Shared Backup Network Provision for Virtual Network Embedding

Tao Guo, Ning Wang, Klaus Moessner, and Rahim Tafazolli
Centre for Communication Systems Research, University of Surrey
Guildford, GU2 7XH, United Kingdom
Email: {t.guo, n.wang, k.moessner, r.tafazolli, }@surrey.ac.uk

***Abstract*—Network virtualization has been recognized as a promising solution to enable the rapid deployment of customized services by building multiple Virtual Networks (VNs) on a shared substrate network. Whereas various VN embedding schemes have been proposed to allocate the substrate resources to each VN requests, little work has been done to provide backup mechanisms in case of substrate network failures. In a virtualized infrastructure, a single substrate failure will affect all the VNs sharing that resource. Provisioning a dedicated backup network for each VN is not efficient in terms of substrate resource utilization. In this paper, we investigate the problem of shared backup network provision for VN embedding and propose two schemes: shared on-demand and shared proactive backup schemes. Simulations experiments show that both proposed schemes make better utilization of substrate resources than the dedicated backup scheme without sharing, while each of them has its own advantages.**

I. Introduction

The concept of network virtualization has recently attracted significant attentions as a promising solution to provide versatile customized services over a shared substrate network [1]. Multiple service providers can lease the sliced physical resources from one or more infrastructure providers to form their own Virtual Networks (VNs) without significant investment on hardware. Each VN is implemented by connecting virtual nodes with virtual links. Each VN request may arrive and leave online. From the point of view of a service provider, enough physical resources should be secured to run customized services, such as node processing resources and link bandwidth. From the point of view of an infrastructure provider, the cost of hosting an individual virtual network should be minimized in order to maximize the long-term revenue by accepting more VN requests. The efficient embedding/mapping of virtual networks to physical networks provide an immediate challenge toward future virtualized Internet.

While a number of schemes have been recently proposed for resource allocation in network virtualization environments [2]–[6], they did not provide the backup mechanisms to combat potential node/link failures except [6]. Given a failure-prone Internet due to various disruption causes such as maintenance, fiber cut, policy change and misconfiguration, the substrate nodes/links may not operate properly all the time. In case of a substrate node/link failure, all the virtual networks using that node/link will be affected. The infrastructure providers may incur economic penalty due to the breaking of Service Level Agreements (SLAs) with the service providers. In [6], a reactive backup mechanism is proposed for virtual network embedding to protect against single substrate link failures. Before any VN request arrives, the bandwidth of a substrate link is divided into two parts: one reserved for primary flows and the other reserved for backup flows. When a VN request arrives, it will be mapped to the residual primary bandwidth of the substrate links. In the event of a substrate link failure, a reactive online optimization mechanism is invoked to allocate the resources from the available backup bandwidth of the substrate links for rerouting the affected primary traffic. With the increase of the transmission capabilities, any failure may cause a vast amount of data loss and it may not be possible to activate the restoration mechanisms after the failure. Furthermore, while the effect of different bandwidth divisions for the substrate links has been evaluated in [6], it did not provide a mechanism to find the optimal division solution.

In this paper, we investigate the problem of allocating the protection resources for virtual networks before a failure happens. As a first step, we consider the solutions to protect against any single substrate link failure since multiple concurrent failures rarely occur in real world. In principle, a dedicated backup network can be allocated to each virtual network. However, it is not resource efficient from the perspective of an infrastructure provider. It is expected that the backup resources can be reused by other virtual networks to make room for accepting more VN requests. We propose two shared backup network provision mechanisms for virtual network embedding: Shared On-Demand approach (SOD_BK) and Shared Proactive approach (SP_BK). In the first approach, backup resources are allocated upon the arrival of each VN request. In the second approach, backup resources are pre-allocated during the configuration phase before any VN request arrives to protect against any potential single link failure. In either approach, the backup paths have been configured with capable bandwidth when the VN request is mapped.

The rest of the paper is organized as follows. Section II summarizes the related work. Section III formally defines the problem of backup network provision for virtual network embedding. Section IV presents the shared backup network design approaches. In Section V, the performance of the

This work has been supported by the Virtual Centre of Excellence in Mobile & Personal Communications (Mobile VCE) Core 5 Program.

proposed schemes is evaluated in a comparison to a baseline backup scheme without sharing and the original virtual network embedding scheme without backup. Finally, Section VI summarizes the key points and draws a conclusion.

