SVNE: Survivable Virtual Network Embedding Algorithms for Network Virtualization

Muntasir Raihan Rahman and Raouf Boutaba, *Fellow, IEEE*

***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 this assumption by formulating the survivable virtual network embedding (SVNE) problem. We then develop a pro-active, and a hybrid policy heuristic to solve it, and a baseline policy heuristic to compare to. The hybrid policy is based on a fast re-routing strategy and utilizes a pre-reserved quota for backup on each physical link. Our evaluation results show that our proposed heuristics for SVNE outperform the baseline heuristic in terms of long term business profit for the InP, acceptance ratio, bandwidth efficiency, and response time.**

***Index Terms*—Network virtualization, virtual network embedding, network survivability and resilience.**

I. Introduction

THE current Internet architecture has been supporting various distributed applications and heterogeneous network technologies quite successfully. However the immense popularity of the Internet has also turned out to be its biggest obstacle to seamless growth and innovation. The rigidity of the current Internet architecture has resulted in the so called Internet Ossification problem. Due to its multi-provider nature, adopting a new architecture or modifying the existing one requires consensus among multiple competing stakeholders. As a result, alterations to the current Internet are limited to incremental patches and deployment of new network applications have become increasingly difficult and error-prone.

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 [1]–[3]. 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.

Players in the network virtualization model differ from those in a traditional network environment. Here, the traditional role of the Internet Service Provider (ISP) is divided into two separate entities: (1) the infrastructure providers (InP) who are responsible for deploying and maintaining physical network resources (routers, links etc.) and (2) the service providers (SP) who implement various network protocols on virtual networks (VNs) for the end users by utilizing physical resources leased from multiple infrastructure providers. This allows multiple heterogeneous network architectures to be deployed without the inherent inflexibilities of the existing rigid Internet architecture. A service provider can also create child virtual networks in a recursive manner, and lease its child networks to other service providers, creating a hierarchy of virtual networks.

In network virtualization models, a weighted undirected graph $G^S = (N^S, E^S)$ usually represents the physical topology, where each node in the network is a vertex $v^S \in N^S$, with a set of attributes A_{v^S}. Each physical link between two nodes is represented by an edge $e^S \in E^S$ with an attribute set A_{e^S}. The virtual topology is similarly represented by another weighted graph $G_V = (N^V, E^V)$ with corresponding attribute sets. The virtual topology is also known as a logical topology. A virtual node can be a virtual host or a virtual router. A virtual host acts as a packet source or sink, whereas, a virtual router forwards packets according to the routing protocols specified for the virtual topology. A virtual link can span over multiple physical links, i.e., it usually corresponds to a physical path. Often a single virtual link can be mapped to multiple physical paths in-order to satisfy some of the constraints of the virtual link, e.g., bandwidth constraints that cannot be satisfied using a single path.

Efficient utilization of substrate network resources is dependent on effective techniques for virtual network embedding, which maps virtual networks on physical substrate network resources. The VN embedding problem is quite challenging, due to finite node and link resource constraints, admission control, and the on-line nature of virtual network requests. These properties make the VN embedding problem very difficult. Infact the problem remains computationally intractable even if some conditions are relaxed. Due to multiple constraints, the VNE problem is in general $NP-$hard, even in the off-line case. On the other hand, traditional techniques for solving on-line problems are not practical in this case, since the character-

Manuscript received October 23, 2011; revised October 15, 2012. The associate editor coordinating the review of the paper and approving it for publication was K. Van der Merwe.

This work was supported in part by the Natural Science and Engineering Council of Canada (NSERC) under the Smart Applications on Virtual Infrastructure (SAVI) Research Network, and in part by the World Class University (WCU) Program under the Korea Science and Engineering Foundation funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0).

M. R. Rahman is with the Department of Computer Science, University of Illinois, Urbana-Champaign, IL, 61801, USA (e-mail: mrahman2@illinois.edu).

R. Boutaba is with the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada, and also with the Division of IT Convergence Engineering, Pohang University of Science and Technology (POSTECH), Pohang 790-784, Korea (e-mail: rboutaba@cs.uwaterloo.ca).

Digital Object Identifier 10.1109/TNSM.2013.013013.110202

istics of the incoming VN requests are generally unpredictable and the search space is huge when the underlying substrate network is large.

Studies have shown that link failures occur as part of everyday operation in today's ISP networks [4]. Many networks deploy survivable IP over WDM (wave-length division multiplexing) techniques for resource multiplexing. Here IP is the logical layer, whereas WDM would be the physical layer. Failures in these infrastructures can occur at either *logical* or *physical layer*. Logical failures affect the logical layer only, and are transparent to the physical layer. In contrast, physical failures not only affect the physical layer, but also propagate to the logical layer, and if the network is recursively virtualized, any number of upper layers. We can also distinguish among *single* and *multiple* failures. The single failure case is more important and common, while considering multiple failures can facilitate modeling large systems under high stress. Although survivability from link failures has been thoroughly investigated for ISP and optical networks, the same results do not hold for network virtualization environments.

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. Since multiple link failures is a low probability event, we focus on single substrate link failures. In this paper, we don't explicitly deal with node failures. This is because any node failure aware virtual network embedding algorithm depends on tolerating adjacent link failures. As a result, we need to address link failures before dealing with node failures. Moreover, our proposed heuristic can be extended to deal with multiple link failures, and subsequently combined with a node migration strategy [5] to solve the single substrate node failure problem.

Our main contributions in this paper are as follows:

1) We add survivability mechanisms to the link mapping phase of virtual network embedding using efficient restoration and protection policies that can increase the long term business profit of the InP. We formulate the survivable virtual network embedding (SVNE) problem and provide efficient heuristics to solve it.
2) We add service level agreement (SLA) assurance to the embedding process by prioritizing the restoration of failed virtual links based on customer SLA constraints with the objective of minimizing the overall impact of failure and maximizing the business profit of the InP.
3) We propose a hybrid policy heuristic to solve SVNE. This solution is based on linear programming modules and has a number of configurable parameters. For example, the InP can control the percentage of resources dedicated for backup recovery and the number of paths allowed for primary and detour flows. This gives the InP greater control over its backup resource allocation policies and enables flexibility in determining the optimal allocation based on current failure patterns.
4) We introduce path-flow based optimization formulations for the different recovery and protection policies. Besides reducing the number of constraints and variables, there are other advantages in a path flow based formulation. Most notably, it allows control over the characteristics of the paths selected for embedding and protection. For instance, we can directly control the total number of paths and the number of hops per path for quality of service (QoS) management purposes.

This paper extends the results presented in [6] on several aspects. First, we include detailed mathematical optimization formulations of the proactive policy for SVNE which were omitted from the conference version. Second, the paper covers the basic concepts of network virtualization, virtual network embedding, and network survivability in more detail. Third, we also discuss the trade-offs of penalty function and business utility function formulation for the infrastructure provider in the presence of failures. Finally we have more experimental results including analysis of the effect of path selection algorithm parameters on SVNE policies and performance for special topologies like hub-and-spoke and mesh VN topologies. We also include a theoretical performance analysis for the online proactive policy for solving the SVNE problem.

The rest of the paper is organized as follows. Section II discusses related works on survivability and virtual network embedding. Section III formalizes the network model and formulates the virtual network embedding and survivable virtual network embedding problems. In section IV, we describe our proposed pro-active, and hybrid policy heuristics for the survivable virtual network embedding problem. Sections V, and VI describe node embedding algorithms, and path selection mechanisms, respectively, which we use in conjunction with our proposed SVNE solutions for link embedding. Section VII presents simulation results that evaluate the proposed heuristics. Finally, we conclude in Section VIII by identifying future research directions.

