Efficient Joint Approaches for Location-Constrained Survivable Virtual Network Embedding

Huihui Jiang, Long Gong, Zuqing Zhu[†]
Key Laboratory of Electromagnetic Space Information, CAS
School of Information Science and Technology, University of Science and Technology of China, Hefei, China
[†]Email: {zqzhu}@ieee.org

***Abstract*—With the development of datacenter networks and network virtualization, facility-failure-proof survivable virtual network embedding (SVNE) has recently gained notable attention. In this work, we design novel location-constrained SVNE (LC-SVNE) approaches that consider the working and backup embeddings jointly and protect virtual networks against single-facility-failures efficiently. We first transform the survivable node mapping to a bipartite graph matching problem and solve it for effective resource sharing among working and backup facilities. Then, two survivable link mapping schemes are designed to minimize the bandwidth consumption of virtual links, by leveraging anycast- and multicast-based substrate routing scenarios. Simulation results show that the proposed approaches outperform two existing ones on blocking probability and time-average revenue.**

***Index Terms*—Network virtualization, Survivable virtual network embedding (SVNE), Single facility failure.**

I. INTRODUCTION

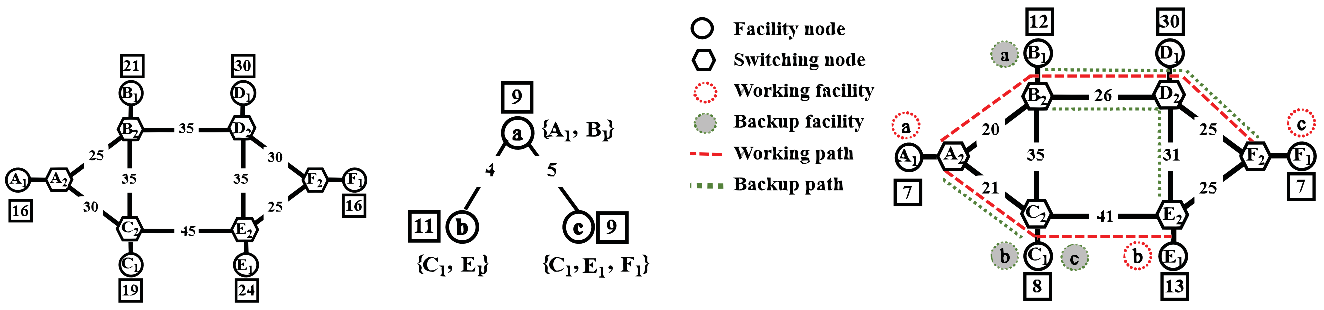
Recently, network virtualization has attracted intensive interests from both academia and industry because it allows multiple virtual networks to coexist on the same substrate infrastructure and provides a promising solution for the "ossification" of current Internet [1]. With network virtualization, the conventional Internet service providers (ISPs) evolve and split into infrastructure providers (InPs) and service providers (SPs), where the SPs lease substrate infrastructure from the InPs, construct virtual networks (VNTs), and provision various applications to the end users. For instance, in an inter-datacenter (inter-DC) network, the InP owns the infrastructure, while one or more SPs lease both bandwidth resources on the substrate links and computing/storage resources in the DCs to construct VNTs for the applications such as video streaming, e-Science and etc. Typically, this procedure of provisioning VNTs over substrate infrastructure involves virtual network embedding (VNE) [2], which leverages node mapping and link mapping and builds VNTs according to the SP's requirement.

Nevertheless, sharing the substrate infrastructure among SPs brings critical survivability issues. Specifically, a single substrate network failure (*e.g.*, link cut, power outage, system crash and equipment broken-down) can bring down the services of multiple SPs. We still use the network virtualization in an inter-DC network as the example. In such a network, since it is centralized from the geographical point of view and usually carries massive data and applications, a DC (facility node) is vulnerable to nature disasters and human errors [3]. A single facility node failure can cause long service interruption and huge data losses to multiple SPs. To this end, it is desired that VNE considers network resilience and ensures certain VNT survivability against unexpected substrate failures, and such a VNE approach is referred as survivable VNE (SVNE) [4].

Previously, people have designed SVNE with link protection to address substrate link failures [5, 6], and in [7], how to overcome switching node failures was also studied and SVNE that adopted pre-configured cycles (*p*-cycles) was proposed. These studies just simply extended the corresponding protection schemes in conventional networks. The work in [8–10] considered region failures (*i.e.*, multiple node- and link-failures that are geographically-correlated in the substrate network), and proposed SVNE approaches to protect VNTs against them. Note that in a wide-area inter-DC network, region failures are really rare, while these approaches have to reserve a fairly amount of substrate resources for backup.

