Topology-Aware Remapping to Survive Virtual Networks against Substrate Node Failures

Ailing Xiao, Ying Wang, Luoming Meng, Xuesong Qiu, Wenjing Li
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
xiao_ailing@bupt.edu.cn, wangy@bupt.edu.cn, lmmeng@bupt.edu.cn

***Abstract*—Network virtualization has garnered great attention as a promising way to reform the current Internet. Multiple virtual networks (VNs) can be embedded onto the same substrate network through network virtualization. Recently the survivability of the VNs in the event of substrate failures has been attracting a broad attention. A good way to provide survivability is to use a hybrid strategy, which proactively reserves a backup quota on each substrate resource and reactively triggers a fast remapping mechanism using the backup quotas when a substrate failure happens. However, a survivable hybrid strategy for substrate node failures has not been explicitly studied by the existing research. In this paper, we investigate the survivability problem in a network virtualization environment and provide a topology-aware remapping policy to deal with substrate node failures. An integer linear programming (ILP) model is presented to optimize the fast node remapping with consideration of the business profit and the topological attributes. Simulations show that the proposed policy can effectively recover VNs from substrate node failures and increase acceptance ratio and business profit in the presence of substrate node failures.**

Keywords—Network virtualization; Virtual network remapping; Network survivability; Topology awareness; Failure recovery

I. Introduction

Network virtualization has garnered great attention as a promising way to overcome the impasse of the current Internet [1-3]. It decouples the infrastructures from the services in the traditional Internet Service Providers (ISPs) to make it possible to deploy diverse technologies and decrease power consumptions on the substrate infrastructures at the same time. In a network virtualization environment (NVE), an infrastructure provider (InP) owns and maintains substrate networks, while a service provider (SP) assembles virtual networks by aggregating resources leased from one or multiple InPs so as to provide end-to-end services to the users [1-3].

One of the biggest problems in a NVE is the virtual network embedding problem, which calls for the cooperation between the relevant InPs and SPs to map VNs onto substrate networks. The problem is proved to be NP-hard [4] and many heuristic algorithms [12-15] have been proposed to solve it. However, none of them take into account the survivability of VNs in the event of substrate node/link failures, which may affect all the VNs using that node/link and cause serious losses to both the SPs the InPs.

With that in mind, until recently, there are a number of studies focusing on the survivability of VNs. Some of them use a proactive strategy [5-9] to survive VNs against possible substrate failures (e.g. single link failure, single regional failure and single facility node failure), which proactively allocates enough resources for not only primary use but also backup use when accepting a VN request. Although it can ensure substrate failure recovery, the proactive strategy has an obvious drawback of resource wastage. The others try to overcome the limitation through a hybrid strategy [10]. In order to provide the affected VNs a best-effort recovery when a substrate link failure happens, the hybrid strategy reactively triggers a fast rerouting mechanism using the backup quota proactively reserved on the detours in the pre-computed detour set of the failed substrate link. However, the protection against substrate node failures has not been explicitly studied. Due to the large amounts of hardware and complex software that may be involved on a substrate node in a network virtualization environment, the likelihood of substrate node failures increases. Therefore an effective node failure recovery approach is also very important [6-9]. The authors of [10] intend to combine the approach provided in [11] to solve the single node failure problem, but actually it only provides a node migration approach and what we need is a clear and complete survivable node remapping mechanism.

In this paper, we investigate the survivability of VNs against substrate node failures based on the hybrid strategy. The effectiveness and the efficiency of the hybrid strategy are ensured by making good use of the substrate topology. The contributions are as follows: (i) a topology-aware remapping policy is provided to survive VNs against substrate node failures; (ii) to support survivability, we present a method to compute backup candidate set and define node similarity based on substrate network topology; (iii) an ILP model is formulated to solve the fast remapping heuristic, which is location-constrained and uses the business profit and node similarity as two decision factors.

