Survivable Virtual Network Embedding

Muntasir Raihan Rahman[1], Issam Aib[1], and Raouf Boutaba[1,2]

[1] David R. Cheriton School of Computer Science,
University of Waterloo,
Waterloo, Ontario, Canada N2L 3G1
[2] Division of IT Convergence Engineering, POSTECH,
Pohang, Korea KB 790-784
{mr2rahman,iaib,rboutaba}@cs.uwaterloo.ca

Abstract. Network virtualization can offer more flexibility and better manageability for the future Internet by allowing multiple heterogeneous virtual networks (VN) to coexist on a shared infrastructure provider (InP) network. A major challenge in this respect is the VN embedding problem that deals with the efficient mapping of virtual resources on InP network resources. Previous research focused on heuristic algorithms for the VN embedding problem assuming that the InP network remains operational at all times. In this paper, we remove that assumption by formulating the survivable virtual network embedding (SVNE) problem and developing a hybrid policy heuristic to solve it. The policy is based on a fast re-routing strategy and utilizes a pre-reserved quota for backup on each physical link. Evaluation results show that our proposed heuristic for SVNE outperforms baseline heuristics in terms of long term business profit for the InP, acceptance ratio, bandwidth efficiency, and response time.

Keywords: Survivability, Virtual Network Embedding, Network Virtualization.

1 Introduction

Network virtualization has been proposed as a diversifying attribute of the future inter-networking paradigm that can enable seamless integration of new features to the current Internet resulting in rapid evolution of the Internet architecture [4, 8, 5]. By allowing multiple heterogeneous network architectures to cohabit on a shared physical infrastructure, network virtualization promises better flexibility, security, manageability and decreased power consumption for the Internet. In a network virtualization environment (NVE), the traditional role of the Internet Service Provider (ISP) has been divided into two separate entities: (1) the infrastructure providers (InP) who are responsible for deploying and maintaining physical network resources (routers, links etc.) and the (2) service providers (SP) who implement various network protocols and heterogeneous network architectures on virtual networks (VNs) composed from physical network resources leased from one or more infrastructure providers.

M. Crovella et al. (Eds.): NETWORKING 2010, LNCS 6091, pp. 40–52, 2010.

Virtual Network Embedding (VNE) is the central resource allocation problem in network virtualization. It deals with the efficient mapping of virtual networks onto physical network resources. More specifically, for each virtual network creation request, the VNE is responsible for mapping virtual nodes onto physical nodes and virtual edges onto one or more physical paths. The VNE problem, with constraints on virtual nodes and virtual links, can be reduced to the $\mathcal{NP}$-hard multi-way separator problem, even if the schedule of VN requests is known beforehand [3]. Even when all the virtual nodes are already mapped, the virtual link embedding problem remains $\mathcal{NP}$-hard. In order to reduce the hardness of the VN embedding problem and enable efficient heuristics, existing research has been restricting the problem space in different dimensions, e.g., considering the off-line version of the problem [13, 22], ignoring either node or link requirements [7,13], assuming infinite capacity of the substrate nodes and links to obviate admission control [7, 13, 22], and focusing on specific virtual topologies [13]. Recently the authors in [12, 6] have proposed VNE heuristics that combine the node and link embedding phases. The authors in [9] have proposed a distributed algorithm that simultaneously maps virtual nodes and virtual links without any centralized controller. However, a limitation of all these heuristics is that they assume the substrate network to be operational at all times, which is not realistic. The existing heuristics are not capable of handling substrate node and link failures, which may lead to poor performance and increased frustration for the SP.

In this paper, we formulate the survivable virtual network embedding (SVNE) problem to incorporate single substrate link failures in VNE and propose an efficient heuristic for solving it. To the best of our knowledge, this is the first work to consider survivability strategies in the network virtualization environment.

The rest of the paper is organized as follows. Section 2 provides an overview of the related work. Section 3 formalizes the network model and formulates the virtual network embedding and survivable virtual network embedding problems. In Section 4, we present our proposed hybrid policy heuristic as a solution to the survivable virtual network embedding problem. Section 5 presents simulation results that evaluate the proposed hybrid policy heuristic compared to base-line heuristics. Section 6 concludes by identifying future research directions.

