Contribution à une instanciation efficace et robuste des réseaux virtuels sous divers contraintes

Thèse presenté par

Shuopeng LI

Pour obtenir le grade de docteur délivré par

Université Paris 13

Spécialité: **Réseaux et technologies de l'information**

Date de soutenance: **09, Novembre, 2017**

jury:

Pr. Bernard Cousin	IRISA	Rapporteur
Pr. Nadjib Ait Saadi	ESIEE	Rapporteur
Pr. Patrick Siarry	Université Paris-est Créteil	Membre du jury
Dr. Mohand Yazid Saidi	Université Paris 13	Co-encadrant
Pr. Ken Chen	Université Paris 13	Directeur de thèse

Abstract

Title: Contribution to an efficient and resilient embedding of virtual networks under various constraints

Network virtualization allows to create logical or virtual networks on top of a shared physical or substrate network. The resource allocation problem is an important issue in network virtualization. It corresponds to a well known problem called virtual network embedding (VNE). VNE consists in mapping each virtual node to one substrate node and each virtual link to one or several substrate paths in a way that the objective is optimized and the constraints verified. The objective often corresponds to the optimization of the node computational resources and link bandwidth whereas the constraints generally include geographic location of nodes, CPU, bandwidth, etc. In the multi-domain context where the knowledge of routing information is incomplete, the optimization of node and link resources are difficult and often impossible to achieve. Moreover, to ensure service continuity even upon failure, VNE should cope with failures by selecting the best and resilient mappings.

In this thesis, we study the VNE resource allocation problem under different requirements. To embed a virtual network on multi-domain substrate network, we propose a joint peering and intra domain link mapping method. With reduced and limited information disclosed by the domains, our downsizing algorithm maps the intra domain and peering links in the same stage so that the resource utilization is optimized. To enhance the reliability of virtual networks, we propose a failure avoidance approach that minimizes the failure probability of virtual networks. Exact and heuristic solutions are proposed and detailed for the infinite and limited bandwidth link models. Moreover, we combine the failure avoidance with the failure protection in our novel protection-level-aware survivable VNE in order to improve the reliability. With this last approach, the protectable then the less vulnerable links are first selected for link mapping. To determine the protectable links, we propose a *maxflow* based heuristic that checks for the existence of backup paths during the primary mapping stage. In case of insufficient backup resources, the failure probability is reduced.

Key words: Network virtualization, Virtual network embedding, Multi-domain network, Network protection, Survivability, Network reliability.

Résumé

Titre: Contribution à une instanciation efficace et robuste des réseaux virtuels sous divers contraintes

La virtualisation de réseau permet de créer des réseaux logiques, dits virtuels sur un réseau physique partagé dit substrat. Pour ce faire, le problème d'allocation des ressources aux réseaux virtuels doit être résolu efficacement. Appelé VNE (Virtual Network Embedding), ce problème consiste à faire correspondre à chaque nœud virtuel un nœud substrat d'un côté, et de l'autre, à tout lien virtuel un ou plusieurs chemins substrat, de manière à optimiser un objectif tout en satisfaisant un ensemble de contraintes. Les ressources de calcul des nœuds et les ressources de bande passante des liens sont souvent optimisées dans un seul réseau substrat. Dans le contexte multi-domaine où la connaissance de l'information de routage est incomplète, l'optimisation des ressources de nœuds et de liens est difficile et souvent impossible à atteindre. Par ailleurs, pour assurer la continuité de service même après une panne, le VNE doit être réalisé de manière à faire face aux pannes.