II. Related Work

A. Virtual Network Embedding

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 [7]. 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 [8], [9], ignoring either node or link requirements [8], [10], [11], assuming infinite capacity of the substrate nodes and links to obviate admission control [8]–[11], and focusing on specific virtual topologies [8]. Recently the authors in [12]–[14] proposed VNE heuristics that combine the node and link embedding phases. The authors in [15] proposed a distributed algorithm that simultaneously maps virtual nodes and virtual links without any centralized controller. Recently, the intra-domain algorithms to support VN embedding have

been extended across multiple administrative domains [16], [17], and geo-distributed cloud computing environments [18], [19]. The authors in [19] propose a hierarchical approach to solve the inter-domain embedding problem for networked clouds over multiple administrative domains.

However, a limitation of 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.

VN embedding is also related to the network testbed mapping problem [20]. The *Assign* algorithm used in the Emulab testbed [20] considers bandwidth constraints alongside constraints on exclusive use of nodes (i.e., different VNs cannot share a substrate node). But sharing of substrate nodes and links by multiple VNs is one of the core principles of network virtualization [2], and VN embedding algorithms must support these objectives. Emulab itself is aligning its resource mapping policies with that of network virtualization [21].

B. Survivability

Survivable Virtual Network Embedding (SVNE) or virtual network embedding in the presence of arbitrary node and link failures is a research challenge that has yet to be addressed in the network virtualization literature. Node and link failure survivability problems have been investigated extensively for optical and multi-protocol label switched (MPLS) networks [22], and real time systems [23]. Two well known approaches for handling link failures in optical networks are protection and restoration. Protection is normally employed at the substrate network level during the design phase by provisioning backup light-paths. On the other hand restoration is done at the virtual network level by provisioning the network with additional capacity and is more reactive in nature. The key to efficient restoration mechanisms is survivable mapping in the presence of link failures. The authors in [24] mention three existing paradigms for survivable IP-over-WDM mapping algorithms based on (1) Integer Linear Programs (ILP), (2) Meta-heuristics like Genetic Algorithms (GA), Ant Colony Optimization (ACO), Tabu Search, and (3) Graph Theoretic algorithms. The most recent approach based on graph theoretic results called SMART [25], [26] is more efficient and scalable than ILP and heuristic local search approaches.

SMART repeatedly picks connected subgraphs of the logical topology and finds survivable mappings for them. It then reduces the logical topology by contracting the already mapped subgraph and continues the process. The authors in [24] continue working in this direction by exploiting duality between circuits and cuts due to the Max-flow min-cut theorem in Combinatorial Optimization [27]. They propose primal and dual algorithms that extend SMART (called CIRCUIT-SMART and CUTSET-SMART) and develop some heuristics to speed up their algorithms. Recently the authors in [28] extended the Max-flow min-cut theorem for multi-layer networks. They proposed new connectivity metrics suited for multi-layer networks and developed some heuristics for maximizing connectivity in the logical layer.

Our work on survivable virtual network embedding (SVNE) differs in a number of aspects, due to unique challenges introduced by the network virtualization environment (NVE). First, the VNE problem is on-line in nature, whereas the survivable logical topology design problem in optical and multi-protocol label switched (MPLS) networks [24], [28] is off-line. Second, 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 nodes remain connected in the presence of failures, even if they are not connected via a direct overlay 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 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 [9], [12]–[14], [29].

In optical networks, end-to-end connection requests arrive on-line and are processed as soon as they arrive [22]. For a VN request, we have to guarantee survivability of all the VN links simultaneously, which makes the problem harder. We differentiate between *Weak* and *Strong* survivability in the context of SVNE. Weak survivability only ensures that the virtual nodes will stay connected in the presence of failures. Strong survivability guarantees that the original VN topology remains intact in the presence of failures. Failures in the underlying physical network can give rise to complex multi-layer failures in the network virtualization environment. Any such failure can effectively cause a cascading series of errors in the virtual networks directly hosted on those substrate network components, and possibly in many others that are recursively designed. In NVE, we require strong survivability since in the basic revenue model for NVE, the service provider pays an amount that is proportional to the resource (cpu for nodes, bandwidth for links) and VN topology requirements of the virtual network request. This means that provisioning of backup resources is essential in an NVE, since without backup provisioning we can only ensure weak survivability.

III. SVNE Problem Formulation and Solutions

In this section, we provide a mathematical formulation of the survivable virtual network embedding (SVNE) problem as an extension of the VNE problem. We then devise efficient heuristics to solve SVNE. Since we deal with substrate link failures in this paper, our main focus is on the second phase of VNE, that is the link embedding phase. For node embedding, we use the existing heuristics proposed in the literature. As a result our approach to on-line SVNE for each incoming VN request is as follows:

TABLE I
SUMMARY OF KEY NOTATIONS USED IN THE PAPER

Notation	Description
$G^S(N^S, E^S)$	Substrate graph
$G^V(N^V, E^V)$	Virtual network request
$R_N(x)$	Residual cpu capacity of a substrate node
$R_E(s)$	Residual bandwidth capacity of a substrate link
$\alpha(s)(\beta(s))$	Percentage of bandwidth reserved for primary (backup) flows on substrate link s
$\mathcal{P}^S$	Set of simple paths in the substrate graph
$\Gamma_N : N^V \leftarrow N^S$	Node embedding function
$\Gamma_E : E^V \leftarrow \mathcal{P}^S$	Link embedding function
$\Pi(.)$	Revenue function
$\mathcal{X}(.)$	Penalty function
$\delta_s(p)$	Link-path indicator variable, $\delta_s(p) = 1$ if link $s \in p$
$b(p, v)$	Bandwidth allocated on path p for virtual link v
$b(d, p, v)$	Re-routed bandwidth on detour d for $B(p, v)$

- Node Embedding: Greedy [9], [29], Mixed Integer Programming [13].
- Link Embedding: Add survivability policies to handle arbitrary substrate link failures. [our contribution].

The existing node embedding heuristics and path selection mechanisms used in our SVNE solutions are described in subsequent sections. The key mathematical notations used for the SVNE problem formulation and solutions are summarized in Table I.

A. 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 $loc(x)$. 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.

B. 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). A virtual node $y \in N^V$ 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[1] 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.

[1]A substrate path that repeats no substrate node.

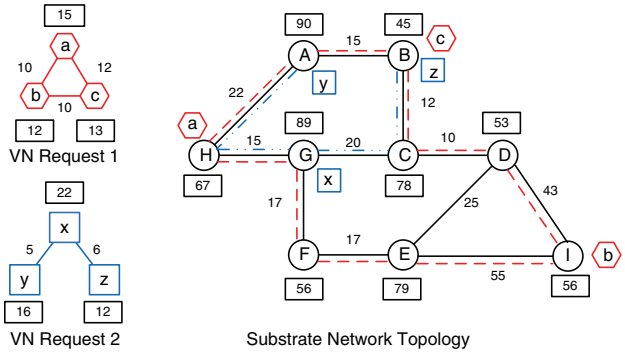

Fig. 1. Mapping of VN requests onto a shared substrate network.

C. 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 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 III-D). 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 occurrences and repairs.

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.