The rest of this paper is organized as follows. Section II reviews the related work on survivability of VNs. Section III

This work was supported by the National High Technology Research and Development Program of China (2013AA013502), the Funds for Creative Research Groups of China (61121061) and the Fundamental Research Funds for the Central Universities BUPT 2013RC1103.

formalizes the network model and describes the topology-aware remapping problem. Section IV introduces the definition of candidate node set and node similarity, and formulates the fast node remapping as an ILP. The simulation results that evaluate our proposed remapping policy are presented in Section V. Section VI concludes this paper.

II. Related Work

In general, the strategies used in the existing studies on the survivability of VNs can be classified into two categories: the proactive strategy and the hybrid strategy.

The proactive strategy embeds a VN request with the condition that the substrate can offer enough resources for both primary and backup needs, thus to provide VNs the survivability from possible failures. For example, the authors of [5] provide a profit driven method to achieve survivability from single substrate link failures. It decomposes the residual bandwidth on each substrate link into primary and backup parts, and accepts a VN request only if there are enough residual primary and backup bandwidth for primary and backup flows needed for embedding and protecting that VN request, respectively. In the event of a single regional failure, the study of [6] remaps the affected virtual networks within that failed region to the pre-allocated backup substrate resources in other normal regions. In [7], the same authors propose a two-step method to protect critical virtual nodes against single facility node failures. Firstly, to keep the protection process transparent to the InP, a reliable VN graph is designed by adding redundant virtual nodes and relevant virtual links to the initial VN; secondly, the reliable VN graph is mapped to the substrate network with a maximum possible link-sharing mechanism so as to minimize the cost. The study in [8] also uses a two-step approach to deal with single facility node failures, but it tries to minimize the cost by adding to the initial VN the minimum necessary virtual resources in the first step. In [9], the authors define and solve the LSVNE problem which imposes the location constraint of a virtual node to both the primary and backup substrate nodes for that virtual node to survive VNs against single facility node failures.

Although the proactive strategy can guarantee complete failure recovery, it has an obvious drawback of resource wastage and often associates with complex mixed integer linear programs in order to offer substrate resource sharing. To overcome these limitations, the authors in [10] propose a hybrid strategy to provide survivability against single substrate link failures. It proactively dedicates a fixed percentage of residual bandwidth on each substrate link for backup flows, and accepts a VN request if only the available primary bandwidth capacity can afford the primary bandwidth need of the VN request. In the event of a single substrate link failure, it reactively triggers a best-effort recovery mechanism which fast reroutes the affected virtual links to the detours in the pre-computed detour set of the failed substrate link.

In this paper, we provide a topology-aware remapping policy to recover the VNs from substrate node failures. An ILP optimization solution is used to optimize the fast remapping of the affected virtual nodes. We only deal with the single substrate node failure, thus to focus on the remapping of the virtual nodes on one failed substrate node at a time.

III. Network Model and Problem Statement

A. Substrate Network

We model the substrate network as an undirected weighted graph $G^S(N^S, E^S)$, where N^S is the set of substrate nodes and E^S is the set of substrate links. Each substrate node $x \in N^S$ has an associated available CPU capacity $cpu(x)$ and a geographical location *loc(x)*. A substrate link $s \in E^S$ has an available bandwidth capacity $bw(s)$. In order to survive a substrate node failure, a backup quota is reserved on each substrate resource. For a substrate node with available CPU capacity $cpu(x)$, $a(x)cpu(x)$ is reserved for primary use denoted as $R_a(x)$, and the remaining $b(x)$ percent is dedicated for backup use referred to as $R_b(x)$, where $a(x) + b(x) = 1$. Similarly, the available bandwidth capacity of a substrate link is decomposed into available primary bandwidth and available backup bandwidth with percentage of $a(s)$ and $b(s)$.