Since in an inter-DC network, the facility nodes (*i.e.*,DCs) are very valuable and vulnerable assets, facility-failure-proof SVNE has recently gained notable attention. The SVNE approaches that can protect against single-facility-failures have been investigated in [11–13]. In [11], Yu *et al.* studied the shared protection schemes and formulated a mixed integer linear programming (MILP) model. By leveraging the multi-commodity flows, the authors of [12] formulated two integer linear programming (ILP) models to consider both the splittable and non-splittable flows for SVNE. Note that these two studies considered the resource allocations for working and backup separately, and thus might result in sub-optimal solutions for SVNE, as we will show later in this paper. The study in [13] proposed failure dependent protection (FDP) for SVNE, which rearranges all the node mappings of a VNT when a failure happens. Even though the global node re-mapping in FDP leads to efficient resource utilization during restoration, the significantly increased number of service migrations can become an issue. More recently, Khan *et al.* considered multiple-facility-failures and designed an SVNE algorithm that provided one-to-one facility protection [14].

In this work, we design novel location-constrained SVNE (LC-SVNE) approaches that consider the working and backup embeddings jointly and protect VNTs in an inter-DC network against single-facility-failures efficiently. In the node mapping, we add backup facility nodes with location constraints to address geographically-correlated facility failures and transform



(a) Substrate Network (SNT) (b) Virtual network (VNT) (c) LC-SVNE result

Fig. 1. Example of LC-SVNE.

the joint embedding to a bipartite graph matching problem. Then, two joint link mapping schemes are designed to minimize the bandwidth consumption of VNTs, by using anycast- and multicast-based substrate routing scenarios. Simulation results show that the proposed approaches outperform two existing ones on blocking probability and time-average revenue.

The rest of the paper is organized as follows. We describe the network models and formulate the problem of LC-SVNE in Section II. Section III discusses our proposed LC-SVNE approaches, and the performance evaluation with simulations is showed in Section IV. Finally, Section V summarizes the paper.

II. Network Models and Problem Description

A. Network Model

1) Substrate Network: In the LC-SVNE problem, the substrate network (SNT) is modeled as an undirected graph, $G^s(V^s_f, V^s_s, E^s)$, where E^s represents the set of substrate links (SLs), and V^s_f and V^s_s are the sets of facility nodes (FNs) and switching nodes, respectively. We assume that there are two types of substrate nodes (SNs) in the SNT, *i.e.*, FNs and switching nodes. The switching nodes are in charge of network connectivity and each of them connects to an FN locally. Note that similar to the studies in [11, 15], we only consider FN failures, while assume that the switching nodes are operational all the time. For instance, human misconduct or server broken-down can make a DC unavailable, while the local networking equipments located in a separate switching office can be intact. For each FN $v^s \in V^s_f$, we define $c^s_{v^s}$ and $l^s_{v^s}$ as its computing capacity and physical location, respectively, while for each SL $e^s \in E^s$, its bandwidth capacity is denoted as $b^s_{e^s}$. Fig. 1(a) shows an example of the SNT, in which the circular nodes are the FNs (*e.g.*, A_1) while the hexagon ones (*e.g.*, A_2) represent the switching nodes. The number in the square around each FN describes its computing capacity, and those on the SLs are the corresponding bandwidth capacities.

2) Virtual Network: Each virtual network (VNT) can also be modeled as an undirected graph, $G^r(V^r, E^r)$. For each virtual node (VN) $v^r \in V^r$ and virtual link (VL) $e^r \in E^r$, we use notations $c^r_{v^r}$ and $b^r_{e^r}$ for the computing and bandwidth requirements, respectively. Besides, each VN v^r has a preferred location $l^r_{v^r}$, which points to a set of candidate FNs, denoted as Φ_{v^r}[1]. Fig. 1(b) shows a VNT. The letters in the brace around each VN form its candidate FN set, the number in the square around each VN is the computing requirement, and the bandwidth requirements are labeled on each VL.

B. Problem Description

Basically, LC-SVNE needs to allocate resources in the SNT properly to build working VNTs that satisfy the computing and bandwidth requirements. Meanwhile, backup resources should be reserved to protect the VNTs against single-facility-failures. The problem has two components, *i.e.*, survivable node mapping and survivable link mapping.

