Evaluation Results

We conducted experiments to evaluate survivability of `SiMPLE-GR`, FBS, and SBS in the event of failures.

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. The CDF for $\gamma = 5$ is shown in Fig. 4a. Second, we measure the fraction of failed VNs to total accepted VNs in SN. For different γ, these results are shown in Fig. 4b. These figures show that both simultaneous and total VN failures are less likely to occur in `SiMPLE-GR`. In contrast, these quantities are higher in FBS, and highest in SBS. For larger γ, the number of failed VNs is approximately $50-100\%$ higher in FBS and SBS than that in `SiMPLE-GR`. These results reveal that `SiMPLE-GR` provides the best resilience to failures, whereas SBS performs worse than others.

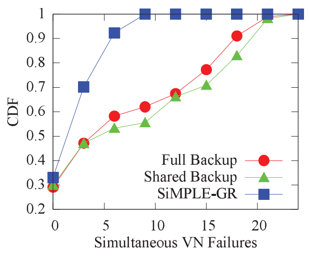
2) Availability: The CDF of nine availability of the failed VNs for $\gamma = 5$ are depicted in Fig. 4c. We see that a small fraction of VNs have low nine availability in `SiMPLE-GR`. In contrast, this fraction is much higher in case of FBS and SBS. Therefore, compared to these two schemes, `SiMPLE-GR` provides high availability to a higher number of VNs. For example, the number of VNs with 68% or less availability (0.5 nines) in FBS and SBS are roughly four times than that in `SiMPLE-GR`.

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. 4d presents the changes in $\hat{\mathbb{F}}$ for different values of γ. In these figures, we see that the $\hat{\mathbb{F}}$ obtained in `SiMPLE-GR` 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-GR`, which demonstrates a poor performance of SBS in presence of frequent failures.

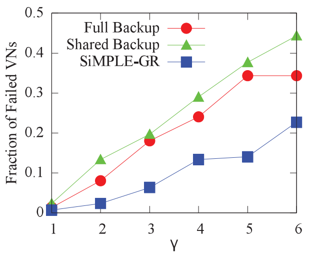
4) Discussion:

a) Impact of Failures vs. Availability: We see that SiMPLE outperforms FBS and SBS in both minimizing failure impact (Fig. 4a, Fig. 4b) and achieving better availability (Fig. 4c). 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.

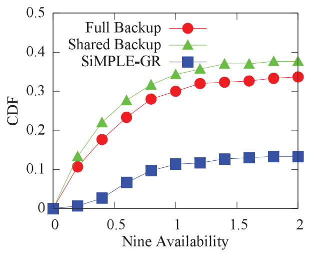
b) Impact of Failures vs. Fault Tolerance: The correlation between low impact of failures (Fig. 4a, Fig. 4b) and high



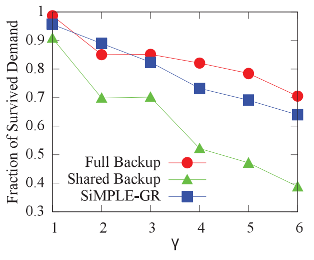
(a) CDF of $Prob(p^i)$, $\gamma = 5$



(b) Failed VNs vs. γ



(c) Nine availability CDF, $\gamma = 5$



(d) Survived B/W, $\hat{\mathbb{F}}$ vs. γ

Fig. 4: Survivability Analysis

fault tolerance (Fig. 4d) in SiMPLE is also part of its main concept, i.e., path splitting with minimal backup. 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.

VI. Related Works

SVNE literature can be broadly categorized into two classes: protection and restoration [16]. Protection is performed by provisioning additional resources as backup before any failure. Protection schemes in [7] and [27] 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 bandwidth allocated to each end-to-end path [8] or SLink [15]. These approaches do not offer bandwidth guarantee. In contrast, restoration approaches do not allocate any backup bandwidth in advance, and attempt to re-embed an affected VN after an SLink fails [21]. Such reactive approaches require time to converge, leaving VNs inactive during such periods. Rahman et al. [26] proposed a hybrid mechanism which utilizes a set of pre-computed detours to reroute the affected data streams due to an SLink failure. However, in a highly saturated SN, this mechanism may not find enough bandwidth left for the recovery.