B. VN Request

A virtual network request $G^V(N^V, E^V)$ is also modeled as an undirected weighted graph. A virtual node $y \in N^V$ has a CPU capacity requirement $cpu(y)$ and a preferred location requirement $loc(y)$. A virtual link $v \in E^V$ is characterized by a bandwidth capacity requirement $bw(v)$. The entire VN request is guaranteed by a service level agreement (SLA) signed between the SP and the InP for uninterrupted service throughout its lifetime. A VN gv is labeled with revenue $Re(gv)$ and penalty $S(gv)$ to express the revenue of accepting gv and the penalty incurred for service interruption to gv, respectively. $Re(gv)$ depends on the lifetime of gv and the amount of substrate resources required by gv. $S(gv)$ has a linear relationship with $Re(gv)$ and should be paid by the InP if any virtual resource of gv is affected by a substrate node failure. The business profit to the InP equals the revenues earned from accepting VNs minus the penalties incurred for substrate node failures.

The location constraint $loc(i)$ of a virtual node i describes the possible embedding scope of i. Usually, it can be denoted as a triple $loc(i) = (x_i, y_i, d_i)$, in which the coordinate (x_i, y_i) is the center of a circle in G^S and d_i represents the maximum geographical distance allowed from (x_i, y_i). We model the location constraint as a mapping $LC(i): loc(i) \rightarrow \{loc(x)\}$, the result of which forms a location-constrained set including all the substrate nodes that satisfy the location constraint.

C. Problem Statement

Given: $G^S(N^S, E^S)$, $G^V(N^V, E^V)$, a SLA between the SP and the InP.

Question: how to map a VN request upon its arrival with consideration only about the primary resources it needs but still be able to fast recover the affected virtual nodes from possible substrate node failures in the future, such that the total penalty incurred for substrate node failures to the InP is minimized and the long term average business profit to the InP is maximized?

Our survivable remapping mechanism consists of four phases: initial candidate node and candidate path set computing, incoming VN request embedding, virtual node check-pointing and affected virtual resource remapping.

In the first phase, before any VN request arrives, the InP pro-actively computes a set of backup candidate nodes for each substrate node using a node selection algorithm in Algorithm 1 and a set of candidate paths for each pair of substrate nodes using a path selection algorithm adapted from the k-shortest path algorithm.

When a VN request arrives, the second phase of our policy is invoked. Without consideration of possible failures, the InP simply embeds the VN request to the available primary part of each substrate resource using the existing heuristic in [15].

In order to avoid the data loss of a virtual node when it is interrupted by a substrate node failure, the InP needs to check-point the virtual node by periodically sending its snapshot to some spare substrate node such that the virtual node could be restored from a saved state.

Finally, when a substrate node failure happens, an optimal fast remapping is reactively triggered. Firstly, an optimal node remapping scheme is computed for the affected virtual nodes, which can remap them to the appropriate candidate nodes through a best-effort manner. Secondly, an optimal link rerouting scheme is computed to reroute the relevant virtual links along the available candidate paths. Both the candidate nodes and paths are selected in the first phase. Lastly, with the check-pointing results in the third phase, the affected virtual nodes are migrated and the relevant flows are rerouted according to the schemes.

IV. ILP FOURMULATION FOR CANDIDATE NODE OPTIMIZATION

In this section, we provide a method to compute a backup candidate node set for each substrate node and define the similarity of two substrate nodes considering only the topological attributes of the substrate network. Then an ILP formulation is presented to optimize the fast node remapping problem.

A. Candidate Node Set

When a substrate node fails to work, all the virtual nodes mapped on it have to be remapped to other normal substrate nodes, and all the virtual links whose carrying substrate paths flow through that failed substrate node need to be rerouted. Therefore, a candidate backup node should be able to reach all the substrate nodes that have direct or indirect connections with the failed one within a certain geographical area. Now, we introduce a method based on the substrate topology to compute the candidate node set for each substrate node.

First, we define the function $f(G^S, x, h)$, which returns a set of substrate nodes that can be reached through exactly h hop(s) from a substrate node x. It means that the shortest paths between the substrate nodes and x consists of exactly h hop(s) in G^S. Here, the hop-count is used as a service quality measure since each hop will bring in related transmission and processing delay. Formally, the function can be denoted as:

$$f(G^S, x, h) = \{n \mid MinimumHop(x, n) = h, h \geq 0\} \qquad (1)$$

An example of a substrate network G^S is shown in Fig. 1. For a substrate node A, $f(G^S, A, 0) = \{A\}, f(G^S, A, 1) = \{B, C, E\}$.