1) Survivable Node Mapping: The LC-SVNE tried to find two distinct FNs both with enough computing capacity for each VN as its working and backup facilities. This process can be considered as two related one-to-one mappings, *i.e.*,

$$\left.\begin{array}{lcl} f^w_N(v^r) & = & v^s_1, \\ f^b_N(v^r) & = & v^s_2, \end{array}\right\} \quad v^r \in V^r,\ v^s_1, v^s_2 \in V^s_f, v^s_1 \neq v^s_2, \quad (1)$$

while satisfying the following constraints:

- For each VN, both the working and backup facilities should belong to its candidate FN set,
$$v^s_1, v^s_2 \in \Phi_{v^r}, \quad \forall v^r \in V^r. \quad (2)$$
- The computing resource allocation on each VN's working and backup facilities should satisfy the requirement,
$$\left.\begin{array}{lcl} c^s_{v^s_1} & \geq & c^r_{v^r}, \\ c^s_{v^s_2} & \geq & c^r_{v^r}, \end{array}\right\} \quad \forall v^r \in V^r. \quad (3)$$
- Each VN's working facility is not sharable, while a backup facility can be shared by multiple VNs.

For example, in Fig. 1(c), the nodes mapping for working facilities is $\{a \to A_1, b \to E_1, c \to F_1\}$, while the one for backup facilities is $\{a \to B_1, b \to C_1, c \to C_1\}$. *FN* C_1 is the shared backup facility for *VNs* b and c.

2) Survivable Link Mapping: In order to ensure the survivability against single-facility-failures, LC-SVNE builds substrate paths to guarantee the bandwidth requirement between any two VNs in a VNT, considering both the working and backup facilities. Suppose $f_L(e^r) \subset E^s$ denotes the set of SLs that VL e^r is embedded on, and let $G^i_{e^r}$ denote the subgraph of G^s whose frontier set is $f_L(e^r)$ and node set is the FNs for the

[1]Note that, in the rest of the paper, we use the preferred FN set instead of the location to represent the location constraint.

end-nodes of e^r, respectively. Here, the FNs for the end-nodes of e^r are $\{f_N^w(e_+^r), f_N^w(e_-^r), f_N^b(e_+^r), f_N^b(e_-^r)\}$, if in the VNT, the end-nodes of VL e^r are e_+^r and e_-^r, *i.e.*, $e^r = (e_+^r, e_-^r)$. Here, $G_{e^r}^i$ should satisfy the following constraints,

$$\left.\begin{matrix} MF(G_{e^r}^i, f_N^w(e_+^r), f_N^w(e_-^r)) \\ MF(G_{e^r}^i, f_N^w(e_+^r), f_N^b(e_-^r)) \\ MF(G_{e^r}^i, f_N^b(e_+^r), f_N^w(e_-^r)) \end{matrix}\right\} \geq c_{e^r}^r, \quad \forall e^r \in E^r, \quad (4)$$

where $MF(G_{e^r}^i, u, v)$ is the maximum flow from *Node* u to *Node* v in $G_{e^r}^i$. In the link mapping in Fig. 1(c), the red dash lines are for the working substrate paths, *i.e.*, $\{(a,b) \rightarrow (A_1, C_2, E_1), (a,c) \rightarrow (A_1, B_2, D_2, F_1)\}$, while the green dot lines are for the backup ones, *i.e.*, $\{(A_1, C_1), (B_1, D_2, E_1)\}$ protect *VL* (a,b) and (B_1, D_2, F_1) protects *VL* (a,c). Sharing is considered on the SL (B_1, D_2) and (D_2, F_1) since *VL* (a,c) and its backup path go through these links.

III. LC-SVNE Algorithms

A. Survivable Node Mapping

In the survivable node mapping, each VN needs to be embedded onto two FNs as its working and backup facilities. We consider joint embedding and construct an auxiliary graph to assist it. Therefore, we can avoid the situations where backup FNs cannot be embedded due to the improper selection of working FNs. For instance, in Fig. 1(c), if we embed the working facilities of a, b and c onto *FNs* A_1, C_1 and E_1, respectively, the remaining FNs will not have sufficient resources for accommodating the backup facilities.

For the auxiliary graph, we define a weight w_{v^s} for each FN $v^s \in V_f^s$ to quantify its embedding potential,

$$w_{v^s} = c_{v^s}^s \cdot \sum_{v^{s\prime} \in e^s, e^s \neq (v^s, v^{s\prime})} b_{e^s}^s, \quad (5)$$

where $v^{s\prime}$ is the switching node that connects to v^s locally, and $v^{s\prime} \in e^s$ means that $v^{s\prime}$ is an end-node of *SL* e^s. In this work, we assume that there is always enough bandwidth on the SL between a switching node and its local FN. Therefore, in Eq. (5), we ignore *SL* $(v^s, v^{s\prime})$ when calculating w_{v^s}.