2 Related Work

Node and link failure survivability problems have been investigated extensively for optical and multi-protocol label switched (MPLS) networks [16], and real time systems [21]. Our work differs in a number of aspects, due to unique challenges introduced by the network virtualization environment. First, the VNE problem is on-line in nature, whereas the survivable logical topology design problem in optical and mpls networks [17, 11, 10] is off-line. Secondly, in NVEs, we need to ensure that all virtual links are intact in the presence of failures. This restriction is not present, for example, in optical networks where the goal is to only ensure that all virtual nodes remain connected in the presence of failures, even if they

are not connected via a direct virtual link. Our contribution also differs from existing work in terms of the objective formulation. Our aim is to develop a survivable virtual network embedding solution that simultaneously maximizes the long term revenue for the InP and, at the same time minimizes the long term penalty incurred by the InP due to service violations caused by failures. This dual nature of the objective function in the presence of failures is absent both in the existing research on optical and mpls networking domains and the existing VNE heuristics. Another novel aspect of our work is that we utilize path-flow based optimization formulations for solving the SVNE problem. The path formulation allows control over the characteristics of the paths selected for embedding and survivability against failures. For instance, we can directly control the total number of paths, number of hops per path, and impose delay constraints on virtual links for QoS purposes. This is not possible with a link-flow based formulation which has been used for the previous VNE heuristics [6, 20, 12, 22].

3 Problem Formulation

3.1 Substrate Network

We model the substrate network as a weighted graph $G^S(N^S, E^S)$, where N^S and E^S represent the set of substrate nodes and links respectively. Each substrate node $x \in N^S$ has an associated cpu capacity $cpu(x)$ and a geographical location value $loc(x)$. A substrate link $s = (s_x, s_y) \in E^S$ between substrate nodes $s_x, s_y \in N^S$ has a bandwidth capacity $b(s)$. From now on, we denote the endpoints of any substrate link s as s_x and s_y.

3.2 Virtual Network Request

A Virtual Network (VN) request $G^V(N^V, E^V)$ is also modeled as a weighted graph. VN requests are associated with constraints and QoS requirements embodied into service level agreements (SLA) [2]. A virtual nodes $y \in N^V$ has a cpu capacity requirement $cpu(y)$ and geographical location requirement $loc(y)$. A virtual link $v \in E^V$ is characterized by a bandwidth capacity requirement $b(v)$ and a delay constraint $d(v)$. $d(v)$ is used to preselect the set of admissible simple substrate paths that can be used to embed v. An example of a typical substrate network and two virtual network topologies are shown in figure 1. The numerical values beside the substrate nodes and links represent cpu and bandwidth constraints of those nodes and links respectively.

3.3 Resource Usage Metrics

We assume that substrate network resources are finite. As a result, the amount of residual substrate network resources diminishes as new VN requests are processed. We keep track of the residual substrate node and link capacities in order

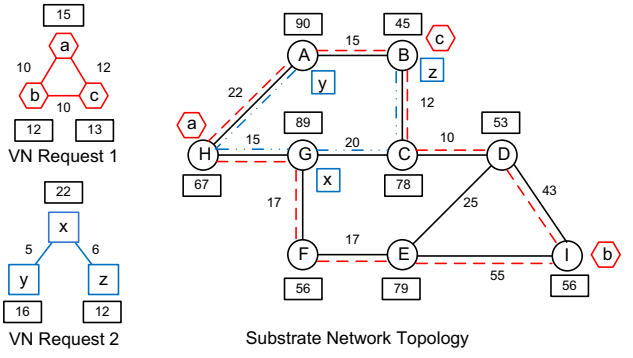

Fig. 1. Mapping of VN requests onto a shared substrate network

to make sure we don't accept a request unless there are adequate resources to serve it. The residual capacity of a substrate node $x \in N^S$ is defined as:

$$R_N(x) = cpu(x) - \sum_{y \in V(x)} cpu(y), \tag{1}$$

where $V(x)$ denotes the set of virtual nodes mapped onto x. Similarly the residual capacity of a substrate link $s \in E^S$ is defined as:

$$R_E(s) = b(s) - \sum_{\{v : \exists p \in \Gamma_E(v),\, s \in p\}} b(v), \tag{2}$$

where, $\Gamma_E(v)$ defines the set of paths in the InP that are used to embed the virtual link v (Section 3.4). The residual capacity values are updated after each new VN request has been successfully mapped on top of the substrate network as long as there remains adequate residual resources. The values are also updated after a VN departs and link failure arrivals and departures.