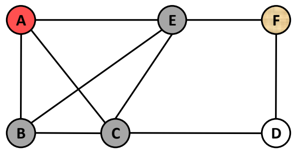

Fig. 1. Example of a substrate network

Second is the function of $Cov(G^S, x, h)$, it returns a set of substrate nodes that can be covered within h hop(s) from a substrate node x, which means that the shortest paths between the substrate nodes and x consists of no more than h hop(s) in G^S. We call it the cover set and denote it as:

$$Cov(G^S, x, h) = \{\, n \mid MinimumHop(x, n) \leq h, h \geq 0\} \qquad (2)$$

Hence, it's natural to have the formula of $Cov(G^S, x, h) = \bigcup_{i=0}^{h} f(G^S, x, i)$. The cover set of A within one hop is $Cov(G^S, A, 1) = \bigcup_{i=0}^{1} f(G^S, A, i) = \{A, B, C, E\}$.

If $r \in G^S$ is a candidate node of a substrate node x with a certain hop constraint h, all the substrate nodes, which can be reached within h hops from x, should be covered by r within the same hop-count. In other words, the cover set of r must include all the nodes in the cover set of x, except x itself. It can be expressed by the following formula:

$$Cov(G^S/x, r, h) \supseteq \left(Cov(G^S, x, h) - f(G^S, x, 0)\right) \qquad (3)$$

The notation G^S/x represents the new substrate network graph generated by removing x from G^S. All nodes satisfying condition (3) compose the candidate set of x with a hop constraint h. We denote it as $C(x, h)$, where h is controlled by the InP with consideration of the connectivity of G^S (e.g., if the substrate network is in poor connectivity, the InP should assign a higher h value to ensure each substrate node with at least one backup candidate node) and the long term business profit (e.g., if the use of substrate links caused by some candidate nodes is not cost-efficient, the InP may choose a smaller h). In Fig.1, the candidate set of A with the constraint of one hop is $C(A, 1) = \{B, C, E\}$. The candidate node selection algorithm used in the first phase to compute $C(x, h)$ for each substrate node x is shown in Algorithm 1. We start searching the substrate network G^S from x using the width first search (WFS) algorithm. The variable "i" stands for the layer of substrate nodes that are being searched. The variable "num" represents the number of nodes in the i_{th} layer that can be put into $C(x, h)$. If num>0, the algorithm continues to search for the next layer; else the algorithm is terminated.

B. Node Similarity

If the candidate set of a substrate node has more than one element, we want to use them in a particular order with the purpose that we can save resources for more VNs to be recovered from possible substrate node failures in the future.

The Jaccard similarity coefficient [16] is extensively used for comparing the similarity of two sample sets, which is defined as the size of the intersection divided by the size of the union of two sets A and B:

$$J(A, B) = \frac{A \cap B}{A \cup B} \qquad (4)$$

Algorithm 1 Candidate Node Selection Algorithm

```
INPUT: $G^S(N^S,E^S)$, $x \in G^S$, $h$ (specified hop-count)
OUTPUT: $C(x,h)$
BEGIN
1 Set $C(x,h)$={}
2 var $i = 1$
3 WHILE true
4     Set $S = f(G^S,x,i)$
5     var $num = 0$
6     FOR each $r \in S$
7         IF $Cov(G^S / x,r,h) \supseteq Cov(G^S,x,h) - f(G^S,x,0)$
8             $C(x,h)$.add($r$)
9             $num$++
10        END IF
11    END FOR
12    IF $num$ >0
13        $i$ ++
14    ELSE
15        break
16    END IF
17 END WHILE
18 RETURN $C(x,h)$
END
```

Inspired by the Jaccard similarity coefficient, we define the similarity of two substrate nodes and use it as the sort criteria. The similarity value of two substrate nodes with a location constraint of h hops is formulated as the sum of h Jaccard similarity coefficient values:

$$Sim(x,r,h) = \sum_{i=1}^{h} J\left(f(G^S,x,i), f(G^S/x,r,i)\right), r \in C(x,h) \qquad (5)$$

Since the h Jaccard similarity coefficient values are all about the similarity of x and r compared in their connectivity with the other nodes in G^S, the value of $Sim(x,r,h)$ actually represents their degree of similarity in respect of connectivity.