Algorithm 1 shows the detailed procedure that consists of two phases for constructing the auxiliary graph G^a. *Lines* 1-19 describe the phase for working facilities. *Lines* 2-8 update Φ_{v^r} for each $v^r \in V^r$ and delete those candidate FNs that have insufficient computing resources. Then, for each v^r, *Line* 12 inserts a node in U^a to represent its working facility, and *Line* 14 adds a node in V^a for each candidate FN $v^s \in \Phi_{v^r}$. The nodes in U^a and V^a are then connected with weighted links as in *Lines* 15-16. Fig. 2(a) shows the G^a for the example in Fig. 1 after this phase, where the nodes in the left and right columns are for U^a and V^a, respectively. The phase for backup facilities is in *Lines* 20-33. Basically, in order to maximize the protection efficiency, *Lines* 22-25 choose the FNs that can be shared by the most VNs as their backup facility candidates, and store them in $\widehat{\Phi}$. Then, *Lines* 26-30 insert the nodes for VNs' backup facilities in U^a and connect them to the corresponding FNs in $\widehat{\Phi}$. These steps are repeated until all the backup facility nodes for $v^r \in V^r$ are inserted in G^a. Fig. 2(b) shows the G^a after this second phase, which indicates that *VNs* b and c can use *FNs* C_1 and E_1 for shared protection.

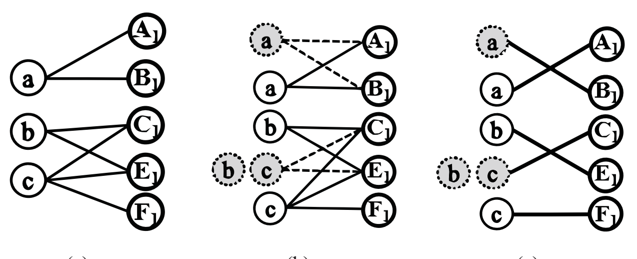
Algorithm 1: Auxiliary Graph Construction

```
input : Virtual network $G^r$, substrate network $G^s$
output: Auxiliary graph $G^a(U^a, V^a, E^a)$
// Working Facility related Graph Construction:
1  foreach $v^r \in V^r$ do
2      foreach $v^s \in \Phi_{v^r}$ do
3          if $c_{v^s}^s < c_r^r$ then
4              $\Phi_{v^r} = \Phi_{v^r} \setminus \{v^s\}$;
5          else
6              $p_{v^s} = p_{v^s} + 1$;
7          end
8      end
9      if $|\Phi_{v^r}| < 2$ then
10         return(FAILURE);
11     else
12         add a node in $U^a$ for working facility of $v^r$;
13         foreach $v^s \in \Phi_{v^r}$ do
14             add a node in $V^a$ to represent $v^s$;
15             connect the nodes for $v^r$ and $v^s$ in $G^a$;
16             mark the new link's weight as $w_{v^s}$;
17         end
18     end
19 end
// Backup Facility related Graph Construction:
20 $V_t = V^r$;
21 while $V_t \neq \emptyset$ do
22     $V_t^s = \{v^s : v^s \in \Phi_{v^r}, v^r \in V_t\}$;
23     $v_t^s = \arg\max_{v^s \in V_t^s} p_{v^s}$;
24     $V_t^r = \{v^r : v_t^s \in \Phi_{v^r}, v^r \in V_t\}$;
25     $\widehat{\Phi} = \bigcap_{v^r \in V_t^r} \Phi_{v^r}$;
26     foreach $v^r \in V_t^r$ do
27         add a node in $U^a$ for backup facility of $v^r$;
28         connect the node and those for $v^s \in \widehat{\Phi}$ in $G^a$;
29         mark each new link's weight as $w_{v^s}$;
30     end
31     $V_t = V_t \setminus V_t^r$;
32 end
33 return($G^a$);
```

With G^a, we transfer the survivable node mapping to a maximum weighted bipartite graph matching problem. Specifically, for each VN node in U^a, we find a connected FN node in V^a such that the total weight of all the selected links is the maximum, and each VN uses different FNs as its working and backup facilities. We use the Hungarian method in [16] to solve this problem, and Fig. 2(c) shows the result for the case in Fig. 2(b).

B. Survivable Link Mapping

During a single-facility-failure, the corresponding backup FN is activated for restoration. Therefore, the survivable link mapping needs to ensure that there are enough bandwidths



(a) (b) (c)

Fig. 2. Survivable node mapping with the help of an auxiliary graph; (a) Original bipartite graph, (b) Enhanced bipartite graph , (c) Maximum matching

reserved in the SNT for the backup FN and it can still communicate with the rest of working FNs in the VNT. Here, in order to improve the protection efficiency, we still utilize shared protection. Again, joint embedding of the working and backup VLs is considered and we design two schemes based on anycast- and multicast-based substrate routing, respectively.

