One Node-Fault Tolerant Graph with Node's types for survivable Virtual Network Request with Service Type

Kun Xie[1], *Member, IEEE*,Heng Tao[1]
[1] College of Computer Science and Electronic Engineering,Hunan University, China

***Abstract*—As virtualization is becoming a promising way to support various emerging application, provisioning survivability to requested embeded virtual network(VN) which is embedded in substrate networks (SN) in a resource efficient way is important. In this paper, First, we consider the failure dependent protection (FDP) in which each primary facility node would have a different backup facility node, as opposed to the Failure Independent Protection (FIP) which has been studied before, in order to provide the same degree of protection against a single node failure with less substrate resources. Secondly, we enhance the embeded VN with additional computing and communication resources and design the Enhanced eVN (or EeVN) after embedding it to the substrate in order to further reduce the amount of substrate resources needed to survive a single facility node failure. In a virtualized infrastructure where physical resources are shared, a single physical server failure will terminate several virtual serves and crippling the virtual infrastructures which contained those virtual servers. To guarantee some level of reliability, each virtual infrastructure, at instantiation, should be augmented with backup virtual nodes and links.**

I. Introduction

In network virtualization, the primary entity is the Virtual Network (VN). A VN is a combination of active and passive network elements (network nodes and network links) on top of a Substrate Network (SN). Virtual nodes are interconnected through virtual links, forming a virtual topology. By virtualizing both node and link resources of a SN, multiple virtual network topologies with widely varying characteristics can be created and co-hosted on the same physical hardware.

With network infrastructure rapidly becoming virtualized, shared and dynamically changing, it is essential to have strong reliability support from the physical infrastructure, since a single physical server or link failure affects several shared virtual entities. Providing reliability is often linked with over-provisioning both computational and network capacities, and employing load balancing for additional robustness. However, we do not talk about the capacity requirement and geographical location requirement of virtual node, and the bandwidth capacity requirement of virtual link in this paper.

However, with network virtualization gaining momentum, the survivability challenges in VNE should also be well investigated. In a large networked computing system, hardware and software failures of facility nodes and communication resources (e.g., links and switching nodes) are norm instead of exception, such as power outages caused by virus attack, disk failures, misconfiguration or fiber cut [1], [2], [3], [4], [5]. Such failures will force the virtual node (links) assigned to it to be migrated/re-embeded to another facility node at a geographically different location (link disjoint substrate path). This means that, in order to survive from the disruptions due to such failures, one must reserve redundant facility nodes and bandwidth on fiber links such that after any failure, there are adequate remaining computing and networking resources to migrate/remap the VN request. Accordingly, the problem of minimizing the resources, including computing (CP.) and communication (CM.) resources, reserved for VN request to tolerate substrate failures, (hereafter called Survivable VNE problem, SVNE) is both critical and challenging. Actually, SVNE problem is quite different with the protection approach in IP over WDM network by investigating facility node failure caused virtual node failure problem and employing node migration induced VN remapping as an unique recovery strategy.

In terms of SVNE capable of recovering/re-embedding the task nodes after a facility node failure, there are two basic approaches: Failure Dependent Protection (FDP [6] and Failure Independent Protection (FIP) [7], and the differences between them are as follows. In FIP, a host (facility) node is assigned and dedicated to backup all working host (facility) nodes. That is, no matter which working host node fails, the affected task node will be migrated to the only one backup host node. On the other hand, with FDP, each working host node can have a different backup host node under different failure scenarios. In fact, after a failure, even an unaffected task node may be migrated from a working host node to its corresponding backup host node, as a result of re-embedding the entire task graph. In other words, FDP could provide more flexibility in survivable VN designing by allowing task nodes migrating freely after failure, so FIP could be considered as a special case of FDP and FDP is expected to use fewer resources at the cost of more task nodes migrations after a failure.

To survive a substrate link failure, pre-computed alternative paths inVN are used in general and the bandwidths are allocated before or after a failure. For instance, the reactive detour solution is employed after a substrate link failure in [2], while authors in [4] propose a proactive backup approach (considering the backup resources sharing) to avoid service disruption in reactive restoration approach and improve the substrate resource utilization.

Through synchronization [8], [9] and migration techniques [10], [11] on virtual machines and routers, we suppose that every virtual network have specific service functions, in addition that fault tolerant can be introduced at the virtualization layer [7].

As nodes often represent expensive components (servers) and

edges represent less expensive interconnections (links), most attention has been devoted to node faults, i.e., the removal of nodes (and their incident edges), rather than edge faults where only edges are removed.