Given two candidate nodes r' and r'' in $C(x,h)$. If r' has a higher value of $Sim(x,r',h)$, it means the connectivity of r' is more close to x. While if r'' has a smaller similarity value, it means r'' has better connectivity than x. It can be inferred by transitivity that r'' has better connectivity than r', which means r'' may appear in the candidate set of more substrate nodes besides x compared with r'. It's possible that, in some extreme cases, r'' is the only element in $C(m,h)$ of another substrate node m. Therefore we should give priority to r' when recover the affected virtual nodes on x and leave r'' to the substrate nodes like m to deal with their possible failures. Videlicet, to guarantee all the possible failure nodes with at least one available candidate node in the global scope, candidate nodes with higher similarity values in a candidate set should be preferentially used.

As shown in Fig.1, for $C(A,1) = \{B,C,E\}$, the similarity values of A and its candidate nodes can be computed with formula (5). The results are as follow: $Sim(A,B,1) = 0.67$, $Sim(A,C,1) = 0.5$, $Sim(A,E,1) = 0.5$. Compared with B, C also has connection to D. Hence C has better connectivity than B. Moreover, C is the only one element in $C(F,1)$. Assuming that the InP uses $C(x,1)$ of each substrate node x for node failure recovery and C has already been used up to recover the affected virtual nodes on the failed node A, then the virtual nodes on substrate node F will not be survived if F fails to work subsequently. So the InP should give preference to B over C when dealing with the failure of A and try to leave as much CPU capacity on C as possible for F.

C. ILP Formulation for Fast Node Remapping

Now we provide the ILP formulation for the backup candidate node optimization solution ILP_CNO, which is used in the fourth phase to compute the remapping scheme for the affected virtual nodes.

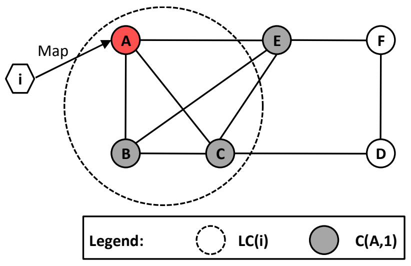

Fig. 2. Location constraint when remapping a virtual node i

When remapping the affected virtual nodes, their location constraints should still be considered. Only the nodes included in both the candidate set of the failed substrate node and the location-constrained set of an affected virtual node can be used to remap that affected virtual node. In Fig.2, $C(A,1) = \{B,C,E\}$ is used by the InP to recover from the failure of A. The substrate nodes that the virtual node i can be mapped to are restricted in a dotted circle by $LC(i) = \{A,B,C\}$. Given that i is originally mapped on A. When A fails to work, the location-constrained set of i is updated with $LC(i) = \{B,C\}$. The intersection of $C(A,1)$ and $LC(i)$ is $\{B,C\}$. Therefore the feasible substrate nodes that can be used to recover i from the failure of A are actually B and C, excluding E. The notations used in our formulations are described in Table I.