In order to protect against single substrate link failures, we dedicate a certain percentage of bandwidth resources on each substrate link for backup purposes. For a substrate link s with total bandwidth $b(s)$, $\alpha(s)b(s)$ bandwidth is reserved for primary flows, whereas $\beta(s)b(s)$ is reserved for backup flows, where $\alpha(s) + \beta(s) = 1$. The residual bandwidth measure is accordingly decomposed into primary and backup residual bandwidth measures $\mathcal{R}_\alpha(s)$ and $\mathcal{R}_\beta(s)$ respectively. As a result, we need to keep track of these two residual bandwidth measures separately.

3.4 VN Embedding

The VN Embedding process refers to the mapping of the virtual network topology (logical topology) on top of the substrate network topology (physical topology) subject to certain constraints. The constraints are normally manifested in

terms of the residual resource availability of the substrate network and the QoS parameters specified by the VN request. An example of a VN embedding can be seen in figure 1. From graph theoretic standpoint, the VN embedding process can be separated into two separate stages:

1-Node Embedding Phase: Each virtual node from a VN request is mapped to a single distinct substrate node by a one-to-one mapping:

$$\Gamma_N : N^V \leftarrow N^S, \tag{3}$$

such that $\Gamma_N(x) = \Gamma_N(y), \;\; iff \;\; x = y \;\; \forall x, y \in N^V$, subject to the cpu capacity constraints: $cpu(x) \leq R_N(\Gamma_N(x)) \;\; \forall x \in N^V$.

2-Link Embedding Phase: Each virtual link is mapped to either an unsplittable substrate path or a splittable multi-commodity flow based set of multiple paths between the substrate nodes corresponding to the endpoints of the virtual link. Mathematically, we have a mapping:

$$\Gamma_E : E^V \leftarrow \mathcal{P}^S, \tag{4}$$

such that $\forall v = (v_x, v_y) \in E^V$, and $\mathcal{P}^S$ is the set of simple paths of G^S. We have $\Gamma_E(v) \subseteq \mathcal{P}(\Gamma_N(v_x), \Gamma_N(v_y))$, subject to the bandwidth capacity constraints: $b(v) \leq R_E(p), \;\; \forall p \in \Gamma_E(e^V)$, where $\mathcal{P}(z, w)$ denotes the set of simple substrate paths between substrate nodes z and w, and $R_E(p) = \min_{s \in p} R_E(s)$. For any virtual link $v \in E^V$, we specify the set of QoS constrained substrate paths for v as $\mathcal{P}(v) = \{p \in \mathcal{P}^S | delay(p) \leq d(v)\}$.

3.5 Formulation of SVNE

We represent the input to SVNE as a tuple $< G^S, G^V, j, l, \{\alpha(s)\}_{s \in E^S} >$, where G^S and G^V represent the substrate and virtual networks respectively, j represents the service class of the SP owning G^V, $l \in E^S$ is the failed substrate link, and $\beta(s) = 1 - \alpha(s)$, such that $\beta(s)$ represents the percentage of bandwidth on each substrate link s reserved for backups. Let $\Pi(G^V)$ denote the revenue generated from G^V, where

$$\Pi(G^V) = T(G^V)[C_1 \sum_{v \in E^V} b\,(v) + C_2 \sum_{x \in N^V} cpu\,(x)] \tag{5}$$

C_1 and C_2 are weight factors which represent the relative importance of bandwidth and cpu to the generated revenue respectively. $T(G^V)$ represents the lifetime of the VN characterized by G^V. Each service class j is associated with a penalty function $\mathcal{S}_j(.)$, where $\mathcal{S}_j(v)$ represents the monetary penalty incurred if the bandwidth contract of virtual link v is violated.