Path splitting [33] approaches can provide some level of VN survivability without using backup resources. Oliveira et al. [24], [25] proposed multi-path embedding for VN survivability using both proactive and reactive approaches. The proactive strategy attempts to mitigate the impact of failures in one of the survived paths. On the other hand, the reactive strategy aims at partially or fully recovering the capacity of the affected SLinks through reconfiguration. However, none of these approaches can guarantee full recovery of a VLink demand even in the case of a single SLink failure.

Survivability in Optical and Multi-Protocol Label Switched (MPLS) networks is usually considered during network design. The solutions in these domains, e.g., [19], [20], [29] assume that bandwidth 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 provide the bandwidth 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.

Recently, there is a trend towards designing survivable resource allocation schemes for data center networks providing cloud services [2], [4], [32]. Xu et al. [30] proposed a scheme for provisioning virtual data centers with backup virtual machines and links. Bodik et al. [5] proposed an optimization framework for improving survivability, while reducing the total bandwidth consumption. Zhang et al. [34] proposed a framework for reliable virtual data center embedding in clouds. SiMPLE differs from these works in its objective of simultaneously optimizing VN survivability, bandwidth usage, and path splitting overhead.

VII. Conclusion and Future Work

In this paper, we have presented SiMPLE that exploits path splitting capability of an SN for survivable embedding of VNs. Compared to existing approaches, SiMPLE reserves less backup bandwidth, yet guarantees VLink survivability in presence of a single SLink failure. In case of multiple SLink failures, the survived bandwidth of the affected VLink(s) is better than that of FBS and SBS. Simulation results have demonstrated that SiMPLE 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 is 50% less than that of FBS, and lies very close to SBS. Finally, the path splitting overhead incurred by SiMPLE is compensated by guaranteed survivability, increased profit, better acceptance ratio, and lower backup bandwidth requirement. However, multi-path embedding of SiMPLE requires high path diversity in the SN, and may incur delay in large data transfers and multimedia streaming.

As a future extension of this work, we intend to evaluate the performance of SiMPLE through a prototype implementation in an SDN environment for supporting path splitting in the SN. We also would like to extend SiMPLE's link embedding concept towards a coordinated node and link mapping strategy.

VIII. Acknowledgments

This work is supported by the Natural Science and Engineering Council of Canada (NSERC) under the Smart Applications on Virtual Infrastructure (SAVI) Research Network.

References

[1] Available at: http://www.informationweek.com/it-downtime-costs-$265-billion-in-lost-revenue/d/d-id/1097919.

[2] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan. Volley: Automated data placement for geo-distributed cloud services. In *NSDI*, pages 17–32, 2010.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, Commodity Data Center Network Architecture. 38:63–74, Oct 2008.

[4] N. Bansal, R. Bhagwan, N. Jain, Y. Park, D. Turaga, and C. Venkatramani. Towards optimal resource allocation in partial fault-tolerant applications. In *IEEE INFOCOM*, 2008.

[5] P. Bodik, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica. Surviving failures in bandwidth-constrained datacenters. In *ACM SIGCOMM*, pages 431–442, 2012.

[6] N. F. Butt, N. M. M. K. Chowdhury, and R. Boutaba. Topology-awareness and reoptimization mechanism for virtual network embedding. *NETWORKING 2010*, 6091:27–39, 2010.

[7] Q. Chen, Y. Wan, X. Qiu, W. Li, and A. Xiao. A Survivable Virtual Network Embedding Scheme Based on Load Balancing and Reconfiguration. In *IEEE NOMS*, pages 1–7, Poland, May 2014.

[8] Y. Chen, J. Li, T. Wo, C. Hu, and W. Liu. Resilient Virtual Network Service Provision in Network Virtualization Environments. In *IEEE ICPADS*, pages 51–58, Shanghai, China, Dec 2010.

[9] N. M. M. K. Chowdhury and R. Boutaba. Network virtualization: state of the art and research challenges. *Communications Magazine, IEEE*, 47(7):20–26, 2009.

[10] N. M. M. K. Chowdhury and R. Boutaba. A Survey of Network Virtualization. *Computer Networks*, 54:862–876, Apr 2010.

[11] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba. Virtual Network Embedding with Coordinated Node and Link Mapping. In *INFOCOM 2009, IEEE*, pages 783–791. IEEE, 2009.

[12] J. R. Douceur. Is remote host availability governed by a universal law? *ACM SIGMETRICS PER*, 31:25–29, 2003.