Dans cette thèse, nous étudions le problème d'allocation de ressources (VNE) sous diverses exigences. Pour offrir la virtualisation dans le contexte de réseau substrat multi-domaines, nous proposons une méthode de mappage conjoint des liens inter-domaines et intra-domaines. Avec une information réduite et limitées annoncées par les domaines, notre méthode est capable de mapper simultanément les liens intra-domaines et les liens inter-domaines afin d'optimiser les ressources. De plus, pour améliorer la robustesse des réseaux virtuels, nous proposons un algorithme d'évitement des pannes qui minimise la probabilité de panne des réseaux virtuels. Des solutions exactes et heuristiques sont proposées et détaillées pour des liens à bande passante infinie ou limitée. En outre, nous combinons l'algorithme d'évitement des pannes avec la protection pour proposer un VNE robuste et résistant aux pannes. Avec cette nouvelle approche, les liens protégeables puis les liens les moins vulnérables sont prioritairement sélectionnés pour le mappage des liens. Pour déterminer les liens protégeables, nous proposons une heuristique qui utilise l'algorithme du *maxflow* afin de vérifier et de déterminer les liens protégeables à l'étape du mappage des liens primaires. En cas d'insuffisance de ressources pour protéger tous les liens primaires, notre approche sélectionne les liens réduisant la probabilité de panne.

Mots-clés: Virtualisation des réseaux, Mappage des réseaux virtuels, Réseaux multi-domaines, Protection des réseaux, Capacité de survie, Fiabilité des réseaux.

Table of contents

Chapter 1

Introduction

1.1 Background

With the rapid development of Internet, more and more Internet services ask for a robust and flexible network. Network Virtualization (NV) [APST05] is one of the key technologies to accommodate the novel network requirements. Network Virtualization allows to create various specific-purpose independent logical networks on top of a shared physical network. Infrastructure resources of a network are abstracted and sliced into multiple virtual networks (VN). Each virtual network works as an isolated network with its own resources and performs its own network function in an independent way. In other words, virtualizing a network consists in realizing the process of combining hardware and software resources and network functionality into a single software based virtual network.

Network Virtualization has shown a lot of benefits [BHK12]. Since the resources are shared, Network Virtualization cuts down the cost of maintaining and replacing of infrastructure. The network can be treated as a flexible pool with capacity that can be consumed and changed on demand. As a result, it speeds up the development, testing and deployment of new protocols.

Generally speaking, virtual network is a kind of overlay network. Overlay network is not a new concept [LCP+05]. Since the birth of computer network, we have seen various overlay networks. Among these overlay technologies, we can cite (1) VLAN which partitions and isolates a network at Layer 2 through VLAN tagging and (2) VPN (Virtual Private Network) which establishes a virtual point-to-point connection through the use of virtual tunneling protocols (IPSec and MPLS).

Although overlay network can be considered as an important past of network virtualization, it is different from network virtualization. Indeed, overlay network is only adapted for address space isolation (for example, isolating duplicate MAC addresses, duplicate IP

addresses, or duplicate VLAN IDs). Whereas network virtualization involves some new features, such as flexible and dynamic management of VNs, easy configuration of VNs and virtualization of network services. To provide a platform for on-line setup networks that supports multiple types of applications, network virtualization seems to be investible, specially it allows rapid configuration.

With the arrival of SDN (Software Defined Networking) [KRV+15] and NFV (Network Function Virtualization) [MSG+16], the requirements of Network Virtualization can be implemented in an easy, efficient and realistic way. The separation of control plane and data plane allows the controller to have a general view of the network. The controller can establish VNs in an on-demand and dynamic way. OpenFlow [MAB+08], which is a standard communication protocol between the control and forwarding layers of an SDN architecture, makes it possible to easily isolate networks and to configure special requirements (e.g. backup port).

1.2 Challenge

On the road to network virtualization (NV), many research challenges remain to be addressed [CB10]. Some of the key issues are isolation, resource allocation and security. Isolation enable the configuration and customization of shared virtual networks. Resource allocation solves the problem of efficient physical network resource utilization. Security issues, like authentication, are also need to be considered in Network Virtualization.

In this thesis, we focus on the problem of resource allocation for Network Virtualization. Since multiple virtual networks work on the same physical network, the basic problem consists in determining an efficient (optimal) method to allocate physical resources to the virtual networks. This problem is referred as Virtual Network Embedding (VNE) [FBTB+13]. Basically, the objective consists in increasing the revenue by allocating a maximum number of virtual networks on the physical network. Apart from the revenue, other virtual network demands may appear and should be satisfied, like VNE in multi-domain physical network, reliable VNs, survivable VNs, etc.. Accordingly, the goal of this thesis is to design efficient algorithms/frameworks for VNE such that the constraints/objectives related to VNs are satisfied/optimized. In other words, the designed algorithms/frameworks should provide efficient, flexible and dynamic resource allocation methods to improve the physical resource utilization while satisfying/optimizing various constraints/objectives (like revenue, the reliability, etc.) in different contexts (intra or inter domain physical networks).