Let $\mathcal{V}$ denote the set of all virtual links affected by the failure of l. Then the expected total penalty incurred by the InP to the corresponding SP is:

$$\mathcal{X}(G^V; l) = MTTR(l) \sum_{v \in \mathcal{V} \cap E^V} \mathcal{S}_j(v) \frac{db(v)}{b(v)} \tag{6}$$

$MTTR(l)$ is the mean time to repair for l. The difference between the bandwidth requested for v, and the actual bandwidth supplied by the InP is represented as $db(v)$. Let $G_1^S, G_2^S, G_3^S, \ldots$ be the sequence of VN requests, and $l_1, l_2, l_3, \ldots$ be the sequence of substrate link failure events. Then the objective of SVNE is to maximize long term business profit expressed as:

$$\Pi_\infty = \sum_{p=1}^{\infty} \sum_{q=1}^{\infty} [\Pi(G_q^V) - \mathcal{X}(G_q^V; l_p)] \tag{7}$$

4 HYBRID Policy Heuristic for SVNE

We propose a *hybrid policy* heuristic for solving SVNE. The heuristic consists of three separate phases. In the first phase, before any VN request arrives, the InP pro-actively computes a set of possible backup detours for each substrate link using a path selection algorithm. Therefore, for each substrate link l, we have a set D_l of candidate backup detours. The InP is free to utilize any path selection algorithm that suits its purposes, e. g. k-shortest path algorithm [18], column generation or primal dual methods [1]. The second phase is invoked when a VN request arrives. In this phase, the InP performs a node embedding using existing heuristics [22,6] and a multi-commodity flow based link embedding, that we denote as HYBRID_LP_LE. Finally, in the event of a substrate link failure, a reactive backup detour optimization solution HYBRID_LP_BDO is invoked which reroutes the affected bandwidth along candidate backup detours selected in the first phase. The pseudo-code for the hybrid policy is shown in the following algorithm (Figure 2).

```
1: procedure HRP(G^S(N^S, E^S))
2:     for all s ∈ E^S do
3:         pre-compute candidate detour set D_s.
4:     end for
5:     for all event arrivals do
6:         if event type == VN arrival then
7:             compute node embedding for VN G^V(N^V, E^V).
8:             solve HYBRID_LP_LE.
9:             update R_α(s), ∀s involved in HYBRID_LP_LE.
10:        end if
11:        if event type == Failure arrival then
12:            solve HYBRID_LP_BDO.
13:            update R_β(s), ∀s involved in HYBRID_LP_BDO.
14:        end if
15:    end for
16: end procedure
```

Fig. 2. Hybrid Recovery Policy

We now show the formulations of HYBRID_LP_LE and HYBRID_LP_BDO.

4.1 Formulation of HYBRID_LP_LE

In this phase we use a path flow based multi-commodity flow to embed all the virtual links simultaneously. For each pair $(x, y) \in V^S \times V^S$, we have a set of preselected end-to-end paths $\mathcal{P}(x, y)$. For a virtual link $v \in E^V$, we denote $\mathcal{P}(v) = \mathcal{P}(v_x, v_y)$ as the set of pre-selected QoS constrained simple paths for embedding v, where v_x and v_y are the end-points of v. Since the node embedding phase precedes the link embedding phase, we already know which virtual node is mapped to which substrate node. For any virtual link $v = (x', y') \in E^V$, we denote this as $x' \rightarrow \Gamma_N(x') = x$ and $y' \rightarrow \Gamma_N(y') = y$. HYBRID_LP_LE can be expressed as the following linear program:

HYBRID_LP_LE

-Objective Function

$$\text{Minimize} \sum_{v \in E^V} \sum_{p \in \mathcal{P}(v)} b(p, v) \tag{8}$$

Subject to

-Primary Capacity Constraint

$$\sum_{v \in E^V} \sum_{p \in \mathcal{P}(v)} \delta_s(p) b(p, v) \leq \mathcal{R}_\alpha(s), \ \ \forall s \in E^S. \tag{9}$$

-Primary Bandwidth Constraint

$$\sum_{p \in \mathcal{P}(v)} b(p, v) = b(v), \ \ \forall v \in E^V \tag{10}$$

Remarks

1. $\delta_s(p)$ is the link-path indicator variable, that is, $\delta_s(p) = 1$ if $s \in p$, 0 otherwise.
2. The objective function 8 corresponds to the revenue function Π for the VN.
3. $b(p, v)$ is the amount of bandwidth allocated on path p for virtual link v. A strictly positive value for $b(p, v)$ will indicate that p is a substrate path used for v. The values of $b(p, v)$ are stored and later used in the subsequent phase of the heuristic.
4. Constraint 9 is the primary capacity constraint which states that the total primary bandwidth allocated for all virtual links must be within the primary residual capacity of each substrate link.
5. Constraint 10 is the primary bandwidth constraint which specifies that the total bandwidth requirement of each virtual link must be distributed among all the QoS constrained paths allows for that virtual link.