[13] A. Fischer, J. F. Botero, M. T. Beck, H. D. Meer, and X. Hesselbach. Virtual Network Embedding: A Survey. *IEEE Communications Surveys and Tutorials*, 15:1888–1906, Feb 2013.

[14] P. Gill, N. Jain, and N. Nagappan. Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications. In *ACM SIGCOMM*, volume 41, pages 350–361, Aug 2011.

[15] T. Guo, N. Wang, K. Moessner, and R. Tafazolli. Shared Backup Network Provision for Virtual Network Embedding. In *IEEE ICC*, pages 1–5, Kyoto, Japan, Jun 2011.

[16] S. Herker, A. Khan, and X. An. Survey on Survivable Virtual Network Embedding Problem and Solutions. In *ICNS*, pages 99–104, Portugal, 2013.

[17] 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*, 3:299–325, 1974.

[18] S. G. Kolliopoulos and C. Stein. Improved approximation algorithms for unsplittable flow problems. In *IEEE SFCS*, pages 426–436, Oct 1997.

[19] W. Lau and S. Jha. Failure-Oriented Path Restoration Algorithm for Survivable Networks. *IEEE Transactions on Network and Service Management*, 1:11–20, Apr 2004.

[20] H. Lee, K. Lee, and E. Modiano. Cross-Layer Survivability. *Cross-Layer Design in Optical Networks*, 15:243–262, 2013.

[21] B. LU, H. Tao, S. Xiao-chuan, C. Jian-ya, and L. Yun-jie. Dynamic Recovery for Survivable Virtual Network Embedding. *The Journal of China Universities of Posts and Telecommunications*, 21:77–84, Jun 2014.

[22] J. Lu and J. Turner. Efficient Mapping of Virtual Networks onto a Shared Substrate. *Washington University (WUCSE) Tech. Rep*, 2006.

[23] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, and C. Diot. Characterization of Failures in an IP Backbone. In *INFOCOM*, volume 4, pages 123–133, Mar 2004.

[24] R. R. Oliveira, L. R. Bays, D. S. Marcon, M. C. Neves, L. S. Buriol, L. P. Gaspary, and M. P. Barcellos. DoS-resilient Virtual Networks through Multipath Embedding and Opportunistic Recovery. In *SAC*, pages 597–602, Coimbra, Portugal, Mar 2013.

[25] R. R. Oliveira, D. S. Marcon, L. R. Bays, M. C. Neves, L. S. Buriol, L. P. Gaspary, and M. P. Barcellos. No more backups: Toward Efficient Embedding of Survivable Virtual Networks. In *IEEE ICC*, pages 2128–2132, Budapest, Hungary, Jun 2013.

[26] M. Rahman, I. Aib, and R. Boutaba. Survivable Virtual Network Embedding. In *NETWORKING*, pages 40–52, May 2010.

[27] M. R. Rahman and R. Boutaba. SVNE: Survivable Virtual Network Embedding Algorithms for Network Virtualization. *IEEE Transactions on Network and Service Management*, 10:105–118, Feb 2013.

[28] A. Todimala and B. Ramamurthy. A scalable approach for survivable virtual topology routing in optical wdm networks. *IEEE Journal on Selected Areas in Communications*, 25(6):63–69, August 2007.

[29] Y. Xiong and L. G. Mason. Restoration strategies and spare capacity requirements in self-healing atm networks. *Networking, IEEE/ACM Transactions on*, 7(1):98–110, 1999.

[30] J. Xu, J. Tang, K. Kwiat, W. Zhang, and G. Xue. Survivable virtual infrastructure mapping in virtualized data centers. In *IEEE CLOUD*, pages 196–203, 2012.

[31] J. Y. Yen. An Algorithm for Finding Shortest Routes from All Source Nodes to a Given Destination in General Networks. *The Quarterly Journal of Pure and Applied Mathematics*, 27:526–530, 1970.

[32] W. Yeow, C. Westphal, and U. Kozat. Designing and embedding reliable virtual infrastructures. *ACM SIGCOMM CCR*, 41(2):57–64, 2011.

[33] M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration. In *ACM SIGCOMM CCR*, volume 38, pages 17–29, Apr 2008.

[34] Q. Zhang, M. F. Zhani, M. Jabri, and R. Boutaba. Venice: Reliable virtual data center embedding in clouds. In *INFOCOM, 2014 Proceedings IEEE*, pages 289–297, April 2014.

[35] Y. Zhu and M. H. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *INFOCOM*, pages 1–12, 2006.