1) Anycast-based Substrate Routing (AC): In order to achieve effective bandwidth sharing between working and backup substrate paths, we design a survivable link mapping scheme that leverages anycast-based substrate routing (AC). *Algorithm 2* shows the detailed procedure. Firstly, *Lines* 1-10 embed all the working VLs with shortest-path routing. Then, *Lines* 11-34 accomplish backup VL embedding with AC. Basically, when the working substrate path of a VL e^r is determined, each of its end-VN's backup FN can reach the working FN of the other end-VN by connecting to any switching node on the working path. Therefore, the bandwidths on the working path can be shared by the backup one. In order to achieve the most bandwidth sharing between working and backup, we find a path between an end-VN's backup FN and any switching node on the working path and make sure that it has the smallest hop-count (as shown in *Lines* 16-33), which can be considered as a typical anycast routing problem.

Fig. 3(a) illustrates an intuitive example for AC. *VL* (a, b) is embedded onto the working path (A_1, C_2, E_1), and the backup FN for a is B_1. If we set up a substrate path to connect B_1 and C_2, then when A_1 fails, the backup substrate path for (a, b) is (B_1, C_2, E_1). Hence, the bandwidth resources allocated on (C_2, E_1) is effectively shared between working and backup.

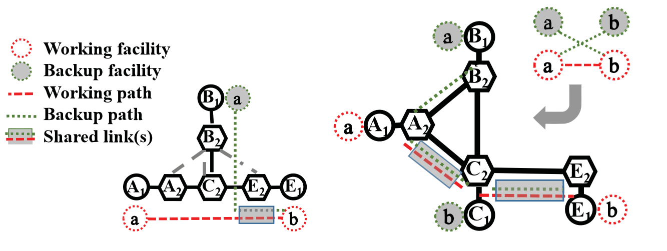
2) Multicast-based Substrate Routing (MC): This scheme processes each VL in the VNT independently. The basic idea is that since the two end-VNs of a VL are embedded onto 3 or 4 FNs[2],there would be 3 substrate paths (*i.e.*, one working and two backup ones) for the same VL, and they can share the bandwidth resource, which can be considered a multicast from one working FN to another working FN and two backup ones[3]. Fig. 3(b) shows an intuitive example. For *VL* (a, b), the corresponding FNs are A_1, B_1, E_1 and C_1, and thus $(A_1/A_2, B_1/B_2)$, $(A_1/A_2, E_1/E_2)$ and $(A_1/A_2, C_1/C_2)$ to ensure that (a, b) is operational all the time. *Algorithm 3* shows

[2]In case of shared protection, we only need 3 FNs to cover the working and backup facilities of the two end-VNs.

[3]Note that this work only considers single-facility-failures, and thus there will be no substrate link failures. Hence, the substrate paths can share the same bandwidths with each other, as at any time, only one of them is active.

Algorithm 2: AC-based Survivable Link Mapping

```
   input : Virtual network $G^r$, substrate network $G^s$,
           node mapping $\{f_N^w(v^r), f_N^b(v^r)\}$ for $v^r \in V^r$
   // Working VL Embedding:
 1 foreach $e^r \in E^r$ do
 2     $v_1^r = e^{r+}$, $v_2^r = e^{r-}$, $v_1^s = f_N^w(v_1^r)$, $v_2^s = f_N^w(v_2^r)$;
 3     cut all $e^s$ whose $b_{e^s}^s < b_{e^r}^r$ in $G^s$ temporally;
 4     obtain the shortest path $\mathcal{P}$ for $(v_1^s, v_2^s)$ in $G^s$;
 5     if $\mathcal{P}$ cannot be found then
 6         return(FAILURE);
 7     else
 8         allocate $b_{e^r}^r$ on all $e^s$ on $\mathcal{P}$;
 9     end
10 end
   // Backup VL Embedding:
11 foreach $e^r \in E^r$ do
12     $v_1^r = e^{r+}$, $v_2^r = e^{r-}$, $v_1^s = f_N^b(v_1^r)$, $v_2^s = f_N^b(v_2^r)$;
13     cut all $e^s$ whose $b_{e^s}^s < b_{e^r}^r$ in $G^s$ temporally;
14     retrieve the working substrate path $\mathcal{P}$ for $e^r$;
15     $\mathcal{P}_1 = \emptyset$, $\mathcal{P}_2 = \emptyset$, $h_1 = +\infty$, $h_2 = +\infty$;
16     foreach $v^s$ on $\mathcal{P}$ do
17         for $i = 1$ to 2 do
18             obtain the shortest path $\mathcal{P}_i^t$ for $(v_i^s, v^s)$;
19             if $\mathcal{P}_i^t$ cannot be found then
20                 $h_i^t = +\infty$;
21             else
22                 set $h_i^t$ as the hop-count of $\mathcal{P}_i^t$;
23             end
24             if $h_i^t < h_i$ then
25                 $h_i = h_i^t$, $\mathcal{P}_i = \mathcal{P}_i^t$;
26             end
27         end
28         if $\mathcal{P}_1 = \emptyset$ OR $\mathcal{P}_2 = \emptyset$ then
29             return(FAILURE);
30         else
31             allocate $b_{e^r}^r$ on all $e^s$ on $\mathcal{P}_1$ and $\mathcal{P}_2$;
32         end
33     end
34 end
```