Our paper focus on the problem of resource deployment for virtual network infrastructure with reliability guarantee. Practical issue such as forwarding graph embedding and chain composition.

The organization os this paper is as follow. In the next section, we briefly describe the background, notations and define reliability in Sec.II.

II. Problem Formulation

We consider a resource deployment problem in a virtualized network infrastructure, such as a data center, where the virtualized resources can be leased with reliability guarantees. Each resource lease request is modeled as undirected graph $G = (V, E, S)$. V is a set of compute nodes, E is a set of edges and S is a types set of node V service functions. We call this a virtual network(VN).

A. Reliability,Survivable

Reliability is guaranteed on the set of critical nodes $C \subset V$ of a VN through redundant virtual nodes with backup nodes set $B(V, S)$. A backup (redundant) node b_i must be able to assume full execution function of a failed critical node c_i. Hence, the backup node must have sufficient resources in terms of computation resource.

B. How many backups?

The number of backup nodes depend on the physical mapping, and the failure models of both the physical nodes and the virtual infrastructure.

C. Node-Fault Graph

1) NFT

We refer to the concept of k-NFT [12]. Let $G(V, E)$ be a graph with n nodes and q edges. An(n+k)-node graph G^* is k-node fault-tolerant, or k-NFT, with respect to G if every graph $G^* - R$ obtained by removing any nodes set R of $k > 0$ nodes from G^* contains G. Generally speaking, G is subgraph isomorphism of $G^* - R$. We will refer to G^* as a k-NFT supergraphs of G or simply as a k-NFT(G). We also say $G^* \cong$ k-NFT(G), the set of all k-NFT(G) supergraphs of G.

The complete graph K_{n+k} of $n + k$ nodes is trivially a k-NFT supergraph of every G that contains up to n nodes. We are concerned mainly with k-NFT graphs that satisfy the following optimality criterion: If G^* has the smallest number $|E(G^*)|$ of edges among all (n + k)-node supergraphs that are k-NFT with respect to G, then G^* is optimally k-NFT with respect to G. The number NFT_{ec}(G,k) =$|E(G^*) - E(G)|$ is called the k-NFT edge cost of G. The number NFT_{nc}(G,k) =$|V(G^*) - V(G)|$ is called the k-NFT node cost of G, however NFT_{nc}(G,k) = k with respect with k node fault tolerant graph of graph G .

2) SNFT

When every nodes of graph $G(V, E, S)$ have a specific flags set S(corresponding to service functions) and every nodes is marked by only one specific flag, which is denoted by node flag. An (n+k)-node graph G^o is k-specific node fault tolerant, or k-SNFT, with respect to $G(V, E, S)$ if every graph $G^o - R$ obtained by removing any nodes set R of $k > 0$ nodes from G^o contains G, G is subgraph isomorphism of $G^o - R$ and the node flag of any nodes n of G belong specific flags set of node n^o corresponding to node n. We will refer to G^o as a k-SNFT supergraphs of G or simply as a k-SNFT(G). The number $SNFT_{nc}$(G,k) =$|V(G^o) - V(G)|$ is called the k-SNFT node cost of G. The number $SNFT_{ec}$(G,k) =$|E(G^o) - E(G)|$ is called the k-SNFT edge cost of G. every node of added nodes contain all specific flags.

3) SNFT with B

After our simple inference, if we set the $SNFT_{nc}$(G,k) is the main cost than edge cost, moreover the added node could have any specific flags. but the added node should be limited in specific flags set after our analyzing the real-world practical phenomenon.

Suppose added nodes is from backup nodes set $B(V, S)$, where every nodes of backup nodes set B have a specific flags set. The k-SNFT(G, B) is that requesting a k specific node fault tolerant graph of graph G and added nodes belong backup nodes set B.

D. Complexity of Problem

This computation complexity of k-SNFT problem k-NFT($G(V, E, S)$) is NP, whose proof is below. when just exist one service function type in SVNR, the problem is degenerated into one-node fault-tolerant graph problem. when VN just have one type of specify flags(service function), we must add only one backup node and add some edges into graph of VN to construct SVN, meantime the simplified problem is of equivalance to problem that one-node fault-tolerant graph problem which is NP.

when inserted node sets is previous given, the problem k-SNFT($G(V, E, S), B(V, S)$) is subproblem of k-SNFT($G(V, E, S)$).