D. 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 on the right in figure 1. Here the solid lines represent the substrate network topology. The dashed (colored) lines represent the substrate links used for embedding the corresponding virtual networks. The virtual nodes are shown beside the substrate node they have been mapped on. From a graph theoretic standpoint, the VN embedding process can be divided into two 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: $\Gamma_N : N^V \leftarrow N^S$, 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 paths between the substrate nodes corresponding to the endpoints of the virtual link. Mathematically, we have a mapping: $\Gamma_E : E^V \leftarrow \mathcal{P}^S$, 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 $\mathcal{P}(v) = \{p \in \mathcal{P}^S | delay(p) \leq d(v)\}$.

Figure 1 shows an example of the embedding process. The substrate and virtual nodes are labeled with letters inside the corresponding node. We have the embedding of G^V on G^S as Γ, where Γ is defined as follows: $\Gamma_N(a) = C, \Gamma_N(b) = H, \Gamma_N(c) = B$ and $\Gamma_E(ab) = \{CD, DG, GH\}, \Gamma_E(bc) = \{HF, FE, EB\}, \Gamma_E(ac) = \{CA, AB\}$.

E. Penalty Function and Business Utility for InP

An SP negotiates a Service Level Agreement (SLA) with the InP for uninterrupted service throughout the lifetime of its requested VN. If the SLA contract is violated due to a substrate resource failure, then this results in frustration on part of the SP and subsequent penalty for the InP based on the level of frustration of the SP. Each SP owning a VN is characterized by a Service class which is represented by a function $\mathcal{S}_j(db)$, where $j \in \{1, 2, \ldots, C\}$ and C denotes the number of distinct service classes and db denotes the bandwidth differential, that is the difference between requested bandwidth and the bandwidth granted by the InP. We can model $\mathcal{S}_j(db)$ as an increasing function, however for simplicity, we assume that the function takes the shape of a step function, that is for $db < T_B$, $\mathcal{S}_j(db) = 0$, and for $t > T_B$, $\mathcal{S}_j(db) = P$. We call T_B the *frustration threshold* for the service class j. Therefore the set of all SPs is partitioned into equivalence classes based on their respective service class associations. We denote the mapping between VNs and service classes as $\varphi(.)$, where $\varphi(i) = j$ means that VN i is associated with service class j. Since we have reserved a percentage of bandwidth on each substrate link for backups, it cannot be ensured that all the SPs will retain their complete VN topology when a failure occurs. In that case our objective will be to minimize the total penalty incurred due to SP frustration. For each $v \in E^V$ and service class j for a VN, we will denote $\mathcal{S}_j(v)$ as the penalty incurred due to service disruption.

F. Formulation of SVNE

We represent the input to SVNE as an $|E^S| + 4$ 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

$$\Pi(G^V) = T(G^V)[C_1 \sum_{v \in E^V} b(v) + C_2 \sum_{x \in N^V} cpu(x)] \quad (3)$$

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:

$$\mathcal{X}(G^V; l) = MTTR(l) \sum_{v \in \mathcal{V} \cap E^V} \mathcal{S}_j(v) \frac{db(v)}{b(v)} \quad (4)$$

$MTTR(l)$ 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 $G_1^V, G_2^V, G_3^V, \ldots$ 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:

$$\Pi_\infty = \sum_{p=1}^{\infty} \sum_{q=1}^{\infty} [\Pi(G_q^V) - \mathcal{X}(G_q^V; l_p)] \quad (5)$$

G. Protection and Restoration Models

For fast protection against substrate link failures, we employ two types of restoration mechanisms in this paper, namely link (local) restoration and path (end-to-end) restoration. In the existing literature on survivable topology design, both of these mechanisms fall under the category of fast restoration mechanisms, due to their low restoration latency. The main objective in this paper is to provide link embedding heuristics with fast restoration.

1) Link Protection and Restoration: For protecting a substrate path p corresponding to a virtual link against single link failures, we associate a primary path $W(p)$ and for each substrate link $e \in p$, a local backup detour $B_e(p)$. So for a substrate path with k substrate links, there will be k link detours for fast restoration against single link failures. From now on, when we refer to a path p in this model, it will consist of $\{W(p), B_e(p, \forall e \in p)\}$. It should also be mentioned that backup detours for different substrate paths can share bandwidth on their common substrate links. Link restoration is also known as local restoration due to the localized fault tolerance mechanism around each substrate link.

2) Path Protection and Restoration: In the simplest approach to path restoration, namely $1:1$ protection, a connection p consists of a primary working path and a link disjoint backup path. Our approach to path restoration is more sophisticated in that we use a survivable version of the multi-commodity flow problem for path restoration in link embedding [30]. A survivable flow from a substrate node u to v consists of a primary flow of value f among the paths from u to v, and a distinct secondary flow of the same value f in such a way that both flows pass through link disjoint paths. Path restoration is also known as global restoration due to its end-to-end fault tolerance nature.

IV. Heuristics for SVNE

In this section, we describe our proposed solutions to the SVNE problem. Our first solution is proactive, that is, it provisions redundant resources for possible failures in the future. This solution is suitable for scenarios where even the slightest delay due to failures is unacceptable. A drawback of this approach is wastage of resources. In a failure-free case, we will waste approximately 50% substrate network resourses. Our second solution overcomes this limitation by using a reactive approach, where failures are handled after we have an actual substrate link failure. We describe these two solutions in the next sub-sections.

A. PROACTIVE Policy Heuristic for SVNE

The PROACTIVE policy protects each virtual link using a survivable version of the multi-commodity flow problem [30]. For each virtual link v, we send a primary flow of value $b(v)$ and also a secondary flow of value $b(v)$ among the QoS constrained paths allowed for v. To protect against single substrate link failures, we have to ensure that primary and secondary flows are edge disjoint. We formulate the problem as a mixed integer program in the following manner:

PROACTIVE_MIP_LE

Minimize

-Objective Function

$$\sum_{v\in E^V}\mathcal{S}_j(v)[1-\sum_{p\in\mathcal{P}(v)}\frac{b_2(p,v)}{b(v)}]+\sum_{v\in E^V,p\in\mathcal{P}(v)}[b_1(p,v)+b_2(p,v)] \tag{6}$$

Subject to

-Primary and Secondary Capacity Constraints

$$\sum_{v\in E^V,p\in\mathcal{P}(v)}\delta_s(p)b_1(p,v)\leq\mathcal{R}_\alpha(s)\ \ \forall s\in E^S \tag{7}$$

$$\sum_{v\in E^V,p\in\mathcal{P}(v)}\delta_s(p)b_2(p,v)\leq\mathcal{R}_\beta(s)\ \ \forall s\in E^S \tag{8}$$

-Primary and Secondary Bandwidth Constraints

$$\sum_{p\in\mathcal{P}(v)}b_1(p,v)=b(v),\ \ \forall v\in E^V \tag{9}$$

$$\sum_{p\in\mathcal{P}(v)}b_2(p,v)\leq b(v),\ \ \forall v\in E^V \tag{10}$$

-Disjointness Constraints

$$b_1(p,v)\leq b(v)\sigma_1(p,v),\ \ \forall v\in E^V,\forall p\in\mathcal{P}(v) \tag{11}$$

$$b_2(p,v)\leq b(v)\sigma_2(p,v),\ \ \forall v\in E^V,\forall p\in\mathcal{P}(v) \tag{12}$$

$$\delta_s(p)\delta_s(q)[\sigma_1(p,v)+\sigma_2(p,v)]\leq 1,\ \ \forall s\in E^S \tag{13}$$