The VNE problem can be solved in two stages (node mapping and link mapping) or in one stage. In the two stage solutions, virtual nodes are first mapped to the substrate nodes

by satisfying the geographic location constraints. Substrate link resources are also taken into consideration in node mapping stage so that the virtual nodes are mapped to substrate nodes which have sufficient adjacent link resources. After node mapping stage, the link mapping stage determines the substrate paths for virtual links. In this step, link resources are optimized, load balancing is performed and various link constraints are verified. The single substrate path or multiple substrate paths for one virtual link are two possible solutions. The unsplitable (single) paths are often computed by shortest path algorithm, while the splittable (multi) paths are determined by Multi-Commodity Flow (MCF) problem.

The virtual network demands are multifarious. In real world, substrate network is often composed of multi-operator's network (each network is referred as *domain*). Some of the services need to be established over a substrate network which is constituted by networks of several operators (multi-domain). Resource allocation in multi-domain is different from that in single domain. Node mapping in multi-domain need to take into account not only the constraint/optimization (geographic location, CPU) in every domain, but also the domain characteristics and policy to select a substrate domain for each virtual node. The link mapping is also a challenge because infrastructure providers do not expose all the topology information to other providers. The optimal solution can not be found because of lack of exact information. The existing solutions often build a virtual network provider level model and map peering links with simple algorithm (e.g. shortest path algorithm) before mapping sub VN in each domain. The mapping of inter-domain connection and sub VN mapping are generally separated, leading to a bad resource utilization in substrate network.

Some of the VN services have QoS requirements. This kind of requests ask for a VN resource allocation scheme with survivability/reliability [1] consideration. The network failures concerns often single components (node or link). Several components can fail at the same time (region/multiple failure). Many methods are proposed to enhance the VN reliability before and after the failure. Before failure occurrences, failure avoidance methods aim to avoid as much as possible the vulnerable components on the network, while failure protection pre-reserves backup paths and resources for primary links/paths. Although failure avoidance is a simple method that enhances the reliability of a network without extra configuration, there is few works studying it in the context of VNs.

Protection needs backup resource reservation and configurations, but it has a high reliability level. The protection can be implemented on the substrate paths or substrate links. With the path protection, a pair of primary and backup paths can be jointly determined by the disjoint shortest path algorithm. This protection scheme suffers from the high recovery latency problem. With the link protection, the backup path is determined for each primary

[1] In this thesis, the terms *survivability*, *reliability* and *resilience* are interchangeable

link. The recovery is preformed immediately on the upper component of the failure. The backup paths are often pre-selected and computed independently with primary path so that the research space is reduced. As a result, the backup path could be non optimal and even not available because of bad choice of the primary path.

At the same time, since the protection method consumes a great amount of backup resources, the failure avoidance can be a good complement method to protection framework in case of lack of resource.

1.3 Contribution

After the systematic analysis of existing works [LSC15], we found that there are still some important challenges in resource allocation of Network Virtualization. We decided to work on the following directions:

1. In multi-domain network virtualization, there is no efficient link mapping method with partial network information. We aim to propose a link mapping method which increases the efficiency of network resource allocation.

2. Reliability is an important issue for network design, idem for the virtual networks. However, there is few methods which enhance the VN reliability without any extra configuration. Our goal is to take into account the link failure probability and to minimize the VN failure probability during VN mapping.

3. Protection is often used to protect against a network failure. Since it consumes lots of backup resource, the availability of backup resources need to be verified before the primary mapping and remedial methods need to be preformed in case of lack of resource.

The contributions of this thesis are summarized and described hereafter:

1. In order to embed virtual networks on multi-physical (multi-domain) network, we develop a framework for node and link mappings. In our proposition, link mapping is improved by combining and jointly mapping the intra domain and peering (inter domain) links. The information discrimination policy in our method is easy to get. The intermediate layer is adopted to coordinate the link mapping and to facilitate the information exchange. Our framework [LSC16b] simplifies the mapping into two stages: VN partitioning and link mapping. The virtual nodes are first partitioned and mapped to the domains by applying existing multi-domain node mapping algorithm.