(a) Anycast-based Routing (b) Multicast-based Routing

Fig. 3. Survivable link mapping.

the detailed procedure for MC-based survivable link mapping.

IV. Performance Evaluation

A. Simulation Setup

The simulations generate the SNT and VNTs randomly with the GT-ITM tool [17]. The SNT is located within a 100×100

Algorithm 3: MC-based Survivable Link Mapping

```
input : Virtual network $G^r$, substrate network $G^s$,
        node mapping $\{f_N^w(v^r), f_N^b(v^r)\}$ for $v^r \in V^r$
1  foreach $e^r \in E^r$ do
2      $v_1^r = e^{r+}$, $v_2^r = e^{r-}$;
3      $V_t = \{f_N^w(v_1^r), f_N^b(v_1^r), f_N^w(v_2^r), f_N^b(v_2^r)\}$;
4      cut all $e^s$ whose $b_{e^s}^s < b_{e^r}^r$ in $G^s$ temporally;
5      obtain a Steiner tree $\mathcal{T}$ in $G^s$ to cover $V_t$;
6      if $\mathcal{T}$ cannot be found then
7          return(FAILURE);
8      else
9          allocate $b_{e^r}^r$ on all $e^s$ on $\mathcal{T}$;
10     end
11 end
```

TABLE I
SIMULATION PARAMETERS

Initial FN computing capacity	$[50, 100]$ units
Initial SL bandwidth	$[50, 100]$ units
Number of VNs in each VNT	$[2, 10]$
VNs' connectivity ratio	0.5
VN computing requirement	$[1, 10]$ units
VL bandwidth requirement	$[1, 10]$ units

grid, with 50 FNs and 147 SLs. For each VN, five candidate FNs are randomly selected in average, within a preset radius. VNT requests come in following the Poisson process with an average rate of λ per time-unit, while the lifetime of each request follows the negative exponential distribution with an average of $\frac{1}{\mu}$ time-units. Hence, the traffic load is $\frac{\lambda}{\mu}$ in Erlangs. The rest simulation parameters are shown in Table I.

B. Performance Metrics

We evaluate the LC-VNE algorithms with the following three performance metrics,

- Blocking Probability: the ratio of blocked to total number of VNT requests in each simulation.
- Resource Protection Efficiencies:
$$r_N = \frac{C_w}{C_b} \quad (6)$$
$$r_L = \frac{B_w}{B_b} \quad (7)$$
where r_N and r_L are the node- and link-resource protection efficiencies, respectively. In each simulation, C_w and B_w are the total computing and bandwidth resources allocated for working, and C_b and B_b are the total computing and bandwidth resources for backup. r_N and r_L measure the efficiency of shared protection, as the higher they are, the more effective the shared backup is.
- Time-Average Revenue:
$$R_{avg} = \lim_{T\to\infty} \frac{SUM(R_{VNR})}{T} \quad (8)$$
where, $R_{VNR} = \alpha \sum c_{v^v}^v + \beta \sum b_{v^v}^v$. We adopt the conventional revenue model in VNE [18] since there is not a good revenue model in SVNE. T is the simulation time in terms of time-units. We set α and β 1 in our simulation.

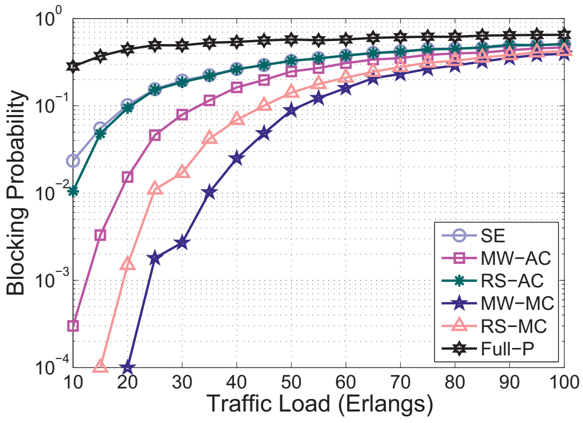


Fig. 4. Results on blocking probability.