4.2 Formulation of HYBRID_LP_BDO

HYBRID_LP_BDO can be expressed as the following linear program.

HYBRID_LP_BDO

-Objective Function

$$\text{Minimize} \sum_{v \in E^V} \mathcal{S}_j(v) \sum_{p \in \mathcal{P}(v)} \delta_l(p) \lceil b(p,v) \rceil [1 - \sum_{d \in \mathcal{P}_l} \frac{b(d,p,v)}{b(p,v)}] \tag{11}$$

Subject to

-Backup Capacity Constraint

$$\sum_{v \in E^V, p \in \mathcal{P}(v), d \in \mathcal{D}_l} \lceil b(p,v) \rceil \ \delta_s(d) \ b(d,p,v) \delta_l(p) \leq \mathcal{R}_\beta(s) \forall s \in E^S \tag{12}$$

-Recovery Constraint

$$\sum_{d \in \mathcal{D}_l, v \in E^V, p \in \mathcal{P}(v)} \delta_l(p) \ \lceil b(p,v) \rceil \ \delta_s(d) \ b(d,p,v) \leq \sum_{d \in \mathcal{D}_l, v \in E^V, p \in \mathcal{P}(v)} \delta_l(p) \ b(p,v) \tag{13}$$

Remarks

1. j represents the service class associated with the VN. Subsequently $\mathcal{S}_j(v)$ denotes the penalty incurred for violating the bandwidth reservation for a virtual link v belonging to a VN of service type j.
2. $\lceil x \rceil$ denotes the ceiling of x, that is $\lceil x \rceil = 1$ iff $x > 0$. So $\lceil b(p,v) \rceil = 1$ indicates that p is a path used for the embedding of v. Note that the $b(p,v)$ values are calculated and stored in the HYBRID_LP_LE phase.
3. For the failed substrate link l, we have the set of candidate backup detours, $\mathcal{D}_l = \mathcal{P}(l_x, l_y) \setminus \{l\}$.
4. $b(d,p,v)$ denotes the amount of rerouted bandwidth on detour $d \in \mathcal{D}_l$ for $b(p,v)$, that is for the primary path p allocated for virtual link v.
5. The objective (equation 11) refers to the penalty function formulated in equation 6.
6. Constraint 12 is the backup capacity constraint which states that the total backup flow on all the detours passing through a substrate link must be within the backup residual capacity of that substrate link.
7. Constraint 13 is the recovery constraint and it signifies that the total disrupted primary bandwidth must be allocated along the precomputed set of detours. The objective function ensures that the virtual links that have higher penalty values will be given priority in the recovery constraint.

Both HYBRID_LP_LE and HYBRID_LP_BDO are linear programs, as a result our proposed HYBRID policy is a polynomial time heuristic for SVNE. Another important feature of HYBRID is that it decouples primary and backup bandwidth provisioning. As a result, we don't need complex disjoint constraints in our solution which would have resulted in a hard mixed integer program. The objective functions of HYBRID_LP_LE and HYBRID_LP_BDO jointly solve the long term objective of SVNE as expressed in equation 7.

5 Performance Analysis

In this section, we first describe our simulation environment, then present evaluation results. Our evaluation is aimed at quantifying the performance of the proposed HYBRID policy approach to SVNE in terms of long term business profit for the InP by maximizing revenue earned from VN's and minimizing the penalty incurred due to substrate link failures.

5.1 Simulation Environment

We implemented a discrete event simulator for SVNE adapted from our ViNE-Yard simulator [6]. Since network virtualization is still not widely deployed, the characteristics of VN's and failure are not well understood. Specifically there are no analytical or experimental results on the substrate and virtual network topology characteristics, VN arrival dynamics or link failure dynamics in network virtualization. As a result, we use synthetic network topologies, and poisson arrival processes for VN's and link failures in our simulations. However our choice of substrate and virtual topologies and VN arrival process parameters are chosen in accordance with previous work on this problem [6, 20]. We used glpk [14] to solve the linear programs.