Subsequently, a series of iterative downsizing link mappings are performed. In each domain, an augmented graph and local sub VN are created with partial and incomplete information about other domains. The downsizing algorithm maps the intra domain and peering links in the same MCF (multi-commodity flow) problem. In this way, the chose of substrate peering link is no longer an independent step, but it is co-optimized with intra domain links. Our reinforced method [LSC16a] explores the problem of the sequence of mapping the domains and proposes a recursive method to find the best mapping solution.

2. In order to provide the reliability of Network Virtualization, we design a reliable virtual network embedding framework [LSC17a]. In our framework, we assume an independent failure probability for each physical link and we adopt a failure avoidance method to enhance the reliability of VN. Embedding a reliable virtual network corresponds to the problem of minimizing the failure probability of the VN. With the unlimited bandwidth case, we show that the reliability optimization corresponds to the problem of Steiner tree, which is NP-hard. The problem becomes more complex when the bandwidth constraints need to be satisfied. For this case, we propose an exact ILP formulation and two efficient heuristics (baseline and reinforced) to solve the problem. Baseline heuristic reduces only the failure probability by mapping virtual links to shortest path with reliable links. It reuses the mapped substrate links to minimize the overall failure probability of VN. Furthermore, reinforced heuristic includes the bandwidth factor in the cost function so that it reduces also the bandwidth resource utilization.

3. We apply the share link protection technique to Network Virtualization and proposed a survivable virtual network embedding framework. This framework [LSC17b] combines the failure avoidance and the failure protection. With the separation of primary and backup resources, we can predict the existence of the backup path upon primary mapping. The virtual links are mapped to the primary paths that have available backup paths on each link of the primary path. In case of insufficient backup resource, failure avoidance method is adopted to reduce the overall failure probability. Our formulation protects substrate links and optimizes the probability of surviving upon a simple link failure. Furthermore, we find that the maximum bandwidth capacity for link protection corresponds the maxflow value of the backup capacity on the graph. In this way, all the potential backup paths are explored and managed as independent resources without complex computations and configurations. These backup paths determined by maxflow

algorithm, are allocated and consumable linearly (like primary resources) and thus are easier to manage than the existing shared link protection scheme.

1.4 Plan

This manuscript is organized as follows. In Chapter 2, we introduce the network virtualization concept, architecture and virtual network embedding problem. We give an in-deep analysis of the existing work of VNE, including basic node/link mapping, one stage VNE, and multi-domain VNE. We also classified the existing work of survivable VNE into node protection, link protection, failure avoidance and post failure recovery.

Chapter 3 presents the VNE problem in multi-domain and our contribution that consists in coordinating intra and peering link mapping. We compare the existing multi-domain VNE framework with our work in terms of VN partitioning, inter-domain connection, sub VN mapping and information disclosure model. Comprehensive examples are provided to show the differences. We characterize and show the information transmitted in the network to achieve the mapping. With the partial information, we show how the augmented graph is created and how the downsizing mapping is performed. We also introduce a reinforcement of our method which uses recursive ideal. The methods are validated by 5 different scenarios.

Chapter 4 describes the virtual network reliability model and details our methods to enhance the reliability. We first introduce the generic model which associates probability failures for substrate network components. With links of infinite bandwidth, we transform the probability metric to an additive metric and deduce the exact solution which corresponds to the Steiner tree. Based on this result, we formulate the problem by including the bandwidth constraints. Since the problem is NP-hard, we propose a baseline heuristic, which only reduces the failure probability with verifying the bandwidth constraint, and a reinforced heuristic which also decreases the bandwidth utilization.

In chapter 5, we focus on the design of survivable virtual network by combining the failure avoidance and failure protection technique. We formulate the protection-level-aware VNE problem. A simple on-line backup verification method is proposed to predict the existence of backup path. Furthermore, we propose the pre-computed backup verification method that is based on maxflow algorithm.

Finally, we conclude this thesis and give some perspectives for the future working directions in Chapter 6.