Suppose topological structure of graph G is path P, k-SNFT($P(V, E, S), B(V, S)$) is NP, the proof is below, exist one node n(n$\in V$) and we add backup nodes and edge to protect the node n, the former graph P firstly is changed into $P + n$, nevertheless topological structure of $P + n$ may is not path, or be tree and non-tree graph. then protect other nodes of V.

edge fault tolerant problem is subgraph of node fault tolerant problem

E. why happen Survivable embeded Virtual Network Request

The software-defined NFV architecture further offers agile traffic steering and joint optimization of network functions and resources In this section, we define the SeVN design problem as follows: for a given VN request with N task nodes, the VN is already embedded in substrate network SN and running service $s_i(s_i \in S)$, protect the VN with one additional node and a set of appropriate links to connect these nodes, and reserve sufficient computing and communication resources in these nodes and

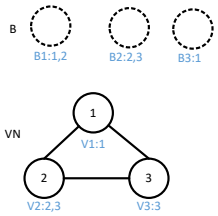

Fig. 1. SVNR

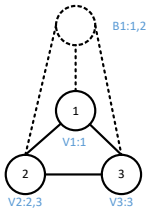

Fig. 2. optimal SVNR

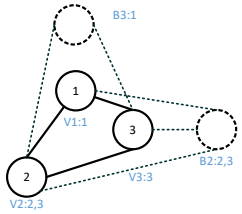

Fig. 3. non-optimal SVNR

links to guarantee the restorability of VN request after a facility node failure.

In any real virtual networks, any nodes of virtual network are deployed with a service function and the corresponding physical node could run some various service functions. After virtual network VN is embedded into substrate network, a running node of physical network could fail multiple virtual network so that we should obtain survivable eVN to be embedded into substrate network so that recovery from the failure even though one or multiple nodes of virtual network failed. The added nodes of survivable embedded virtual network are embedded into substrate network ultimately, the added nodes have specific service functions similarly.

F. VN Reqeust Model

A task graph for a VN request is an undirected attributed graph, denoted as $G^V(V^V, E^V, S^V)$, where V^V corresponds to a set of task nodes, E^V denotes a set of bidirectional edges among the VN nodes and S^V denotes a function service among the VN nodes. Each task node v_i^V specifies the needed computing resource b_i^V, and each edge specifies the requested amount of communication (bandwidth) resource b_{st}^V.

G. Survivable embedded Virtual Network Request

There are different combinations of service function among all nodes of virtual network, and there is only one type of service function running on the corresponding virtual node at that moment.

There exist many backup virtual nodes $B(V, S)$ abstracted from un-occupied by VN's corresponding node for guaranteing maintaining former topological structure of virtual network request $G(V, E, S)$ once one fault node v appeared in virtual network request $G(V, E, S)$. Solving the survivable request of embedded virtual network is of equivalence with asking 1-SNFT(G,B) of graph G and backup nodes set B.

An example of a typical survivable request of embedded virtual network is show in Fig.1.

Generally speaking, survivable request of virtual network is 1-specific node fault-tolerant of virtual network $G(V, E, S)$ with backup nodes set $B(V, S)$. Inserting any edges among nodes $V \cup B$ to construct G^o as shown in Fig.2,3, for which G is a subgraph of $G^o - v(\forall v \in V \cup B^o, B^o \subset B)$.

H. reduce immigration when node fail

However, this poses a limitation since the result only guarantees graph isomorphism and not equality. In other words, there may be a need to physically swap remaining VMs while recovering from some failure in order to return to the original infrastructure G. Recovery may then be delayed, or require more resources for such swapping operations.

I. EeVSN Design Formulation

With the discussion above, for a given N-nodes VN, EVN is designed within an Edit Grid with N+1 nodes through properly selecting the necessary links between these nodes and dimensioning the resources requirements associated with these nodes and links. Our design objective is to minimize the total amount of such resources while still guaranteeing that if a node fails (that is, the node and its adjacent links are removed in the Edit Grid), we can still assign each node/link in VN to a node/link in the EVN, that has sufficient CP.CM. resources respectively.

Furthermore, there are two different cases. If re-embedding or migrating the unaffected task node is allowed for failure recovering, it is known as the FD-EVN design problem.Otherwise, after each failure, if the failed node is restored in the only one backup node without migrating other unaffected node, it is referred to as the FI-EVN design problem. Generally speaking, designing FD-EVN is a combinatorial optimization problem and needs to be investigated in depth, while FI-EVN exists exclusively and could be figured out easily (meaning not very clear).