The substrate network topologies in our experiments are randomly generated with 50 nodes using the GT-ITM tool [18] in 25 x 25 grids. Each pair of substrate nodes is randomly connected with probability 0.5. The cpu and bandwidth resources of the substrate nodes and links are real numbers uniformly distributed between 50 and 100. We assume that both VN requests and substrate link failure events follow a Poisson process with arrival rates λ_V and λ_F. The ratio $\gamma = \frac{\lambda_F}{\lambda_V}$ is a parameter that we vary in our simulations. We use realistic values for $MTTR(l)$ based on failure characteristics of real ISP networks [15] which represent InP networks in a NVE. In each VN request, the number of virtual nodes is a uniform variable between 2 and 20. The average VN connectivity is fixed at 50%. The bandwidth requirement of a virtual link is a uniform variable between 0 and 50, and the penalty value $\mathcal{S}_j(v)$ for a virtual link v is set to a uniform random variable between 2 and 15 monetary units. In our simulations, we set $\alpha(s) = \alpha, \forall s$ belonging to the substrate network and vary α, where $0 \leq \alpha \leq 1$. For each set of experiments conducted, we plotted the average of 5 values for the performance metrics.

5.2 Comparison Method

Comparing our heuristics with previous work is difficult since the earlier heuristics do not consider substrate resource failures. As a result we evaluate our proposed *hybrid* policy against two base-line policies. The first one (we call it a *blind* policy) recomputes a new embedding for each VN affected by the substrate link failure. The second one is a *proactive* policy which pre-reserves both primary and backup bandwidth for each virtual link on link disjoint substrate paths. We omit details of these baseline policies due to space limitation. For

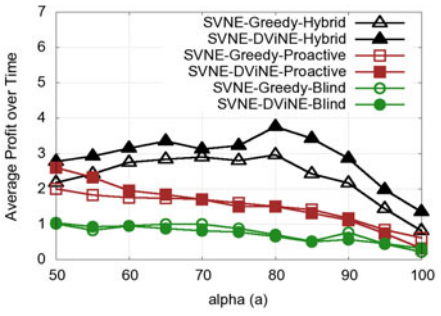

Fig. 3. Business profit against α

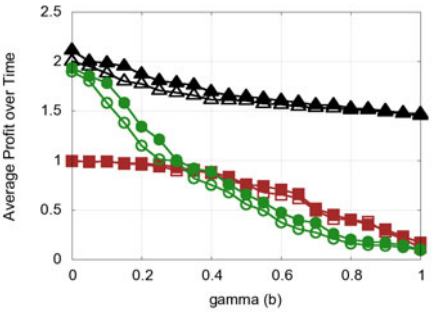

Fig. 4. Business profit against γ

node embedding, we use greedy [22] and DViNE heuristics [6]. In our evaluation, we have compared six algorithms that combine different node embedding strategies [22,6] with our proposed survivable link embedding strategies, namely, SVNE-Greedy-Hybrid, SVNE-DViNE-Hybrid, SVNE-Greedy-Proactive, SVNE-DViNE-Proactive, SVNE-Greedy-Blind, and SVNE-DViNE-Blind.

5.3 Evaluation Results

We use several performance metrics for evaluation purposes in our experiments. We measure the long term average profit earned by the InP by hosting VN's. The profit function depends on both the revenue earned from VN's by leasing resources and penalties incurred due to service disruption caused by substrate link failures. The penalty depends on both the amount of bandwidth violated due to a failure and the time it takes to recover from a failure as expressed in equations 6 and 7. We also measure the long term average acceptance ratio, percentage of backup bandwidth usage and response time to failures. We present our evaluation results by summarizing the key observations.