II. Related Work

The problem of virtual network embedding itself is already extremely challenging and is *NP*-hard even in the offline case. Heuristic based approaches have been proposed in [2] to perform a two-stage embedding approach by selecting the capable substrate nodes firstly and connecting the selected nodes based on the shortest path routing thereafter. To make better resource utilization and improve acceptance ratio of VN requests, a multi-commodity flow approach has been presented in [3] to implement link mapping. By mapping a virtual link to multiple physical paths, the bandwidth constraint of the virtual link can be satisfied as long as the available bandwidth sum of the physical paths is no less than the requested bandwidth. Chowdhury *et al.* [4] has proposed a mathematical programming based scheme to coordinate node and link mapping. A backtracking algorithm based on subgraph isomorphism detection has been proposed in [5] to map nodes and links during the same stage.

Survivable network design has been a fruitful area for optical and Multiprotocol Label Switching (MPLS) networks [7]–[9]. However, the problems studied there are an offline version and/or assume the traffic demand matrix has been given in advance. Under the unpredicted arrival and demand pattern of VN requests, the problem of provisioning the backup networks becomes very challenging. The complexities of the available solutions for the survivable network design are typically prohibitive for online invoking, especially when practical considerations such as the number limit and the hop-count limit of the bypass paths are taken into account.

III. Problem Formulation

A. Substrate Network

The substrate network is modelled as an undirected graph $G^S = (N^S, E^S)$, where N^S is the set of substrate nodes and E^S is the set of substrate links. Each substrate node $x \in N^S$ is associated with a CPU computing capability $CPU(x)$ and a location $LOC(x)$. Each substrate link $s \in E^S$ is associated with a bandwidth transport capability $B(s)$. Note that both the CPU and bandwidth capabilities are allocated by the infrastructure providers based on their own policies and agreements with the service providers, and thus, they do not have to be equivalent to the capabilities of the physical routers and cables.

B. VN Request

Similar to the substrate network, a VN request is also modelled as an undirected graph $G^V = (N^V, E^V)$, where N^V is the set of virtual nodes and E^V is the set of virtual links. Each virtual node $y \in N^V$ has a CPU computing demand $CPU(y)$ and a preferred location $LOC(y)$. Each virtual link $v \in E^V$ has a bandwidth transport demand $B(v)$ and a delay constraint $DELAY(v)$. In addition, each VN request defines a distance constraint $DIST^V$ for the virtual nodes, which specifies how deviated y can be placed from its preferred location.

C. VN Embedding

The process of original VN embedding is to find a mapping M from G^V to G^S such that all the demands and constraints in G^V are satisfied. In particular, a virtual node is mapped to a substrate node in a one-to-one manner such that both the CPU demand of the virtual node and its location constraint can be satisfied. A virtual link is mapped to a set of substrate paths such that the total available bandwidth of the substrate paths can support the bandwidth demand of the virtual link and each individual substrate path meets the Quality of Service (QoS) constraint of the virtual link. Let $P_a(x_1, x_2)$ denotes the set of all possible substrate paths from substrate node x_1 to x_2, $R_N(x)$ denotes the residual CPU resource at substrate node x and $R_E(P \subseteq P_a(x_1, x_2))$ denotes the total residual bandwidth of a P_a's subset P from x_1 to x_2. This process can be formulated as follows:

Node mapping $M^N(\forall y \in N^V)$:

$$M^N(y) \in N^S$$
$$M^N(y) = M^N(y'), \quad iff \quad y = y'$$

Link mapping $M^E(\forall v = (y_1, y_2) \in E^V)$:

$$M^E(v = (y_1, y_2)) \subseteq P_a(M^N(y_1), M^N(y_2))$$

Subject to:

$$CPU(y) \leq R_N(M^N(y))$$
$$dist\big(LOC(y), LOC(M^N(y))\big) \leq DIST^V$$
$$B(v) \leq R_E(M^E(v))$$
$$delay(p) \leq DELAY(v), \quad \forall p \in M^E(v)$$

D. VN Backup

In addition to mapping the primary flows of a VN request, the backup paths and bandwidth need to be allocated for the VN request such that in case of any single substrate link failure it is guaranteed that the affected traffic can be routed via the backup paths immediately without disrupting the current services. A VN request can be admitted only if its CPU demands, primary flow demands and restoration flow demands are all satisfied.