In conclusion, the formulation indicates that this problem is a BQP problem and the solution of this problem is a series of permutation matrix which determine the migration reassigning approach after each node failure. Thus, its computation complexity is (N)N , whereN is the computation complexity for one node failure.

J. Survivable embedded Virtual Network Request

We would link some edges amongst nodes $V \cup B$ to construct 1-SNFT graph of arbitrary graph G as show Fig.2,3, when only one node of VN failed, the former graph of VN is also subgraph isomorphism of graph of SVN. For example, when node v3 failed the node v2 run service 3 and backup node run service 2

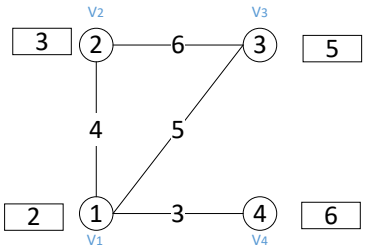

Fig. 4. VSN

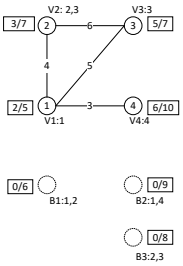

Fig. 5. eVSN

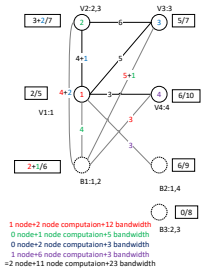

Fig. 6. FD

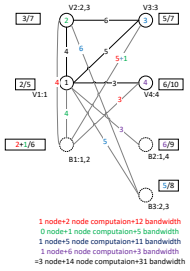

Fig. 7. FI

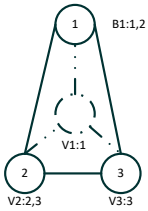

Fig. 8. node v3 failed in optimal SVNR

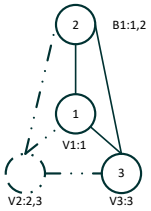

Fig. 9. node v3 failed in optimal SVNR

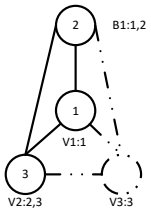

Fig. 10. node v3 failed in optimal SVNR

so that the graph of VNR is subgraph isomorphism of SVN-v3 as shown in figure 8,9,10,the red edge and red node is disable.

It is our objection that firstly minimize number of added back-up nodes, then minimize number of added edges to construct specific node fault tolerant graph. The graph of SVN, which is 1-NFT with node service function with respect to the graph of VN.

III. Requesting Survival Virtual Network Using Integer Linear Programming

The one of contributions of this paper is the modelization of specific node fault tolerant problem as an optimization. The general technique used to solve this problem is a combined algorithm consisting of enumeration and Ullmann Algorithm [13], which is proposed as a algorithm of exponential complexity which is brute force method briefly. More precisely, we propose an Integer Program [14] model of this problem. In the following, We briefly explain how to solve a problem using Integer Program.

A. Modeling the SeVN problem

In this section, we formulate the problems discussed in the previous sections with Integer Program method. Though we have proved that these problems are NP-complete, as we will demonstrate later, the IP formulation of the problem provides a very viable tool for solving it, for most real-world virtual network request which typically have less than a few hundred nodes and edges. Throughout this section, we will focus exclusively on the survival embedded virtual network request $[G(V, E, S), B(V, S))]$, we assume that G is simple undirected service label graph.

To begin with, we introduce some necessary notation:

C^o: $[c_u]_{|V|+|B|}$, node computation matrix of the SNFT graph G^o. where

$$c_u^o = \begin{cases} x \ if \ node \ u \ have \ nodecomputation \ x \\ 0 \ otherwise \end{cases}.$$

C: $[c_u]_{|V|}$, node computation matrix of the arbitrary graph G. where

$c_u = \begin{cases} x\ if\ node\ u\ have\ nodecomputation\ x \\ 0\ otherwise \end{cases}$.

B^o: $[b^o_{u,v}]_{(|V|+|B|)\times(|V|+|B|)}$, the edge's bandwith matrix of graph G^o. where

$b^o_{u,v} = \begin{cases} x\ if\ edge\ e\ originates\ from\ u\ to\ v\ and\ correspongdingbandwith\ scost \\ 0\ otherwise \end{cases}$

B: $[b_{u,v}]_{|V|\times|V|}$, the edge's bandwith matrix of arbitrary graph G. Where

$b_{u,v} = \begin{cases} 1\ if\ edge\ e\ originates\ from\ u\ to\ v\ and\ correspongdingbandwith's cost \\ 0\ otherwise \end{cases}$

T^l: $[t^l_{u,v}]_{|V|\times(|V|+|B|)}$, The l-th node permutation matrix against l-th node failed. Where

$t^l_{u,v} = \begin{cases} 1\ if\ node\ u\ transformed\ to\ node\ v \\ 0\ otherwise \end{cases}$.