Acceptance Ratio and Business Profit. The hybrid policy leads to higher acceptance ratio and increased business profit in the presence of failures. Figures 3 shows the long term business profit against the percentage α of substrate link bandwidth for primary flows, while Figure 4 does it against the ratio of failure and VN rate γ. We notice that over the range of values for α and γ, the hybrid policy outperforms both the blind and proactive policies. Also the hybrid policy generates the highest profit at $\alpha = 80\%$, whereas the proactive and blind policies generate lesser profit with increased values of α. As α increases, there is less bandwidth available for backups on the substrate link and this hinders the performance of these policies. This also affects the hybrid policy, but it still has better performance due to its reactive nature. The profit and acceptance ratio for the blind policy drops more rapidly than the hybrid policy against increase in γ as shown in Figures 4 and 6. Although, the profit for the proactive policy

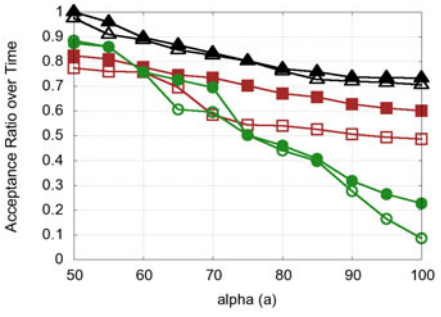

Fig. 5. Acceptance ratio against α

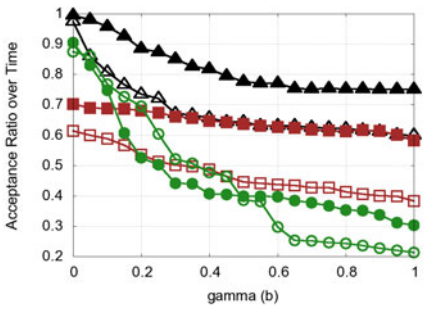

Fig. 6. Acceptance ratio against γ

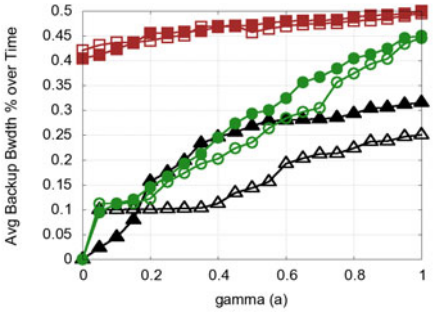

Fig. 7. Backup resource usage against γ

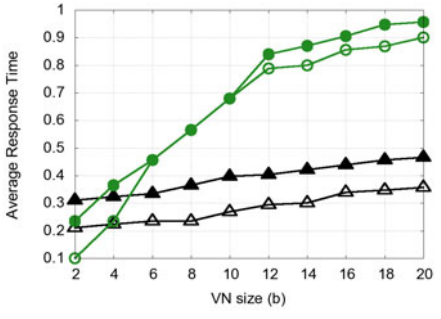

Fig. 8. Response time against VN size

increases with γ, it is still outperformed by the hybrid policy for the range of the simulation parameters.

Responsiveness to Failures. The hybrid policy has faster reaction time to failures than its counterparts. In Figure 8, we notice that the hybrid policy reacts faster than the blind policy when a failure occurs. When a substrate link fails, the blind policy recomputes the entire embedding, which is time consuming. The hybrid policy, on the other hand, only re-routes the bandwidth of the affected virtual links which results in faster response time and ultimately lower penalty values for the InP.

Bandwidth Efficiency. The hybrid policy is bandwidth efficient. The proactive policy pre-reserves additional bandwidth for each virtual link during the instantiation phase. On the other hand, the hybrid policy does not pre-reserve any backup bandwidth during the link embedding phase. It pre-selects the candidate paths for re-routing and allocates backup bandwidth only when an actual failure occurs. As a result, the average bandwidth usage increases less rapidly with γ compared to the blind policy. This is shown in Figure 7.

6 Conclusion

In this paper, we have formulated the SVNE problem to incorporate substrate failures in the virtual network embedding problem. We have also proposed an efficient HYBRID policy heuristic to solve SVNE. To the best of our knowledge this is the first attempt to add survivability to virtual network embedding algorithms along with support for business profit oriented optimization. Moreover, our proposed heuristic can be extended to deal with multiple link failures, and subsequently combined with a node migration strategy [19] to solve the single substrate node failure problem. However, there are a number of future research directions that can be pursued. Survivability in a multi-domain NVE could raise further challenges since it involves both intra and inter domain link failures. It would also be interesting to extend survivability to recursive NVE, where the first level VNs can act as InPs to a second level of VNs. Resource allocation, protection, and restoration issues in such recursive environments could be investigated under cross layer optimization or network utility maximization (NUM) frameworks.