-Variables

$$\sigma_1(p,v)\in\{0,1\},\ \ \forall v\in E^V,\forall p\in\mathcal{P}(v) \tag{14}$$

$$\sigma_2(p,v)\in\{0,1\},\ \ \forall v\in E^V,\forall p\in\mathcal{P}(v) \tag{15}$$

$$b_1(p,v)\geq 0,\ \ \forall v\in E^V,\forall p\in\mathcal{P}(v) \tag{16}$$

$$b_2(p,v)\geq 0,\ \ \forall v\in E^V,\forall p\in\mathcal{P}(v) \tag{17}$$

B. Remarks

- j denotes 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.
- $\delta_s(p)$ is a link-path indicator variable, i.e., $\delta_s(p)=1$ if $s\in p$, 0 otherwise.
- $b_1(p,v)$ and $b_2(p,v)$ represent the primary and backup flows on the simple path p for the virtual link v.
- The objective function 6 has two parts. The first part is for minimizing the total penalty incurred due to bandwidth violations, whereas the second part is concerned with minimizing the overall substrate network usage for primary and secondary flows.
- Constraints 7 and 8 are the primary and secondary capacity constraints, and they specify that for each substrate link, the overall bandwidth used for primary and secondary flows must be within the primary and backup residual capacities of that substrate link respectively.
- Constraints 9 and 10 are bandwidth constraints for primary and secondary flows respectively.
- Constraints $11, 12$ and 13 represent the disjointness constraints. They are expressed in terms of the two integer variables σ_1 and σ_2. The third constraint in this set of constraints enforces that only one of them can take the value 1. If σ_1 is 0, then the first constraint forces b_1 to 0 also. However if σ_1 is 1, then the first constraint is trivially satisfied. The case for σ_2 is similar.

C. Solution Approaches

PROACTIVE_MIP_LE is a mixed integer program, and hence $NP-$hard to solve. The usual approach is to relax the integer constraints and solve the relaxed Linear Program (LP) to obtain a fast heuristic. However the integrality of the MIP stems from the disjointness constraints 13, 14, and 15 which force the primary and backup flows to pass through link disjoint paths. It should be noted that we have a dedicated percentage of bandwidth resources for backups on each substrate link through the $\alpha(s),\beta(s)$ values for each substrate link $s\in E^S$. This *separation property* readily leads towards a fast simple heuristic using two sequential LP's as follows:

PROACTIVE_LP_LE_P

-Objective Function

Minimize

$$\sum_{v\in E^V, p\in\mathcal{P}(v)} b_1(p,v) \tag{18}$$

Subject to

$$\sum_{v\in E^V, p\in\mathcal{P}(v)} \delta_s(p) b_1(p,v) \leq \mathcal{R}_\alpha(s) \ \ \forall s \in E^S \tag{19}$$

$$\sum_{p\in\mathcal{P}(v)} b_1(p,v) = b(v), \ \ \forall v \in E^V \tag{20}$$

We define a boolean variable $\varphi(s), \forall s \in E^S$ which keeps track of the substrate links that have been used for sending primary flow. These values are then used in the second LP to avoid conflicts between primary and backup flows on the same substrate link.

PROACTIVE_LP_LE_B

-Objective Function

Minimize

$$\sum_{v\in E^V} \mathcal{S}_j(v)[1 - \sum_{p\in\mathcal{P}(v)} \frac{b_2(p,v)}{b(v)}] + \sum_{v\in E^V, p\in\mathcal{P}(v)} b_2(p,v) \tag{21}$$

Subject to

$$\sum_{v\in E^V, p\in\mathcal{P}(v)} \delta_s(p) b_2(p,v) \leq (1-\varphi(s))\mathcal{R}_\beta(s) \ \ \forall s \in E^S \tag{22}$$

$$\sum_{p\in\mathcal{P}(v)} b_2(p,v) \leq b(v), \ \ \forall v \in E^V \tag{23}$$

It should be noted that we have multiplied the term $(1 - \varphi(s))$ to the right hand side of the first constraint in 22. If a substrate link s has been used for a primary flow, then $\varphi(s)$ will be 1, forcing the right hand side of that constraint to be 0. This ensures the disjointness of the primary and secondary flows. We now have a polynomial time LP based heuristic shown in Algorithm 1.

Algorithm 1 LP Based Heuristic for Proactive Recovery Policy (LPHPP)

```
procedure LPHPP(G^S, G^V)
    Solve PROACTIVE_LP_LE_P
    for all s ∈ E^S do
        φ(s) = 0
    end for
    for all v ∈ E^V do
        for all p ∈ P(v) do
            if b_1(p,v) > 0 then
                φ(s) = 1
            end if
        end for
    end for
    Solve PROACTIVE_LP_LE_B
end procedure
```

D. Competitive Ratio of LPHPP

We can think of LPHPP as a greedy algorithm, since it first assigns bandwidth for primary flows and marks the substrate links used. Marking a substrate link is equivalent to deleting that link while assigning bandwidth to backup flows, and this is the key property that ensures link disjointness for primary and backup flows for each virtual link. LPHPP is also an online algorithm, since we have to find an embedding for each new virtual network. As a result, LPHPP will perform worse compared to an optimal offline algorithm. We have the following theorem specifying the worst case competitive ratio for LPHPP.

Theorem 1 Let $G^S(N^S, E^S)$ be a substrate network and $< G_i^V >$ be a sequence of virtual networks to be embedded on G^S. Let OPT denote an optimal off-line algorithm for the $SVNE$ problem. Then $\mathcal{A}(LPHPP) \geq (n-1)\mathcal{A}(OPT)$, where $n = |N^S|$ is the size of the substrate network, and $\mathcal{A}(.)$ is a function that returns the number of VN requests fulfilled by a particular algorithm.

Proof

The proof is by construction in two phases. In the first phase, we show how to convert an instance of $SVNE(l)$ to an instance of failure free VNE for LPHPP, where l is the failed substrate link. Next, we show the existence of a worst case substrate graph, and a worst case sequence of input requests that satisfy the competitive ratio bound.

For the first phase, we can create an instance of VNE corresponding to $SVNE(l)$, by replacing each virtual link request v in $SVNE(l)$ with bandwidth requirement $b(v)$ with two virtual links in VNE, each with bandwidth $b(v)$. We can easily see that solving VNE is equivalent to solving $SVNE(l)$ using LPHPP.

For the second phase, consider a chain substrate network consisting of n substrate nodes. Assume the first VN request requires a virtual link $(1, n)$ from node 1 to node n, and there are $n-1$ subsequent single virtual link VN requests of the form $(i, i+1)$, for $i = 1, 2, \ldots, n-1$, where each VN requires a single virtual link with end-points at i, and $i+1$. The greedy algorithm LPHPP accepts the first request and cannot accept any other requests. So $\mathcal{A}(LPHPP) = 1$. However the optimal solution OPT would have accepted the later $n-1$ VN requests, so $\mathcal{A}(OPT) = n-1$. Since this is a worst-case scenario, we have, $\mathcal{A}(LPHPP) \geq (n-1)\mathcal{A}(OPT)$. □

E. HYBRID Policy Heuristic for SVNE

Since, LPHPP has poor worst-case performance, we propose a *hybrid policy* heuristic for solving SVNE, which avoids the complexity associated with mixed-integer programs. 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 [31], or column generation or primal dual methods [32]. The second phase is invoked when a VN request arrives. In this phase, the InP performs a node embedding using existing heuristics [9], [13], [14]

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 flows along candidate backup detours selected in the first phase. The pseudo-code for the hybrid policy is shown in Algorithm 2.