TABLE I. KEY NOTATIONS USED IN ILP_CNO

Notation	Description
ϕ_{vr}	Virtual node→substrate node indicator variable, ϕ_{vr}=1 if a virtual node v is mapped on a substrate node r
m_{yr}	Virtual node→substrate node indicator variable, m_{yr}=1 if an affected virtual node y will be remapped on a substrate node r
N_F^V	Set of affected virtual nodes, due to the failure of x
N_F^S	Set of feasible substrate nodes, $N_F^S = C(x,h) \cap LC(y)$, y is an affected virtual nodes due to the failure of x

Objective Function:

$$\min\{\alpha \times \sum_{y \in N_F^V} S(gv)\left(1 - \sum_{r \in N_F^S} m_{yr}\right) - \beta \times \sum_{r \in N_F^S}\sum_{y \in N_F^V} m_{yr} Sim(x,r,h)\} \qquad (6)$$

Subject to:

Backup Capacity Constraint

$$\sum_{y \in N_F^V} cpu(y) \times m_{yr} \le R_b(r), \qquad \forall r \in C(x,h) \qquad (7)$$

Node Remapping Constraint

$$\sum_{r\in N_F^S} m_{yr} \le 1, \forall y \in N_F^V \qquad (8)$$

$$m_{yr} = 0, \quad y \in gv \cap N_F^V, r \in C(x,h), \exists v \in gv, \phi_{vr} = 1 \quad (9)$$

$$m_{yr} = 0, \quad y \in N_F^V, r \in C(x,h) - LC(y) \qquad (10)$$

D. Remarks:

- The objective function (6) has two parts. The first part is for minimizing the total penalty incurred due to the CPU violation of any virtual node mapped on the failed substrate node x .This is achieved by preferentially remapping the virtual nodes with higher penalties. Whereas the second part is concerned about maximizing the usage of the available candidate nodes with higher similarity values, which can leave substrate nodes with better connectivity to the possible failure nodes that really need them and reduce the penalty incurred in return. We use $\alpha > 0$ and $\beta > 0$ to tune the weight of penalty and node similarity. Usually, a bigger value is assigned to α to ensure that the key priority is given to minimizing the penalty for InP. Based on the result of the first part, the second part adjusts the remapping scheme.
- Constraint (7) is the backup CPU capacity constraint of a candidate node r, the total CPU capacity used for recovering the affected virtual nodes must be within the available backup CPU capacity of r.
- Constraints (8), (9) and (10) are the node remapping constraints. Constraint (8) guarantees the uniqueness of node mapping that no more than one candidate node is selected for each affected virtual node. The exclusiveness of node mapping is supported by constraint (9). It signifies that for an affected virtual node y and another normal virtual node v, which are in the same VN (say gv), they cannot be mapped on the same substrate node. Constraint (10) represents the location constraint. It forces an affected virtual node y to be remapped on a candidate node that satisfies its location constraint $loc(y)$ specified in the VN request.

In this paper, we focus on the backup candidate node optimization solution. The candidate path optimization solution, which can compute the rerouting of the relevant virtual links, can be realized subsequently by reference to the backup detour optimization solution provided in [10].

V. Performance Evaluation

A. Simulation Environment

A substrate network topology with 100 nodes and 370 links is generated via GT-ITM, in which all nodes are randomly distributed in 100 x 100 grids. Both the CPU capacities and the bandwidth capacities follow a uniform distribution between 50 and 100. The VN topologies are generated in the same way. The number of virtual nodes is uniformly distributed from 3 to 10 with an average connectivity of 50%. Both the CPU requirements and the bandwidth requirements follow a uniform distribution ranging in [0-10] and [0-50], respectively. We assume that both VN requests and single substrate node failures follow a Poisson process with arrival rates 8 and 6 per 100 time units. The lasting time of the VN requests follows an exponential distribution with an average of 100 time units. The hop-count h is set to be 3 with consideration of the connectivity of the generated substrate network. Sufficient instances are simulated and the results are reported below.

B. Simulation Results

Three metrics are used for evaluation purposes in our experiments: the VN acceptance ratio, the long term average penalty, and the long term average business profit. Here, two different penalty functions are used to compute the penalty of a virtual network gv. The first function is $S_1(gv) = Re(gv) + c_1$. In the second function, we have a higher penalty $S_2(gv) = c_2 \times Re(gv)$. "$c_1$" and "$c_2$"are constants bigger than one.