S^o: $[s_{u,v}]_{(|V|+|B|)\times|S|}$, incidence matrix of services of the SNFT graph G^o. where

$s^o_{u,v} = \begin{cases} 1\ if\ node\ u\ have\ serivice\ v \\ 0\ otherwise \end{cases}$.

S: $[s_{u,v}]_{|V|\times|S|}$, incidence matrix of services of the arbitrary graph G. where

$s_{u,v} = \begin{cases} 1\ if\ node\ u\ have\ serivice\ v \\ 0\ otherwise \end{cases}$.

A:$[a_i]_{|B|}$, whether the i-th backup nodes is used or not. where

$A_i = \begin{cases} 1\ if\ node\ i\ is\ used \\ 0\ otherwise \end{cases}$.

The problem of requesting 1-service node fault tolerant graph $[G(V,E,S),B(V,S))]$ can be formulated as the following Integer Programming problem:

$$T^l * C^o \geq C \tag{1}$$

$$T^l * (T^l * B^o)' \geq B \tag{2}$$

$$B_{u,v} \leq B^o_{u,v} (1 \leq u \leq n, 1 \leq v \leq n) \tag{3}$$

$$\sum_{1\leq v\leq(n+b)} T^l_{i,v} = 1 (1 \leq i \leq n) \tag{4}$$

$$\sum_{1\leq u\leq n} T^l_{u,j} \leq 1 (1 \leq j \leq (n+b)) \tag{5}$$

$$\sum_{1\leq u\leq n, 1\leq v\leq(n+b)} T^l_{u,v} = n \tag{6}$$

$$T^k_{u,k} = 0 (1 \leq u \leq n) \tag{7}$$

$$T^l * S^o \geq S \tag{8}$$

$$T^l_{u,(n+i)} \leq A_i (1 \leq u \leq n, 1 \leq i \leq b) \tag{9}$$

$$\tag{10}$$

- Constraint(1) guarantees that computation resource of every nodes of augment graph G^o larger than computation resource of every nodes of former graph G when the l-th critical node of graph G^o failed, we could confirm graph G which is subgraph isomorphism of augment graph G^o. we represent the mapping relation as permutation matrix in view of Ullman [13]
- Constraint(2) bandwith resource of every edges of augment graph G^o larger than bandwith resource of every edges of former graph G when the l-th critical node of graph G^o failed.
- Constraint(3) guarantees that augment graph G^o must be the augmented graph of previous graph G.
- Constraint(4,5,6,7) implies that operation of the transform matrix of mapping relation is appropriate and correct format in respect with one node failure.
- Constraint(8) guarantees that service types of every nodes of augment graph G^o contain service types of every nodes of former graph G when the l-th critical node of graph G^o failed

B. Objective Function and Approximation

We seek to minimize the amount of resource used for a SeVN request. The object function of the adapted IP is then

$$\begin{aligned} min: \alpha \sum_{1\leq i\leq b} A_i + \\ \beta \times (\sum_{1\leq u\leq(n+b)} C^o_{u,v} - \sum_{1\leq u\leq n} C_u) + \\ \gamma \times (\sum_{1\leq u\leq(n+b), 1\leq v\leq(n+b)} B^o_{u,v} - \sum_{1\leq u\leq n, 1\leq v\leq n} B_{u,v}) \end{aligned} \tag{11}$$

where α, β, and γ are weight about add new node , node's computation and link's bandwith, respectively. To achieve minimize the amount of nodes, $\alpha \gg \beta, \alpha \gg \gamma$, respectively, $\alpha + \beta + \gamma = 1$. According to [6], [15] the relative cost of computing and bandwidth $\theta = 3$, which means $\beta/\gamma = \theta = 3$.