Acknowledgment

This work was jointly supported by the Natural Science and Engineering Council of Canada (NSERC) under its Discovery program, Cisco Systems, and WCU (World Class University) program through the Korea Science and Engineering Foundation funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0).

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Englewood Cliffs (1993)
2. Aib, I., Boutaba, R.: On leveraging policy-based management for maximizing business profit. IEEE Transactions on Network and Service Management (TNSM) 4(2), 14 (2007)
3. Andersen, D.: Theoretical approaches to node assignment. Unpublished Manuscript (2002), `http://www.cs.cmu.edu/~dga/papers/andersen-assign.ps`
4. Anderson, T., Peterson, L., Shenker, S., Turner, J.: Overcoming the internet impasse through virtualization. Computer 38(4), 34–41 (2005)
5. Chowdhury, N., Boutaba, R.: Network virtualization: state of the art and research challenges (topics in network and service management). IEEE Communications Magazine 47(7), 20–26 (2009)
6. Chowdhury, N., Rahman, M., Boutaba, R.: Virtual network embedding with coordinated node and link mapping, April 2009, pp. 783–791. IEEE INFOCOM (2009)
7. Fan, J., Ammar, M.: Dynamic topology configuration in service overlay networks - a study of reconfiguration policies. In: IEEE INFOCOM (2006)
8. Feamster, N., Gao, L., Rexford, J.: How to lease the internet in your spare time. SIGCOMM Comput. Commun. Rev. 37(1), 61–64 (2007)

9. Houidi, I., Louati, W., Zeghlache, D.: A distributed virtual network mapping algorithm. In: Proceedings of IEEE ICC, pp. 5634–5640 (2008)
10. Kurant, M., Thiran, P.: Survivable Routing of Mesh Topologies in IP-over-WDM Networks by Recursive Graph Contraction. IEEE Journal on Selected Areas in Communications 25(5), 922–933 (2007)
11. Lee, K., Modiano, E.: Cross-layer survivability in wdm-based networks. In: INFOCOM 2009, April 2009, pp. 1017–1025. IEEE, Los Alamitos (2009)
12. Lischka, J., Karl, H.: A virtual network mapping algorithm based on subgraph isomorphism detection. In: Proceedings of ACM SIGCOMM VISA (2009)
13. Lu, J., Turner, J.: Efficient mapping of virtual networks onto a shared substrate. Washington University, Tech. Rep. WUCSE-2006-35 (2006)
14. Makhorin, A.: GNU Linear Programming Kit, Moscow Aviation Institute, Russia (2008), `http://www.gnu.org/software/glpk/`
15. Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C.-N., Ganjali, Y., Diot, C.: Characterization of failures in an operational ip backbone network. IEEE/ACM Trans. Netw. 16(4), 749–762 (2008)
16. Stern, T.E., Bala, K.: Multiwavelength Optical Networks: A Layered Approach. Addison-Wesley Longman Publishing Co., Inc., Boston (1999)
17. Thulasiraman, K., Javed, M.S., Xue, G.L.: Circuits/cutsets duality and a unified algorithmic framework for survivable logical topology design in ip-over-wdm optical networks. In: IEEE INFOCOM (2009)
18. Topkis, D.M.: A k shortest path algorithm for adaptive routing in communications networks. IEEE Transactions on Communications 36(7) (July 1988)
19. Wang, Y., Keller, E., Biskeborn, B., van der Merwe, J., Rexford, J.: Virtual routers on the move: live router migration as a network-management primitive. In: SIGCOMM, pp. 231–242. ACM, New York (2008)
20. Yu, M., Yi, Y., Rexford, J., Chiang, M.: Rethinking virtual network embedding: Substrate support for path splitting and migration. ACM SIGCOMM CCR 38(2), 17–29 (2008)
21. Zheng, Q., Shin, K.G.: Fault-tolerant real-time communication in distributed computing systems. IEEE Trans. Parallel Distrib. Syst. 9(5), 470–480 (1998)
22. Zhu, Y., Ammar, M.: Algorithms for assigning substrate network resources to virtual network components. In: Proceedings of IEEE INFOCOM (2006)