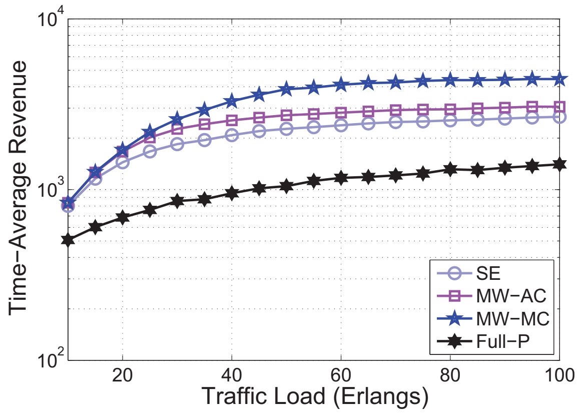


Fig. 5. Results on time-average revenue.

C. Comparison Algorithms

We implement two algorithms from previous work as the benchmarks, *i.e.*, the sequential LC-SVNE (*SE*) in [15] and the one-to-one protection for multiple-facility-failures (*Full-P*) in [14]. Even though we only consider single-facility-failures in this work, we still include *Full-P* as a benchmark because we would like to study the tradeoff between partial- and full-protection schemes for LC-SVNE. For our proposed algorithms, we denote those that use bipartite graph matching with maximum weight for node mapping as *MW-AC* and *MW-MC*, for AC- and MC-based link mapping, respectively. In order to investigate the algorithms' performance thoroughly, we also incorporate the node mapping schemes that uses bipartite graph matching without weight, we called random selection and the corresponding LC-SVNE algorithms are denoted as *RS-AC* and *RS-MC.*

D. Performance Comparisons

Fig. 4 shows the results on blocking probability. As expected, *Full-P* provides the highest blocking probability because it reserves the most resources for each VNT to achieve one-to-one protection. Among the algorithms that target for single-facility-failure-proof, *RS-AC* and *ES* have comparable blocking performance, while *MW-WC* provides the lowest blocking probability. For the four proposed algorithms, we observe that the results from MC-based ones are always lower

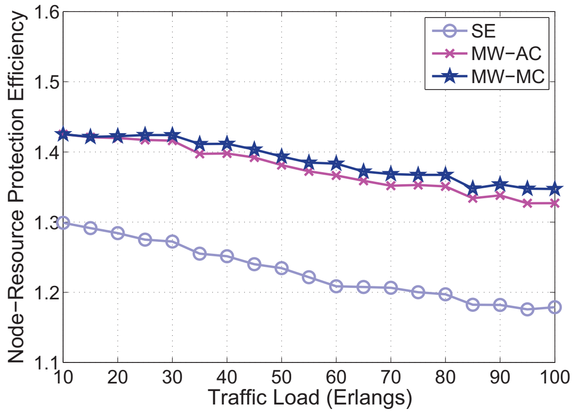


Fig. 6. Results on node-resource protection efficiency.

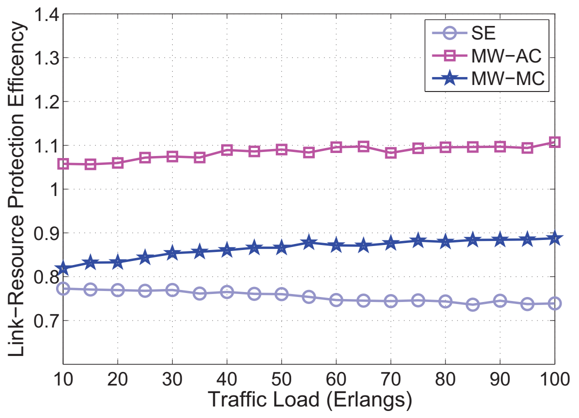


Fig. 7. Results on link-resource protection efficiency.

than those from AC-based ones, in terms of link mapping. This is because AC-based algorithms always use the shortest path in the SNT for working VL embedding, which can generate bottleneck on certain SLs. While MC-based ones optimize the link mapping for working and backup jointly. In terms of node mapping, the MW-based ones always outperform RS-based ones, for the reason that the bipartite graph matching with maximum weight considers the embeding potential of each SN proactively and addresses the "big island" problem [19] properly.