E. Objectives

From the point of view of infrastructure providers, they hope to maximize their revenue in the long run. The revenue here corresponds to the economic profit of accepting VN requests. Assuming each VN request has similar location and QoS constraints, the revenue of accepting a VN request is mainly determined by the total demands of the VN. We define the revenue of accepting a VN request as:

$$Rev(G^V) = T(G^V)\big(\sum_{v \in E^V} B(v) + w \sum_{x \in N^V} CPU(x)\big) \quad (1)$$

where $T(G^V)$ is the duration of the VN request and w is the weight to determine the relative importance of the CPU and bandwidth resource. Note that the allocated backup resource does not produce profit directly. However, this requirement is normally implicitly included in the SLAs between the infrastructure provider and the service provider. Therefore, it is left for the infrastructure provider to decide how to provide effective protection mechanisms.

IV. Shared Backup Network Provision

In this section, we present two shared backup network provision approaches for VN embedding. To protect against a link failure, the restoration can be implemented at link or path level. In link restoration, each link with primary flows is protected by a set of pre-configured bypass paths. Upon a failure of a link, traffic on this link will be locally routed over the bypass paths. In path restoration, each primary path will be protected by a disjoint backup path from source to destination. Upon a failure of a link, error information will be sent to the source and the source will activate the backup path. In context of VN embedding, each substrate link may be concurrently shared by a number of VNs. Path restoration may cause large error information propagation overhead and latency. In this paper, link restoration is adopted for fast recovery. Similar to [6], a set of possible bypass paths is pre-computed for each substrate link. The number of paths and QoS constraint can be forced when pre-computing the set of candidate bypass paths.

A. Shared On-Demand Approach (SOD_BK)

In this approach, we allocate the substrate bandwidth to both the primary flows of a VN request for normal working and the restoration flows for fast recovery upon the arrival of the VN request. For the primary flows of a VN request, no bandwidth sharing is possible since they need to be operating all the time. For the restoration flows to protect against different substrate link failures, they can share the bandwidth to minimize the resource utilization for backup purpose. When a VN request arrives, node embedding phase is first implemented to figure out the placement of virtual nodes. For a virtual link $v = (y_1, y_2) \in E^V$ connecting two virtual nodes y_1 and y_2, $M^N(y_1) = x_1 \in N^S$ and $M^N(y_2) = x_2 \in N^S$. Let $P(v) \subseteq P_a(x_1, x_2)$ denote the set of pre-selected QoS constrained candidate paths for the virtual link v, and let $R(s)$ denote the set of candidate bypass paths for a substrate link $s \in E^S$. We use $I_s(p)$ as an indicator to indicate whether or not a substrate link s is on a substrate path p. $I_s(p) = 1$, if $s \in p$, 0 otherwise. Let R_s denote the residual bandwidth of a substrate link s, Z_s denote the already reserved backup bandwidth on s, and Y_s^f denote the bandwidth requirements of the restoration flows on s in case of another substrate link f failure. The linear program formulation is presented as follows:

Variables:

- $x_p(v)$: The primary flow on the substrate path p for the virtual link v.
- $y_r(f)$: The restoration flow on the substrate path r in case of the substrate link f failure.
- z_s: The total bandwidth need to be reserved on the substrate link s for the restoration flows.

Objective:

Minimize:

$$\sum_{s \in E^S} \frac{1}{R_s + \delta} \Big(\sum_{v \in E^V} \sum_{p \in P(v)} I_s(p) x_p(v) + z_s \Big) \tag{2}$$

Subject to:

Capacity constraint:

$$\sum_{v \in E^V} \sum_{p \in P(v)} I_s(p) x_p(v) + z_s \leq R_s, \quad \forall s \in E^S \tag{3}$$

Primary flow constraint:

$$\sum_{p \in P(v)} x_p(v) = B(v), \quad \forall v \in E^V \tag{4}$$

Restoration flow constraint:

$$\sum_{r \in R(f)} y_r(f) = \sum_{v \in E^V} \sum_{p \in P(v)} I_f(p) x_p(v), \quad \forall s \in E^S \tag{5}$$