Algorithm 2 Hybrid Policy Heuristic (HRP)

procedure HRP(G^S, G^V)
  for all $s \in E^S$ **do**
    pre-compute candidate detour set $\mathcal{D}_s$.
  end for
  for all event arrivals **do**
    if event type == VN arrival **then**
      compute node embedding for $G^V(N^V, E^V)$.
      solve **HYBRID_LP_LE**.
      for all s used in **HYBRID_LP_LE do**
        update $\mathcal{R}_\alpha(s)$.
      end for
    end if
    if event type == Failure arrival **then**
      solve **HYBRID_LP_BDO**.
      for all s used in **HYBRID_LP_BDO do**
        update $\mathcal{R}_\beta(s)$.
      end for
    end if
  end for
end procedure

We now show the formulations of HYBRID_LP_LE and HYBRID_LP_BDO.

F. Formulation of HYBRID_LP_LE

In this phase we use a path based multi-commodity flow formulation 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

$$\text{Minimize} \sum_{v \in E^V} \sum_{p \in \mathcal{P}(v)} b(p, v) \tag{24}$$

Subject to

-Primary Capacity Constraint

$$\sum_{v \in E^V} \sum_{p \in \mathcal{P}(v)} \delta_s(p) b(p, v) \leq \mathcal{R}_\alpha(s), \ \ \forall s \in E^S. \tag{25}$$

-Primary Bandwidth Constraint

$$\sum_{p \in \mathcal{P}(v)} b(p, v) = b(v), \ \ \forall v \in E^V \tag{26}$$

1) Remarks:

- $\delta_s(p)$ is the link-path indicator variable, that is, $\delta_s(p) = 1$ if $s \in p$, 0 otherwise.
- The objective function 24 corresponds to the revenue function Π for the VN.
- $b(p, v)$ 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.
- Constraint 25 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.
- Constraint 26 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 allowed for that virtual link.

G. Formulation of HYBRID_LP_BDO

HYBRID_LP_BDO can be expressed as the following linear program.

HYBRID_LP_BDO

-Objective Function

$$\text{Minimize} \sum_{v \in E^V} \mathcal{S}_j(v) \sum_{p \in \mathcal{P}(v)} \delta_l(p) \lceil b(p, v) \rceil [1 - \sum_{d \in \mathcal{P}_l} \frac{b(d, p, v)}{b(p, v)}] \tag{27}$$

Subject to

-Backup Capacity Constraint

$$\sum_{v \in E^V, p \in \mathcal{P}(v), d \in \mathcal{D}_l} \lceil b(p, v) \rceil \ \delta_s(d) \ b(d, p, v) \delta_l(p) \leq \mathcal{R}_\beta(s) \forall s \in E^S \tag{28}$$

-Recovery Constraint

$$\sum_{d \in \mathcal{D}_l, v \in E^V, p \in \mathcal{P}(v)} \delta_l(p) \ \lceil b(p, v) \rceil \ \delta_s(d) \ b(d, p, v) \leq \sum_{d \in \mathcal{D}_l, v \in E^V, p \in \mathcal{P}(v)} \delta_l(p) \ b(p, v) \tag{29}$$

1) Remarks:

- 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.
- $\lceil x \rceil$ 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.
- For the failed substrate link l, we have the set of candidate backup detours, $\mathcal{D}_l = \mathcal{P}(l_x, l_y) \setminus \{l\}$.
- $b(d, p, v)$ 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.
- The objective (equation 27) refers to the penalty function formulated in equation 4.

- Constraint 28 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.
- Constraint 29 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 during the recovery.

2) Discussion: Our proposed HYBRID policy is a polynomial time heuristic for SVNE, since both HYBRID_LP_LE and HYBRID_LP_BDO are linear programs. 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 5.

V. Heuristics for Node Embedding

A. Greedy Node Embedding

The main advantage of a greedy node embedding heuristic is that it is simple and cost efficient, in contrast to iterative methods or meta-optimization techniques, e.g., simulated annealing. The greedy algorithm maps virtual nodes to substrate nodes with maximum residual substrate resources in order to minimize the use of resources at bottleneck nodes and links [9], [29]. The metric quantifying available substrate resources is $H(x) = R_N(x) \sum_{l \in L(x)} b(l)$, where $L(x)$ is the set of links adjacent to x, and $R_N(x)$ is the residual cpu capacity of x. This metric leads to the greedy node embedding algorithm in Algorithm 3, which assumes batch processing, i.e., the InP collects VN requests at the end of a fixed time interval, allocates them simultaneously. The algorithm can be easily converted to a pure online algorithm.

Algorithm 3 Greedy node embedding algorithm

procedure GNE($G^V = (N^V, E^V)$)
  Sort VN requests according to revenue.
  exit if no requests left.
  Take the request with largest revenue.
  Find the set of substrate nodes S that satisfy restrictions and available cpu capacity.
  if $S = \{\}$ **then**
    exit.
  end if
  For each virtual node $n \in G^V$, find the substrate node $x = argmax_x H(x)$. Map n to x.
end procedure

B. D-ViNE Algorithm

In this section, we describe a node embedding heuristic based on a mixed integer programming formulation that maximizes correlation between node and link embedding phases in order to increase revenue and minimize cost [13], [14]. The basic idea is to augment the substrate graph and simultaneously map the virtual nodes and links using a mixed integer programming formulation. Since mixed integer programs are computationally intractable, the authors used relaxation and rounding to develop polynomial time heuristics. For details, we refer the reader to [13], [14]. We use these existing node embedding algorithms to implement the node embedding phase of our SVNE solutions.

VI. Path Selection Mechanisms

The effectiveness of the proposed heuristics depend on efficient path selection mechanisms. Especially the proposed hybrid policy heuristic can adopt any path selection algorithm that suits its purpose. In this section we delineate various path selection mechanisms that can be utilized in our solutions.

A. Static Path Selection Heuristics

The k-shortest path algorithm is the simplest heuristic that can be employed in our solutions. It is a static algorithm, in the sense that the path set for each virtual link remains constant throughout the duration of the virtual network. In our experiments, we used a k-shortest path algorithm adapted for efficient path computation in communication networks [31].

B. Dynamic Path Selection Heuristics

The first dynamic path selection approach is to use a primal dual formulation, where we first find the dual LP associated with the relaxed primal LP by swapping variables and constraints. The primal dual approach leads to an iterative algorithm which raises another issue of fast convergence. Normally theoretical convergence guarantees of iterative primal dual algorithms do not always work well in practice. That is why we opt for the simpler static path selection algorithms in our experiments.

The second approach for handling dynamic path selection is to employ a column generation approach. Here instead of solving the LP with all the flow variables at once, only a subset of path variables is considered at each step. After solving the LP at each step with an active set of paths, the path set is updated by adding improved paths and removing unused ones. This is also an iterative algorithm, however the convergence test is usually simpler and more practical than the primal dual approach. But an additional issue with this approach is updating the active path set at each step using an efficient path selection heuristic.

VII. Performance Evaluation

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

TABLE II
SIMULATION PARAMETERS AND PERFORMANCE METRICS

Notation	Description
Parameter: α	Percentage of bandwidth of a substrate link for primary flow.
Parameter: $\gamma = \frac{\lambda_F}{\lambda_V}$	Ratio of failure and VN arrival rate.
Parameter: n_V	Number of VN nodes.
Parameter: k	Number of paths allowed for link embedding and detours.
Metric: π	Average business profit in the long run.
Metric: ar	Average acceptance ratio in the long run.
Metric: bru	Average backup resource usage percentage in the long run.
Metric: t	Average response time to a failure.