The presence of the boolean variables let the integer program of SeVN into a NP-Hard problem. An alterative is to relax the boolean variables to real-value variables between 0 and 1, obtain an approximate virtual node transformation matrix, then rounding decimal value into one or zero and judge the node transformation matrix is logically legal.

$$dp[i][W_1][W_2]\dots[W_m] = \left\{ \begin{array}{l} dp[i-1][W_1 - w_i][W_2]\dots[W_m] + v_i \\ dp[i-1][W][W_2 - w_i]\dots[W_m] + v_i \\ dp[i-1][W][W_2]\dots[W_m - w_i] + v_i \end{array} \right\} \tag{12}$$

$n * \prod_{i=1}^{m} C_i$

$m * n * \prod_{i=1}^{m} C_i$

IV. HeuristicAlgorithms

In this section, we propose two heuristic algorithms for FD-SeVN problem, as well as an SeVN problem's algorithm with resources sharing consideration fitting for both FD-SeVN and FI-SeVN.

The general idea of the heuristic for FD-SeVN design is to consider the failure of primary nodes in Edit Grid sequentially, and in each step, compute the minimum additional resources needed to reassign the task graph based on an incremental approach (i.e., recovering from the current node failure should take not only the survived primary nodes/links resources into consideration, but also the survived redundant resources reserved for previous node failure). After examining all the node failures, SeVN is constructed within the Edit Grid with the added redundant resources in each step.

However, as a matter of fact, computing the minimum additional resources needed to convert one attributed graph to another (hereafter called graph alignment problem, GAP) is an NP-complete problem which could be reduced from the Graph Edit Distance problem [26]. Therefore, we relax some

constraints and propose two heuristic algorithms. In detail, we first decompose a graph to a multiset of star structures which retains certain structural information of the original graph. Then, the graph alignment cost could be approximated by the matching cost between two graphs based on their star representations.

This approach is elaborated as follows.

A. FD-SeVN Algorithm

1) Graph Decomposition

Graph Decomposition:Star Structure: A star structure s is an attributed, single-level, rooted tree which can be represented by a 4-tuple $s = (v, L^*, B^*, C^*, S^*)$, where r is the root node, B^* is the bandwidth of each link associated with the root node, C^* is the computation resource requirement of every nodes, L^* is labels of nodes included in this star structure, S^* is service type of every nodes . In this case, the node label is the task node mapped on it in the primary embedding. Edges exist between the root node and its adjacent nodes, and no edge exists among its adjacent nodes.

More exactly, for node v_n in an attributed graph G (V, E, S, L),we can generate a star structure s_n corresponding to v_n in the following way: $star_n = (v_n, L^n, B^n, C^n, S^n)$ where $B^n = \{B^n_{n,u} | \text{for all } e_{n,u} \in E\}$, $C^n = \{C^n_u | \text{for all } e_{n,u} \in E\}$ and $L^n = \{l(v_u) | \text{for all nodes adjacent with } n\}$. Accordingly, we can derive N star structures for a graph with N nodes. In this way, a graph can be transformed to a multiset of star structure.

Due to the particularity of star structure, the alignment cost between two star structures can be computed easily as below. For two star structures $star_x$ and $star_y$, the alignment cost of $star_x$ to $star_y$ is $\lambda(star_x, star_y) = \sum_{l(v_u) \in L^x \cap L^y} [\beta |B^y_{y,u} - B^x_{x,u}|_0 + \gamma |C^y_u - C^x_u|_0] + \sum_{l(v_u) \in L^x - L^y} [\beta B^y_{y,u} + \alpha C^y_u]$

V. Evaluation

In the proposed two step for $EVSNR$, we employ a proactive failure dependent protection based EVN design with sufficient redundancy before embedding process in order to conquer the limitation of existing work, which consider the redundancy within the embedding process and failure independent protection based migration approach, in resource efficiency. Accordingly, the performance of optimal solution of EVN design and embedding formula, and heuristic algorithms are presented in this section in terms of average acceptance ratio, embedding cost, as well as the migration frequency after facility node failure. when a substrate node fail, which is placed with two virtual node.

A. Simulation Settings

For any VN request, the number of VN nodes is randomly determined by a uniform distribution between 4 and 10 and each pair of virtual nodes is randomly connected with probability 0.5. The computing requirements on VN nodes follow a uniform distribution from 5 to 10, as well as the bandwidth on VN links from 10 to 20. The arrivals of VN requests are modeled by a Poisson process (with mean of 10 requests per 100 time units).The duration of the requests follows an exponential distribution with 1000 time units on average. We assume the relative cost of computing and bandwidth is 3 [6], [15], which means $\theta = 3$. The SN topologies used are randomly generated with 40 nodes using the GT-ITM tool [16] The computing (bandwidth) resources of the substrate nodes (links) are real numbers uniformly distributed between 30 and 50 (100 and 150).