Restoration bandwidth constraint:

$$\sum_{r \in R(f)} I_s(r) y_r(f) \leq z_s + Z_s - Y_s(f), \quad \forall s \in E^S, \forall f \in E^S \tag{6}$$

Domain Constraints:

$$\begin{aligned} & x_p(v) \geq 0, \quad \forall v \in E^V, \forall p \in P(v) \\ & y_r(f) \geq 0, \quad \forall f \in E^S, \forall r \in R(f) \\ & z_s \geq 0, \quad \forall s \in E^S \end{aligned} \tag{7}$$

The objective function (2) tries to minimize the total resource need to be reserved for both primary flows and restoration flows as well as balance the load to avoid a bottleneck link separating the substrate topology. $\delta \to 0$ is a small positive constant to avoid division by zero. Constraint (3) bounds the total reserved bandwidth to be at most the residual substrate link capability. Constraint (4) represents that the requirements of the primary flows of all the virtual links should be satisfied. Constraint (5) represents that in case of a substrate link failure, all the affected primary traffic on that link should be able to reroute through the bypass paths. Constraint (6) ensures that the accumulated reserved bandwidth for backup purpose including the previously reserved bandwidth before the VN arrives should be able to support all the restoration traffic of all the VNs.

After the VN request is embedded, the residual CPU, total residual bandwidth, total reserved backup bandwidth and the restoration flow information need to be updated. We use t and $t+1$ to denote the time point before a VN request arrives and after it is admitted, respectively. For each substrate link $s \in E^S$, $R_s(t+1) = R_s(t) - \sum_{v \in E^V} \sum_{p \in P(v)} I_s(p) x_p(v) - z_s$, $Z_s(t+1) = Z_s(t) + z_s$, and $Y_s^f(t+1) = Y_s^f(t) + \sum_{r \in R(f)} I_s(r) y_r(f)$. At the end of the lifetime of the VN request, the corresponding resource

allocated for the VN should be released. Similarly, we use t and $t+1$ to denote the time point before and after a VN request departures, respectively. For each substrate link $s \in E^S$, we first update $Y_s^f(t+1) = Y_s^f(t) - \sum_{r \in R(f)} I_s(r) y_r(f)$, then we find $Z_s(t+1) = \max_{f \in E^S}(Y_s^f(t+1))$, and finally, $R_s(t+1) = R_s(t) + \sum_{v \in E^V} \sum_{p \in P(v)} I_s(p) x_p(v) + Z_s(t) - Z_s(t+1)$.

B. Shared Proactive Approach (SP_BK)

In this approach, we pre-allocate the backup bandwidth for each substrate link during network pre-configuration phase. As it is impossible to predict the arrival and demand pattern of the future VN requests, the allocated bandwidth should be able to protect the maximum allowed primary flows on each substrate link. The linear program formulation is presented as follows:

Variables:

- α_s: The percentage of the bandwidth allocated on a substrate link s for mapping primary flows.
- $y_r(f)$: The restoration flow on the substrate path r in case of the substrate link f failure.

Objective:

Maximize:

$$\sum_{s \in E^S} \alpha_s B(s) \tag{8}$$

Subject to:

Restoration flow constraint:

$$\sum_{r \in R(f)} y_r(f) = \alpha_f B(f), \quad \forall f \in E^S \tag{9}$$

Restoration bandwidth constraint:

$$\sum_{r \in R(f)} I_s(r) y_r(f) \leq (1 - \alpha_s) B(s), \quad \forall s \in E^S, \forall f \in E^S \tag{10}$$

Domain Constraints:

$$\begin{aligned} & y_r(f) \geq 0, \quad \forall f \in E^S, \forall r \in R(f) \\ & \alpha_s \geq 0, \quad \forall s \in E^S \end{aligned} \tag{11}$$

The objective function (8) tries to maximize the total protected bandwidth of the substrate network for mapping primary flows. Constraint (9) represents that the bandwidth allocated on each substrate link for mapping primary flows should be fully protected via the restoration flows over its bypass paths. Constraint (10) ensures that for any substrate link failure, the bandwidth demand of the restoration flows can be satisfied by the pre-allocated backup bandwidth. This pre-allocation approach only needs to be implemented once before any VN request arrives, and thus can significantly reduce the computing load during VN embedding. Furthermore, any proposed VN embedding schemes can be directly applied after the pre-allocation phase.