TABLE III
COMPARED ALGORITHMS

Notation	Algorithm Description
SVNE-Greedy-Hybrid	Greedy Node Embedding with Hybrid Policy
SVNE-DViNE-Hybrid	DViNE Node Embedding with Hybrid Policy
SVNE-Greedy-Proactive	Greedy Node Embedding with Proactive Policy
SVNE-DViNE-Proactive	DViNE Node Embedding with Proactive Policy
SVNE-Greedy-Blind	DViNE Node Embedding with Blind Policy
SVNE-DViNE-Blind	DViNE Node Embedding with Blind Policy

A. Simulation Environment

We implemented a discrete event simulator for SVNE adapted from our ViNE-Yard simulator (http://www.mosharaf.com/ViNE-Yard.tar.gz) [13], [14]. Since network virtualization is still not widely deployed, the characteristics of VNs and failures are not well understood yet. 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 environments. As a result, we use synthetic network topologies, and Poisson arrival processes for VNs 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 [13], [14], [29]. We used the GNU linear programming toolkit (`glpk`) to solve all the linear programs in our formulations on an Ubuntu 12.04 virtual machine on top of Windows 7. The hardware platform used for the simulation experiments was an Intel(R) Core(TM) i5-2410 2.3GHz processor with 8GB RAM.

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, respectively. 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 [33] which represent InP networks in a NVE. The $MTTR$ values were generated using an empirical distribution derived in [33]. 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. The simulation parameters and output performance metrics are shown in Table II.

B. BLIND Policy Heuristic for SVNE

The BLIND policy is the simplest scheme among all the policies, hence the name. This policy is oblivious to any underlying structure of the problem space and the failure pattern. Whenever a substrate link fails, the BLIND policy simply recomputes a new link embedding for each VN affected by the substrate link failure. Although this policy seems simple, it has a high recovery complexity and reconfiguration cost, since even though the substrate link failure will only affect a localized portion of the embedding of a VN, it still recomputes the entire embedding.

C. Comparison Method

Comparing our heuristics with previous work is difficult since the earlier heuristics do not consider substrate resource failures. As a result we compare our proposed *hybrid* and *proactive* policy solutions against the base-line *blind* policy. For node embedding, we use greedy [9] and DViNE heuristics [13], [14]. In our evaluation, we have compared six algorithms that combine different node embedding strategies [9], [13], [14] with our proposed survivable link embedding strategies, as shown in Table III.

D. 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 VNs. The profit function depends on both the revenue earned from VNs 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 4 and 5. 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 in the following subsections.

1) Acceptance ratio and Business profit: The hybrid policy leads to higher acceptance ratio and increased business profit in the presence of failures. Figures 2, and 3 show the long term business profit against the percentage α of substrate link bandwidth for primary flows, and the ratio of failure and VN rate γ, respectively. 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

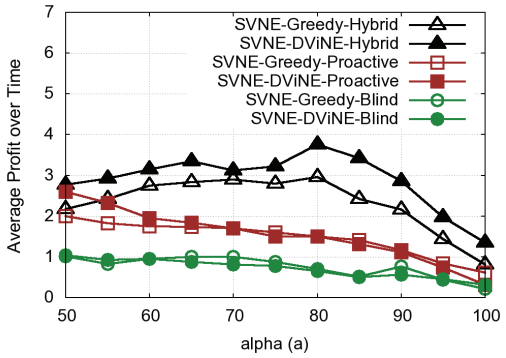

Fig. 2. Business profit against α.

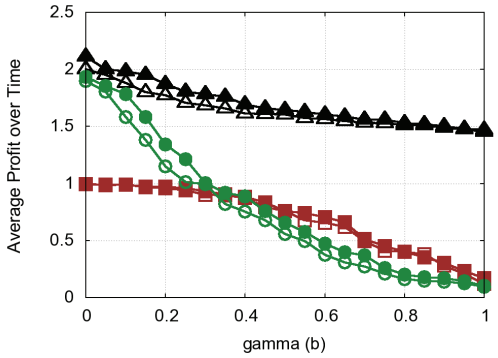

Fig. 3. Business profit against γ.

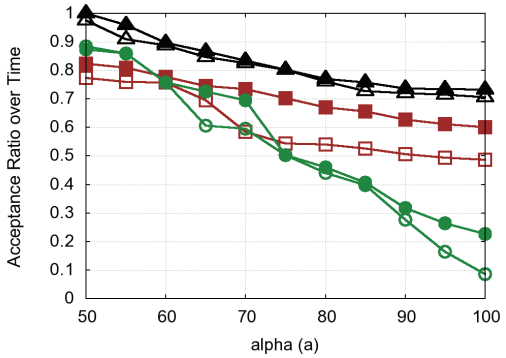

Fig. 4. Acceptance ratio against α.

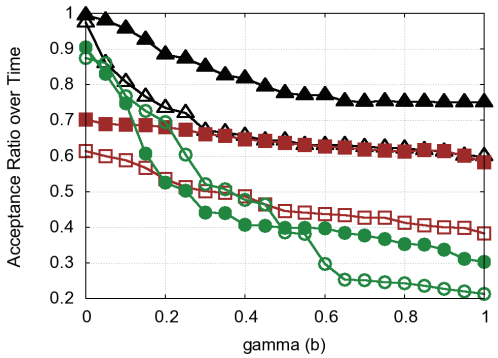

Fig. 5. Acceptance ratio against γ.

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 for the hybrid policy with increase in γ as shown in Figures 3 and 5. Although, the profit for the proactive policy increases with γ, it is still outperformed by the hybrid policy for the range of the simulation parameters.

It should be noted that as α increases, the business profit initially increases, and then starts to go down. The average profit depends on the number of virtual networks admitted and the number of subsequent failures. As we increase α, we have more resources to admit new virtual networks, but less resources for survivability. So as alpha increases, the profit gained due to new requests is gradually lost due to failure penalties. In our experiments, alpha = 80% was the threshold point after which the penalty due to failures dominated the profit from new virtual network requests.

On the other hand, as α increases, the acceptance ratio gradually decreases, as shown in Figure 4. In our experiments, the acceptance ratio is calculated by considering both the number of requests admitted to the systems, and the number of failed virtual networks. If an accepted virtual link is embedded on a substrate link that later fails, we consider that virtual network as a failed request. So as α increases, the algorithms have less resources for tolerating failures, and this leads to lower acceptance ratio.

2) Responsiveness to Failures: The hybrid policy has faster reaction time to failures than its counterparts. In Figure 7, 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 flows of the affected virtual links which results in faster response time and ultimately lower penalty for the InP.

3) Bandwidth Efficiency: The hybrid policy is bandwidth efficient. The proactive policy pre-reserves additional bandwidth for each virtual link during the instantiation phase. In turn, 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 6.

4) Trade-off between Survivability and Bandwidth Efficiency: In our experiments, we measure survivability through business profit, since lower survivability leads to higher penalty, and hence lower profit. We measured profit against α, and backup bandwidth usage against the rate of failures. As α increases, we have less resources for survivability. On the other hand, as γ increases, we have more failures compared to virtual network arrivals. Our results indicate that with more stringent failure scenarios (higher values of α and γ), our proposed algorithms achieve higher business profit, and steady increase in backup bandwidth usage, compared to baseline heuristics.