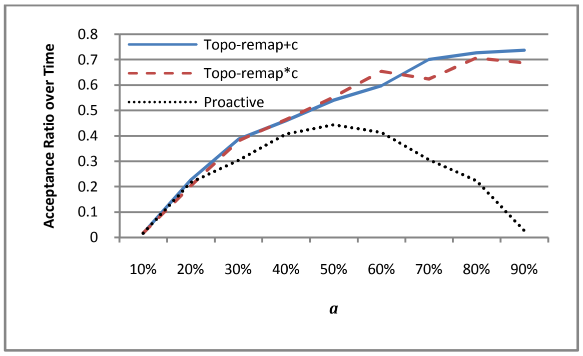

Fig. 3. Proactive vs. topo-aware in acceptance ratio

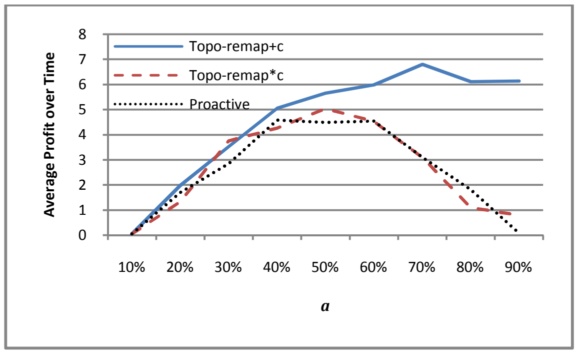

Fig. 4. Proactive vs. topo-aware in business profit

First, our proposed policy is compared with a baseline proactive policy in terms of acceptance ratio and business profit. The proactive approach embodies many of the key ideas in the previous works [5] [9], in which the residual capacity on each substrate resource is decomposed into primary and backup parts and the feasible backup nodes are also location-constrained. Fig.3 and 4 show the VN acceptance ratio and the long term business profit against the percentage a of substrate resource for primary use, respectively. The two different penalty functions mentioned above are used in the proposed policy. As a increases, more primary resources can be used for accepting new VNs, but fewer resources for survivability. Therefore, in Fig.3, the acceptance ratio of the topology-aware remapping policy gradually increases while the proactive approach reaches its highest acceptance ratio at $a = 50\%$. Due to the consideration of the location constraints, the acceptance ratio gets steady as time goes by. In Fig.4, the proposed policy outperforms the baseline approach greatly with the first penalty function, while the result is close to the baseline approach if using the second penalty function because of the huge compensation.

Second, the effectiveness of node similarity is evaluated in Fig. 5 and 6. Without the consideration of node similarity, the proposed policy incurs higher penalty and lower business profit.

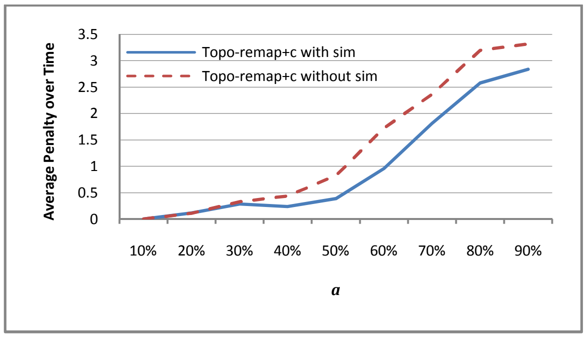

Fig. 5. Effectiveness of similarity in penalty

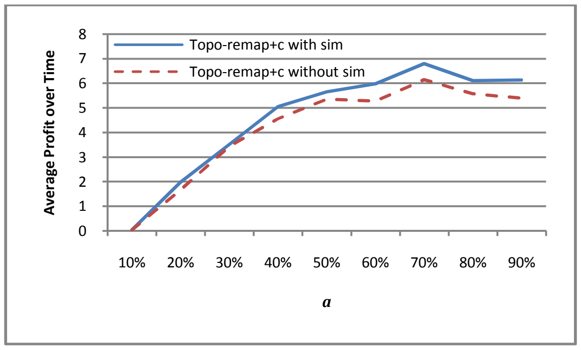

Fig. 6. Effectiveness of similarity in business profit

Lastly, the threshold of a with different penalty functions is discussed. We set $c_1 = 50$ and $c_2 = 3$ in our experiments. It can be seen in Fig.7 that $a = 70\%$ is the threshold point if the first penalty function is used in our topology-aware remapping policy, after which the penalty incurred for substrate node failures dominated the profit from new virtual network requests. And we can achieve the optimal profit at $a = 50\%$ using the second penalty function.