V. Performance Evaluation

In this section, the performance of the proposed schemes is evaluated in a comparison to a baseline on-demand backup scheme without sharing (OD_BK) and the original VN embedding scheme without backup (NO_BK). For simplicity, the greedy heuristic in [2] [3] is used for node embedding.

TABLE I
Average Simulation Results

Schemes	CPU Demands: [0, 5]			CPU Demands: [0, 20]		
	P_{ac}	Rev	P_{bk}	P_{ac}	Rev	P_{bk}
NO_BK	0.85	5026	0	0.70	5618	0
OD_BK	0.61	3097	0.36	0.56	4070	0.35
SOD_BK	0.72	4054	0.18	0.66	5021	0.17
SP_BK	0.71	3846	0.21	0.64	4848	0.21

A. Simulation Environment

A discrete-event simulator is implemented to evaluate the performance of all the four schemes. The simulation setup is similar to previous work [3] [4] [6]. A substrate network topology with 50 nodes and around 250 links is generated via the GT-ITM tool [10], the scale of which corresponds a medium-sized Internet domain. All the nodes are randomly distributed in a 25 $\times$ 25 grid.The substrate node CPU and link bandwidth capacities follow a uniform distribution between 50 and 100 resource units. The VN requests arrive in a Poisson manner with an average rate of 5 VNs per 100 time units. The duration of the requests follows an exponential distribution with an average lifetime of 1000 time units. In a VN request, the number of virtual nodes follows a uniform distribution between 2 and 10. Each VN node has a preferred location uniformly distributed in the same grid as the substrate topology and the distance constraint $DIST^v$ is assumed to be 5 for all the VN requests. The average connectivity ratio between a pair of virtual nodes is 0.5.The delay constraints $DELAY^v$ for the virtual links are included during $P(v)$ pre-selection. The bandwidth demands of the virtual links are uniformly distributed between 0 and 30 resource units. The CPU demands of the virtual nodes are varied in the simulations. The simulation experiments are run for around 50000 time units. Without loss of generality, we use the k-shortest paths ($k = 5$) as the set of QoS constrained candidate paths. And a similar procedure is used to derive the set of candidate bypass paths for each substrate link. The GLPK tool [11] is used to solve the linear programs in this paper.

B. Simulation Results

Three metrics are defined for evaluation purpose: VN request acceptance ratio P_{ac}, generated revenue Rev and backup bandwidth occupation ratio P_{bk}. P_{ac} denotes the ratio of the accepted VN requests over the total VN requests that have arrived. Since each VN request may have different resource demands, we also evaluate the generated revenue from the infrastructure provider's perspective. Without loss of generality, we assume $w = 1$ in (1) which indicates that the importance of the CPU and bandwidth resource is similar. P_{bk} is defined as the ratio of the total reserved bandwidth for backup purpose over the total bandwidth capacities. In the first experiment, the CPU demands of the virtual nodes are uniformly selected in [0,5]. In the second one, they are uniformly selected in [0,20]. Table I lists the average simulation results over the whole simulation durations. It is unsurprising that NO_BK scheme

accepts the most requests and generates the most revenue. However, the revenue may be seriously compromised given a failure-prone Internet. The infrastructure provider may have to pay a costly penalty for the lack of backup mechanisms. By enabling backup bandwidth sharing among VNs, both proposed schemes can reduce the allocated bandwidth for backup purpose, and thus, admit more VN requests and generate more revenue than the scheme without sharing. On average, SOD_BK scheme and SP_BK scheme generate 31% and 24% more revenue than OD_BK scheme with the CPU demands in [0,5], respectively. Whereas SOD_BK scheme performs slightly better than SP_BK scheme, SP_BK scheme removes the requirement to find a backup network each time a VN request arrives and thus, reduces the online calculation time. When the CPU demands of the VN requests increase, all the schemes admit less requests since the node constraints become more important and many requests are rejected due to the difficulty of finding substrate nodes with enough CPU resource. However, the generated revenue may be increased as more CPU demands of each VN request bring the infrastructure provider more node revenue. It is worth mentioning that with more advanced node embedding procedure such as [4] the performance of all the schemes will be improved.