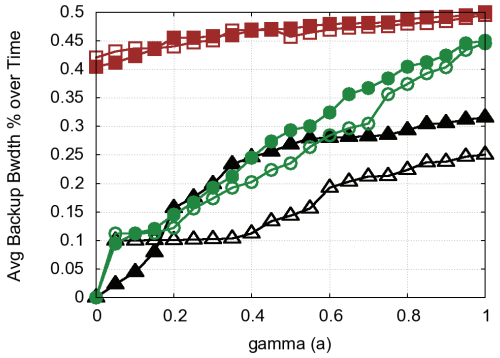

Fig. 6. Backup resource usage against γ.

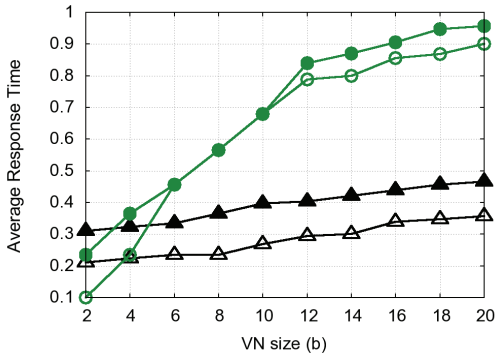

Fig. 7. Response time against VN size.

E. Execution Time

For the node embedding phase, we use existing algorithms from [14], [29]. The run-time of these algorithms have been reported there. The number of variables and constraints in each linear program for SVNE depends on the number of virtual links and the number of paths for each virtual link. Although there can be an exponential number of paths for each virtual link, we tackle this complexity by using the k-shortest path algorithm for path selection.

Let us analyze the time complexity of the proactive policy. The linear programs in LPHPP can be solved in time $\mathcal{O}(k|E^V|)^{3.5}L^2 \ln L \ln\ln L)$, where k is the number of paths selected for each virtual link, and L denotes the desired input precision in terms of the number of bits required to specify inputs to the linear program [34]. So the total time complexity of LPHPP is also $\mathcal{O}(k|E^V|)^{3.5}L^2 \ln L \ln\ln L)$.

For the hybrid policy, HYBRID_LP_LE can be solved in $\mathcal{O}(k|E^V|)^{3.5}L^2 \ln L \ln\ln L)$ time, where k is the number of paths selected for each virtual link. On the other hand HYBRID_LP_BDO takes $\mathcal{O}(k(k+1)|E^V|)^{3.5}L^2 \ln L \ln\ln L)$ time, where k is the number of paths selected for each virtual link, and for each detour. So the time complexity of HRP for each event arrival is dominated by the time complexity of HYBRID_LP_BDO. It should be noted that the time complexity of all our algorithms are independent of the size of the substrate network.

We also measured the actual run-times for each policy. On average, to solve SVNE, the blind policy took 32.15 ms, whereas the proactive and hybrid policies took 5.613 ms, and 3.834 ms respectively. Note that these execution times depend on the linear programming solver (glpk), and the hardware platform specified in section VII-A.

F. Performance on Specific VN Topologies

Up until now we have focused on arbitrary VN request topologies in our evaluations. However, some classes of topologies are naturally expected to be more prevalent than others due to their use in popular applications. For example, hub-and-spoke topologies are commonly used to connect distributed sites to a centralized server, e.g., in content distribution networks. Virtual Private Networks (VPN), which are virtual networks with only bandwidth constraints, usually adhere to standard topologies like hub-and-spoke and mesh [35]. In this section, we compare the performance of the proposed policies on these two special topologies.

TABLE IV
COMPARATIVE PERFORMANCE ON HUB-AND-SPOKE TOPOLOGIES

	Business Profit	Acceptance Ratio	Backup Usage
SVNE-G-H	0.756	0.642	0.459
SVNE-D-H	0.813	0.717	0.395
SVNE-G-P	0.662	0.576	0.835
SVNE-D-P	0.525	0.496	0.887
SVNE-G-B	0.372	0.412	0.695
SVNE-D-B	0.441	0.324	0.760

TABLE V
COMPARATIVE PERFORMANCE ON MESH TOPOLOGIES

	Business Profit	Acceptance Ratio	Backup Usage
SVNE-G-H	0.743	0.675	0.489
SVNE-D-H	0.876	0.777	0.391
SVNE-G-P	0.608	0.580	0.809
SVNE-D-P	0.598	0.501	0.882
SVNE-G-B	0.333	0.417	0.699
SVNE-D-B	0.448	0.321	0.708

1) Hub-and-Spoke Topologies: We have used similar simulation settings for this set of experiments while ensuring hub-and-spoke topologies in the VN requests instead of random graphs. Table IV summarizes the results of the compared algorithms for the five performance metrics. The results presented here are for an arrival rate of 4 VNs per 100 time units, and we present all values after standard deviations of their successive samples become negligible. The algorithms used for this experiment are the exact same ones without any topology-specific modifications. As seen in Table IV, relative performance of the compared algorithms are unchanged for hub-and-spoke topologies. Careful readers will notice that related observations for random graph requests also hold true in this case.

2) Mesh Topologies: Mesh topologies can be considered to be at the opposite end of the spectrum of specific topologies. In this case, we again use similar experimental settings and make sure that the VN requests form full mesh topologies. Simulation results in steady states are summarized in Table V for similar experimental conditions. The algorithms used for this experiment are also without any topology-specific

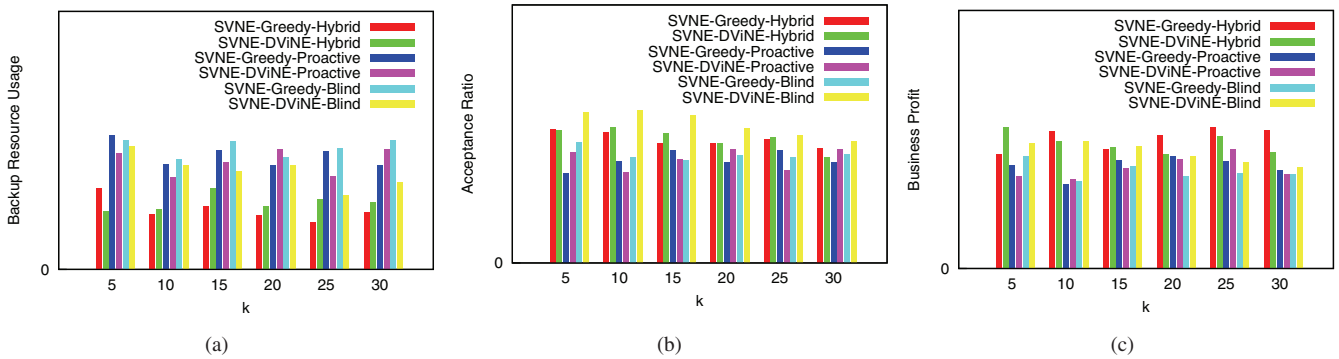

Fig. 8. Effect of k (number of allowed paths) on SVNE policies in terms of: (a) backup resource usage; (b) acceptance ratio; and (c) business profit.

modifications. The natural dense formation of mesh topologies require more resources than substrates can usually provide. As a result, the mesh topologies can lead to degraded performance in some cases. However, relative performance of the compared algorithms are mostly unchanged.

3) Effect of k (Number of Paths Allowed): We also evaluate our performance metrics against k (Figures 8(c), 8(b), 8(a)), which specifies the size of the path-sets for primary and detour flows in our path-flow based formulations. The results indicate the superior performance of the hybrid policy against the baseline policies. However there is some variability among the performance metrics for different values of k, which could suggest that a SVNE solution that continuously updates k in order to improve performance, might be better than a static solution that always uses the same value for k. This might also point towards iterative approaches using primal dual or column generation approaches.