B. Acceptance Ratio

In this section, the service acceptance ratio for VN with the proposed survivable approach is examined, as well as a typical integer liner program method. Also, the acceptance ratio of VN without survivability requirement is presented as a baseline to gauge the impact of additional amount of resources consumed for survivability on the service provisioning capability of SN.We employ the proposed $EVSNR$ as the embedding algorithm for all the EVN designing approach.

In Fig. 6, acceptance ratio drops quickly before 5000 time units because there are sufficient substrate resources for the arrived VN requests, and when the amount of running VN requests in system are stable (after 7500 time units), acceptance ratio would keep consistent. Fig. 6 also depicts that FDEVN: MinFir and FD-EVN:Rad lead to higher acceptance ratio than FI-EVN algorithm through efficient resources sharing. In particular, comparing with FI-EVN, FD-EVN:MinFir approach achieves almost 15long run. Intuitively, it is the result of fact that, for FI-EVN embedding, the backup node is associated with a lot of resources since it has to emulate every primary node after it fails, and so, embedding of backup node is vulnerable. However, the acceptance ratio suffers at least 8resources consumption for survivability purpose.

C. Embedding Cost

In this work, the embedding cost is calculated as the cost of the substrate resources (i.e. cost of CP. on all facility nodes and CM. on all fiber links) consumed to satisfy the EVN resource requirements. And, in this simulation, we assume there are sufficient resources in all the substrate components.

We further compare the embedding cost of different EVN design algorithms with (denoted as +Shared) or without (denoted as +NoShared) consideration of the substrate resources sharing in EVNE approach. Generally, with regular VN arriving and leaving, embedding cost.

D. Migration Frequency

As mentioned above, FD-EVN algorithms achieve higher acceptance ratio and lower embedding cost at the cost of more node migration after facility node failure, which would cause service interruption and should be examined carefully, especially for the application with SLA constraints. We run our simulation in 30000 time unites, which corresponds to about 2000 requests on average in each instance of simulation. The migration frequency after random facility node failure is presented in Fig. 8 in terms of the number of VN nodes.

The physical infrastructure consists of 40 compute nodes with capacity uniformly distributed between 50 and 100 units. These nodes are randomly connected with a probability of 0.4 occurring between any two nodes, and the bandwidth on each physical link is uniformly distributed between 50 and 100 units.

	$C_{P_{2.2}}$	$C_{P_{2.3}}$	C_{P_3}	C_{P_4}	$C_{B_{1.1}}$	$C_{B_{1.2}}$	$C_{B_{2.1}}$	$C_{B_{2.4}}$	$C_{B_{3.2}}$	$C_{B_{3.2}}$
R_{V_1}	∞	∞	∞	∞	N+(2)+4+5+3	∞	N+(2)+4+5+3	∞	∞	∞
R_{V_2}	4	∞	∞	∞	∞	N+(3)+4+6	∞	∞	N+(3)+4+6	∞
R_{V_3}	∞	M+(2)+5	5	∞	∞	∞	∞	∞	∞	N+(5)+5+6
R_{V_4}	∞	∞	∞	3	∞	∞	∞	N+(6)+3	∞	∞

	C_{P_1}	C_{P_3}	C_{P_4}	$C_{B_{1.1}}$	$C_{B_{1.2}}$	$C_{B_{2.1}}$	$C_{B_{2.4}}$	$C_{B_{3.2}}$	$C_{B_{3.2}}$
R_{V_1}	4	∞	∞	M+4	∞	N+(2)+4+5+3	∞	∞	∞
R_{V_2}	∞	∞	∞	∞	M+(1)+4+1	∞	∞	N+(3)+4+6	∞
R_{V_3}	∞	6	∞	∞	∞	∞	∞	∞	N+(5)+5+6
R_{V_4}	∞	∞	0	∞	∞	∞	N+(6)+3	∞	∞

	C_{P_1}	$C_{P_{2.2}}$	$C_{P_{2.3}}$	C_{P_4}	$C_{B_{1.1}}$	$C_{B_{1.2}}$	$C_{B_{2.1}}$	$C_{B_{2.4}}$	$C_{B_{3.2}}$	$C_{B_{3.3}}$
R_{V_1}	5	∞	∞	∞	$M+5$	∞	N+(2)+4+5+3	∞	∞	∞
R_{V_2}	∞	6	∞	∞	∞	M+6	∞	∞	N+(3)+4+6	∞
R_{V_3}	∞	∞	M+(2)+1	∞	∞	∞	∞	∞	∞	N+(5)+5+6
R_{V_4}	∞	∞	∞	0	∞	∞	∞	N+(6)+3	∞	∞