The results on time-average revenue are in Fig. 5, which indicates that *MW-WC* always generates the highest revenue and when the traffic load is 100 Erlangs, its revenue is around three times higher than that of *Full-P*. Therefore, in order to provide high-order resilience against multiple-facility-failures, an InP has to sacrifice a significant amount of revenue. Figs. 6 and 7 show the results on node- and link-resource protection efficiencies, *i.e.*, r_N and r_L, respectively. It can be seen that *MW-MC* provides the highest r_N, *MW-AC*'s r_L is the highest, while both of them achieve higher r_N and r_L than *ES*.

V. Conclusion

We designed novel LC-SVNE approaches that considered the working and backup embeddings jointly and protect VNTs against single-facility-failures efficiently. We first transformed the survivable node mapping to a bipartite graph matching problem and solved it for effective resource sharing among working and backup facilities. Then, two survivable link mapping schemes were designed to minimize the bandwidth utilization of VLs, by leveraging anycast- and multicast-based substrate routing scenarios. Simulation results indicated that the proposed approaches outperformed two existing ones on VNT blocking probability and time-average revenue.

Acknowledgments

This work was supported in part by NCET Project NCET-11-0884, NSFC Project 61371117, Fundamental Research Funds for Central Universities (WK2100060010), and Strategic Priority Research Program of CAS (XDA06010302).

References

[1] M. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, pp. 862–876, Apr. 2010.
[2] M. Chowdhury, M. Rahman, and R. Boutaba, "VineYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, pp. 206–219, Feb. 2012.
[3] Calculating the cost of data center outages. [Online]. Available: http://www.emersonnetworkpower.com/fr-EMEA/Latest-Thinking/white-papers/Pages/calculating-the-cost.aspx
[4] S. Herker, A. Khan, and X. An, "Survey on survivable virtual network embedding problem and solutions," in *Proc. of ICNS 2013*, pp. 99–104, Mar. 2013.
[5] A. Jarray, Y. Song, and A. Karmouch, "Resilient virtual network embedding," in *Proc. of ICC 2013*, pp. 3461–3465, Jun. 2013.
[6] M. Rahman and R. Boutaba, "SVNE: Survivable virtual network embedding algorithms for network virtualization," *IEEE Trans. Netw. Service Manag.*, vol. 10, pp. 105–118, Jun. 2013.
[7] A. Jarray, Y. Song, and A. Karmouch, "*p*-Cycle-based node failure protection for survivable virtual network embedding," in *Proc. of IFIP 2013*, pp. 1–9, May 2013.
[8] H. Yu *et al.*, "Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures," in *Proc. of GLOBECOM 2010*, pp. 1–6, Dec. 2010.
[9] F. Gu, H. Alazemi, A. Rayes, and N. Ghani, "Survivable cloud networking services," in *Proc. of ICNC 2013*, pp. 1016–1020, Feb. 2013.
[10] M. Habib, M. Tornatore, F. Dikbiyik, and B. Mukherjee, "Disaster survivability in optical communication networks," *SIGCOMM Comput. Commun. Rev.*, vol. 36, pp. 630–644, Mar. 2013.
[11] H. Yu, V. Anand, C. Qiao, and G. Sun, "Cost efficient design of survivable virtual infrastructure to recover from facility node failures," in *Proc. of ICC 2011*, pp. 1–6, Jun. 2011.
[12] Q. Hu, Y. Wang, and X. Cao, "Survivable network virtualization for single facility node failure: A network flow perspective," *Opt. Switch. Netw.*, vol. 10, pp. 406–415, Nov. 2013.
[13] B. Guo *et al.*, "Survivable virtual network design and embedding to survive a facility node failure," *J. Lightw. Technol.*, vol. 32, pp. 483–493, Feb. 2014.
[14] A. Khan, X. An, D. Perez-Caparros, and W. Kiess, "Virtual network embedding algorithm for one-to-one site protection," in *Proc. of GLOBECOM 2013*, pp. 1–6, Dec. 2013.
[15] Q. Hu, Y. Wang, and X. Cao, "Location-constrained survivable network virtualization," in *Proc. of SARNOFF 2012*, pp. 1–5, May 2012.
[16] D. West, *Introduction to Graph Theory - Second Edition*. Prentice Hall, 2001.
[17] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. of INFOCOM 1996*, pp. 594–602, Mar. 1996.
[18] M. Rahman, I. Aib, and R. Boutaba, "Survivable virtual network embedding," in *Proc. of NETWORKING 2010*, pp. 40–52, May 2010.
[19] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. of INFOCOM 2014*, pp. 1–9, Apr. 2014.