In this set of experiments, we vary the value of k between 5, 10, 15, 20, 25, and 30. We observe a similar trend in performance against k, since the hybrid policy exhibits better performance compared to the baseline policies. For business profit, we observe that the hybrid policy with DViNE has highest profit for value $k = 5$, whereas, for the other values of k, the hybrid policy with greedy node embedding has maximum profit. For the previous set of experiments, we did not observe any significant variation in performance due to the selected node embedding heuristic. This implies that the performance metrics against k are affected by the selected node embedding mechanism. However the acceptance ratio against k experiments do not exhibit any variation due to the selected node embedding heuristic.

VIII. Conclusion and Future Work

In this paper, we have addressed the important aspect of adding survivability to network virtualization in-order to ensure seamless operation of the virtual networks embedded on top of an InP in the presence of failures. In this regard, we have formulated the SVNE problem to incorporate substrate failures in the virtual network embedding problem. We have also proposed baseline policy solutions and 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 driven optimization. Moreover, our proposed heuristics can be extended to deal with multiple link failures, and subsequently combined with a node migration strategy [5] to solve the single substrate node failure problem. We have shown detailed formulations of our proposed SVNE policies and derived efficient heuristics using optimization techniques. We also performed evaluations to demonstrate the validity and importance of our contributions.

There are many possible research directions that can be directly pursued from our current work. 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 frameworks. Our proposed solutions to SVNE are static in the sense that at any given solution instance we have fixed values for the $\alpha : \beta$ proportions and k. However the revenue of the InP might depend on complex combinations of these parameters which points towards dynamic solutions to SVNE using control theory or statistical machine learning techniques.

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, 2005.
[2] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.* , vol. 54, np. 5, pp. 862–876, 2010.
[3] N. Feamster, L. Gao, and J. Rexford, "How to lease the Internet in your spare time," *2007 ACM SIGCOMM CCR*, pp. 61–64.
[4] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," in *Proc. 2002 ACM SIGCOMM Workshop on Internet Measurment*.
[5] 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*, pp. 231–242.
[6] M. R. Rahman, I. Aib, and R. Boutaba, "Survivable virtual network embedding," in *Proc. 2010 IFIP Netw.*, pp. 40–52.
[7] D. Andersen, "Theoretical approaches to node assignment."
[8] J. Lu and J. Turner, "Efficient mapping of virtual networks onto a shared substrate," Washington University, Tech. Rep. WUCSE-2006-35, 2006.
[9] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. 2006 IEEE INFOCOM*, pp. 1–12.
[10] J. Fan and M. Ammar, "Dynamic topology configuration in service overlay networks—a study of reconfiguration policies," in *Proc. 2006 IEEE INFOCOM*, pp. 1–12.

[11] A. Capone, J. Elias, and F. Martignon, "Models and algorithms for the design of service overlay networks," *IEEE Trans. Netw. and Service Management*, vol. 5, no. 3, pp. 143–156, 2008.

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

[13] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. 2009 IEEE INFOCOM*, pp. 783–791.

[14] 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, 2011.

[15] I. Houidi, W. Louati, and D. Zeghlache, "A distributed virtual network mapping algorithm," in *Proc. 2008 IEEE ICC*, pp. 5634–5640.

[16] M. Chowdhury, F. Samuel, and R. Boutaba, "Polyvine: policy-based virtual network embedding across multiple domains," in *Proc. 2010 ACM SIGCOMM VISA*, pp. 49–56.

[17] I. Houidi, W. Louati, W. Ben Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Comput. Netw.* , vol. 55, no. 4, pp. 1011–1023, 2011.

[18] Y. Xin, I. Baldine, A. Mandal, C. Heermann, J. Chase, and A. Yumerefendi, "Embedding virtual topologies in networked clouds," in *Proc. 2011 International Conference on Future Internet Technologies*, pp. 26–29.

[19] A. Leivadeas, C. Papagianni, and S. Papavassiliou, "Efficient resource mapping framework over networked clouds via iterated local search based request partitioning," *IEEE Trans. Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1–1, 2012.

[20] R. Ricci, C. Alfeld, and J. Lepreau, "A solver for the network testbed mapping problem," *Proc. 2003 ACM SIGCOMM CCR*, pp. 65–81.

[21] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau, "Large-scale virtualization in the emulab network testbed," in *Proc. 2008 USENIX ATC*, pp. 113–128.

[22] W. Lau and S. Jha, "Failure-oriented path restoration algorithm for survivable networks," *IEEE Trans. Netw. and Service Management*, vol. 1, no. 1, pp. 11–20, 2004.

[23] Q. Zheng and K. G. Shin, "Fault-tolerant real-time communication in distributed computing systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 9, no. 5, pp. 470–480, 1988.

[24] K. Thulasiraman, M. S. Javed, and G. L. Xue, "Circuits/cutsets duality and a unified algorithmic framework for survivable logical topology design in IP-over-WDM optical networks," *Proc. 2009 IEEE INFOCOM*, pp. 1026–1034.

[25] M. Kurant and P. Thiran, "Survivable MApping Algorithm by Ring Trimming (SMART) for large IP-over-WDM networks," in *Proc. 2004 Broadband Netw.*, pp. 44–53.

[26] M. Kurant and P. Thiran, "Survivable routing of mesh topologies in IP-over-WDM networks by recursive graph contraction," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 5, 2007, pp. 922–933, 2007.

[27] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, 2nd edition. Dover Publications, 1998.

[28] K. Lee and E. Modiano, "Cross-layer survivability in wdm-based networks," in *Proc. 2009 IEEE INFOCOM*, pp. 1017–1025.

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

[30] A. Todimala and B. Ramamurthy, "Approximation algorithms for survivable multi-commodity flow problems with applications to network design," in *Proc. 2006 IEEE INFOCOM*, pp. 1–12.

[31] D. M. Topkis, "A k shortest path algorithm for adaptive routing in communications networks," *IEEE Trans. Commun.*, vol. 36, no. 7, pp. 855–859, 1988.

[32] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[33] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational IP backbone network," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 749–762, 2008.

[34] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. 1984 ACM STOC*, pp. 302–311.

[35] S. Raghunath, K. K. Ramakrishnan, S. Kalyanaraman, and C. Chase, "Measurement based characterization and provisioning of IP VPNs," in *Proc. 2004 ACM SIGCOMM IMC*, pp. 342–355.

Muntasir Raihan Rahman received his B. Sc. degree in computer science and engineering from Bangladesh University of Engineering and Technology in 2006, and the M. Math. degree in computer science from the University of Waterloo in 2010, and is currently pursuing the Ph.D. degree in computer science at the University of Illinois at Urbana-Champaign. His research interests include distributed storage, cloud infrastructure, and distributed computing theory.

Raouf Boutaba received the M. Sc. and Ph.D. degrees in computer science from the University Pierre & Marie Curie, Paris, in 1990 and 1994, respectively. He is currently a Professor of computer science at the University of Waterloo, Canada and a Distinguished Visiting Professor at the Pohang University of Science and Technology (POSTECH), Korea. His research interests include control and management of networks and distributed systems. He has received several best paper awards and other recognitions such as the Premier's Research Excellence Award, the IEEE Hal Sobol Award in 2007, the Fred W. Ellersick Prize in 2008, the Joe LociCero Award and the Dan Stokesbury Award in 2009. He is a fellow of the IEEE.