Accepted VNR Ratio	Ratio of virtual networks that were successfully embedded into the substrate topology.
Stress	Average number of virtual links/nodes that have been assigned to the substrate links/nodes.
Utilization	Bandwidth/CPU utilization of substrate links/nodes.
Path Length	Length of communication paths assigned to virtual links for virtual network embedding.
Cost, Revenue, and Cost/Revenue	Cost: Sum of CPU and bandwidth resources being used for the embedding. Revenue: Sum of CPU and bandwidth demands realized.
Active Nodes	Number of nodes that need to be active in order to host all the virtual networks. This metric is especially useful in the context of e
Power Consumption	Power consumption of all Active nodes. Several powerconsumption models are implemented.
Runtime	Average runtime of the algorithm.
Initialization Overhead	Some algorithms come with initialization cost (in terms of runtime, e.g., the distributed algorithm presented in initially partitions th
Hidden Hops Ratio	Ratio of hidden hops, e.g., the number of nodes only needed for forwarding packages between other nodes. Especially useful in the
Communication Overhead	Communication overhead of distributed algorithms (i.e., number of messages sent between substrate nodes).

VInf requests arrive randomly over a timespan of 800 time slots and the inter-arrival time is assumed to follow the Geometric distribution at a rate of 0.75 per time slot. The resource lease times of each VInf follows the Geometric distribution as well at a rate of 0.01 per time slot. A high request rate and long lease times ensures that the physical infrastructure is operating at high utilization. Each VInf consists of nodes between 2 to 10, with a compute capacity demand of 5 to 20 per node. Up to 90% of these nodes are critical and all failures are independent with probability 0.01. Connectivity between any two nodes in the VInf is random with probability 0.4, and the minimum bandwidth on any virtual link is 10 units. There are two main sets of results: (i) scaling the maximum bandwidth of a virtual link from 20 to 40 units while reliability guarantee of every VInf is 99.99%, and (ii) scaling the reliability guarantee of each VInf from 99.5% to 99.995% while the maximum bandwidth of a virtual link is 30 units.

Accept ratio mapping cost migration cost cpu time per Vinf

REFERENCES

[1] J. Xu, J. Tang, K. Kwiat, W. Zhang, and G. Xue, "Survivable virtual infrastructure mapping in virtualized data centers," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pp. 196–203, IEEE, 2012.

[2] M. R. Rahman, I. Aib, and R. Boutaba, "Survivable virtual network embedding.," in *Networking*, vol. 6091, pp. 40–52, Springer, 2010.

[3] M. R. Rahman and R. Boutaba, "Svne: Survivable virtual network embedding algorithms for network virtualization," *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 105–118, 2013.

[4] T. Guo, N. Wang, K. Moessner, and R. Tafazolli, "Shared backup network provision for virtual network embedding," in *Communications (ICC), 2011 IEEE International Conference on*, pp. 1–5, IEEE, 2011.

[5] Y. Chen, J. Li, T. Wo, C. Hu, and W. Liu, "Resilient virtual network service provision in network virtualization environments," in *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*, pp. 51–58, IEEE, 2010.

[6] H. Yu, C. Qiao, V. Anand, X. Liu, H. Di, and G. Sun, "Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pp. 1–6, IEEE, 2010.

[7] W.-L. Yeow, C. Westphal, and U. C. Kozat, "Designing and embedding reliable virtual infrastructures," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 57–64, 2011.

[8] T. C. Bressoud and F. B. Schneider, "Hypervisor-based fault tolerance," *ACM Transactions on Computer Systems (TOCS)*, vol. 14, no. 1, pp. 80–107, 1996.

[9] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield, "Remus: High availability via asynchronous virtual machine replication," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pp. 161–174, San Francisco, 2008.

[10] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pp. 273–286, USENIX Association, 2005.

[11] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual routers on the move: live router migration as a network-management primitive," in *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 231–242, ACM, 2008.

[12] F. Harary and J. P. Hayes, "Node fault tolerance in graphs," *Networks*, vol. 27, no. 1, pp. 19–23, 1996.

[13] J. R. Ullmann, "An algorithm for subgraph isomorphism," *Journal of the ACM (JACM)*, vol. 23, no. 1, pp. 31–42, 1976.

[14] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[15] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, *et al.*, "Above the clouds: A berkeley view of cloud computing," tech. rep., Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

[16] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, vol. 2, pp. 594–602, IEEE, 1996.