To show the performance of each scheme in the long run, we also plot the performance metrics Rev and P_{bk} against time with the CPU demands in [0,20]. Both metrics are averaged over a time window T_w during the simulations ($T_w = 500$ time units in this work). As shown in Fig.1, the revenue is increased with the arrivals of the VN requests at the beginning. When the admitted requests have occupied the substrate resource, more requests will be rejected. Fig.2 shows the the backup bandwidth utilizations over time. Both proposed schemes significantly reduce the bandwidth requirements for backup purpose when the substrate network operates in a saturated state, in which more requests have to be rejected. However, SP_BK scheme needs to maintain the backup bandwidth regardless of the arrivals of the VN requests. It may not be effective if the demands of the VN requests are not heavy compared to the substrate resource capacities.

VI. Conclusion

To enable a reliable virtualized infrastructure for future Internet, effective backup mechanisms are required to combat substrate network failures. In this paper, we propose two shared backup network provision schemes for virtual network embedding: shared on-demand approach and shared proactive approach. By sharing the bandwidth used by the restoration flows from different VNs, the required backup bandwidth can be greatly reduced and more substrate resource can be left for admitting future VN requests. Whereas the first approach needs to be implemented during each VN embedding process, the second approach pre-allocates the bandwidth for backup purpose offline before any VN requests arrives. However, the second approach has to always maintain the backup bandwidth regardless of the VN requests, which may not be effective at low VN request loads.

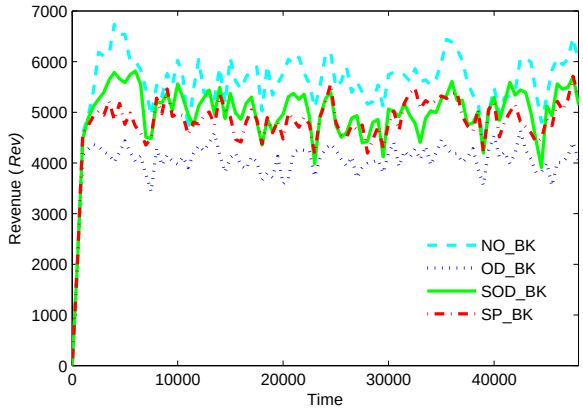

Fig. 1. Revenue over time (CPU demands: $[0, 20]$)

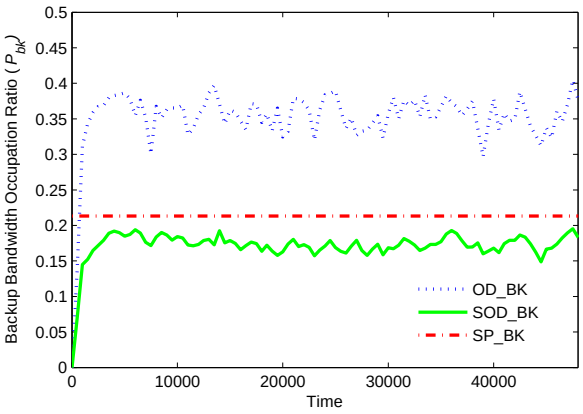

Fig. 2. Backup bandwidth occupation ratio over time (CPU demands: $[0, 20]$)

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] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. IEEE INFOCOM'06*, 2006.

[3] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM CCR*, vol. 38, no. 2, pp. 17–29, 2008.

[4] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE INFOCOM'09*, 2009.

[5] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. ACM SIGCOMM VISA'09*, 2009.

[6] M. R. Rahman, I. Aib, and R. Boutaba, "Survivable virtual network embedding," in *Proc. NETWORKING'10*, 2010.

[7] W. D. Grover, *Mesh-based Survivable Networks: Options and Strategies for Optical, MPLS, and ATM Networking*. Prentice Hall PTR, 2003.

[8] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating optimal spare capacity allocation by successive survivable routing," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 198–211, Dec. 2005.

[9] R. Banner and A. Orda, "Designing low-capacity backup networks for fast restoration," in *Proc. IEEE INFOCOM'10*, 2010.

[10] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM'96*, 1996.

[11] GNU Linear Programming Kit: http://www.gnu.org/software/glpk/.