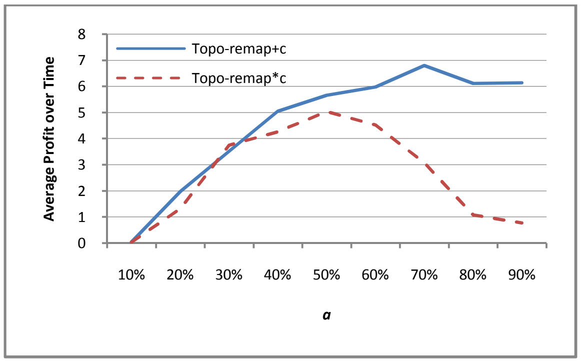

Fig. 7. Threshold of different penalty functions

VI. Conclusion

In this paper, we have investigated the survivability of virtual networks. A topology-aware remapping policy has been proposed to effectively recover VNs from substrate node failures. Particularly, we introduce the definitions of candidate node set and node similarity to support the proposed policy, the effectiveness of which has been verified through experiments. Simulations have also shown that the proposed policy outperforms the baseline algorithm in acceptance ratio and long term average business profit in the presence of single substrate node failures. In the future, we will implement and evaluate the overall remapping process more thoroughly in more dynamic environments with advanced economic models.

References

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp.34–41, Apr.2005.

[2] N. Feamster, L. Gao, and J. Rexford, "How to lease the Internet in your spare time," *2007 ACM SIGCOMM CCR*, vol.37, np.1, pp. 61–64, Jan. 2007.

[3] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.* , vol. 54, np. 5, pp. 862–876, Apr. 2010.

[4] D. Andersen, "Theoretical approaches to node assignment," unpublished.

[5] Muntasir Raihan Rahman, Issam Aib, and Raouf Boutaba, "Survivable virtual network embedding," *in Proc. IFIP Netw.2010*, pp. 40-52.

[6] Hongfang Yu, Chunming Qiao, and Vishal Anand,"Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures," *in Proc. IEEE GLOBECOM2010*, pp. 1-6, Dec. 2010.

[7] Hongfang Yu1, Vishal Anand, and Chunming Qiao, "Cost efficient design of survivable virtual infrastructure to recover from facility node failures," *in Proc. IEEE ICC* 2011, pp. 1-6, Jun. 2011.

[8] Chunming Qiao, Bingli Guo, and Shanguo Huang, "A novel two-step approach to surviving facility failures,"*in OFC 2011*, pp. 1-3, March 2011.

[9] Qian Hu, Yang Wang, and Xiaojun Cao, "Location-constrained survivable network virtualization," *in Proc. 35th IEEE SARNOFF 2012*, pp. 1-5, May. 2012.

[10] Muntasir Raihan Rahman, and Raouf Boutaba, "SVNE: Survivable Virtual Network Embedding Algorithms for Network Virtualization," *in Network and Service Management, IEEE Trans. On*, vol.58, np.1, pp. 1-14, Feb. 2013.

[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 *Proc. 2008 ACM SIGCOMM*, vol.38, issue. 4, pp. 231–242, Oct. 2008.

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

[13] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *2008 ACMSIGCOMM CCR*, vol. 28, no. 2, pp. 17–29, 2008.

[14] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. 2009 ACM SIGCOMM VISA*, pp. 81–88.

[15] N. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[16] V. V. Cross and T. A. Sudkamp, "Similarity and Compatibility in Fuzzy Set Theory: assessment and Application," New York: Physica-Verlag, 2002