1.5 List of Publications

Publication:

- [LSC15] Shuopeng Li, Mohand Yazid Saidi, and Ken Chen. Survivable virtual network embedding with resource sharing and optimization. In Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS), 2015 International Conference on, pages 1–6. IEEE, Paris, July 2015.
- [LSC16b] Shuopeng Li, Mohand Yazid Saidi, and Ken Chen. Multi-domain virtual network embedding with coordinated link mapping. Advances in Science, Technology and Engineering Systems Journal, 2(3):545–552, 2016.
- [LSC16a] Shuopeng Li, Mohand Yazid Saidi, and Ken Chen. A cloud-oriented algorithm for virtual network embedding over multi-domain. In Local Computer Networks Workshops (LCN Workshops), 2016 IEEE 41st Conference on, pages 50–57. IEEE, Dubai, November 2016.
- [LSC17a] Shuopeng Li, Mohand Yazid Saidi, and Ken Chen. A failure avoidance oriented approach for virtual network reliability enhancement. In Proc. IEEE Int. Conf. Communications (ICC), Paris, May 2017.

Preparation:

- [LSC17b] Shuopeng Li, Mohand Yazid Saidi, and Ken Chen. Survivable services oriented protection-level-aware virtual network embedding. Target: International Journal of Network Management.

Chapter 2

Background and State of Art

2.1 Introduction

In this chapter, we first introduce the basic definition and benefits of network virtualization. To achieve the network virtualization, we describe the problem of virtual network embedding (VNE), the general VNE network model, formulation and tools to solve the problem. The single domain problem can be solved in two stage or one stage. Subsequently, we present the VNE problem in multi-domain, the existing works and remaining shortcoming. Taking into account the reliability of the VN, we present the survivable VNE problem, its principle method and remaining challenges.

2.2 Network Virtualization

2.2.1 Benefits

The concept of network virtualization [APST05] has evolved from Virtual Local Area Networks (VLANs), Virtual Private Networks (VPN), active programmable networks and overlay networks into a pluralist philosophy where many heterogeneous architectures co-exist to offer flexible and scalable services. From an architectural perspective, network virtualization allows [CB09] [BHK12]:

- *Coexistence*: many virtual networks (eventually from different service providers) coexist on the same physical infrastructure;
- *Recursion*: also known as nesting, refers to the possibility that a virtual network can be set up on the top of another virtual network which creates a hierarchy in the network virtualization environment;

- *Inheritance*: virtual networks in the higher levels of the recursion hierarchy could inherit properties from virtual networks in lower levels. Such properties could be some restrictions that come directly from the infrastructure provider for example;
- *Revisitation*: a single physical node (physical link) can be used to host more than one virtual node (virtual link) from the same virtual network.

2.2.2 Design goals

The overall goal of enabling multiple heterogeneous virtual networks to coexist together on a shared physical infrastructure can be subdivided into several smaller objectives. These goals provide a guideline to design a protocol or an algorithm for virtual networks. They consist in the following points:

- *Flexibility*: every virtual network is independent from the underlying physical infrastructure and the coexisting virtual networks in terms of topology, routing, forwarding functionalities and control protocols;
- *Manageability*: each virtual network can be managed independently with the capability of service providers to provide end-to-end control over many virtual networks;
- *Scalability*: infrastructure providers should be capable of scaling to support an increasing number of coexisting virtual networks;
- *Isolation*: since many virtual networks should co-exist, they should be isolated from each other in order to ensure security and privacy and to improve fault tolerance;
- *Stability and convergence*: any errors in the underlying physical infrastructure should be limited in order to affect the minimum number of virtual networks on top of the physical infrastructure;
- *Heterogeneity*: not only the underlying physical infrastructure could be composed of heterogeneous technologies (e.g., optical and wireless technologies) but also virtual networks on top of it could be heterogeneous (e.g., by using different protocols).

2.2.3 Architecture

The main idea of network virtualization is to decouple infrastructure from services in the traditional ISPs. This leads to an architecture composed of two new entities: *Service Provider (SP)* and *Infrastructure Provider (InP)*. In this business model, InPs become responsible for

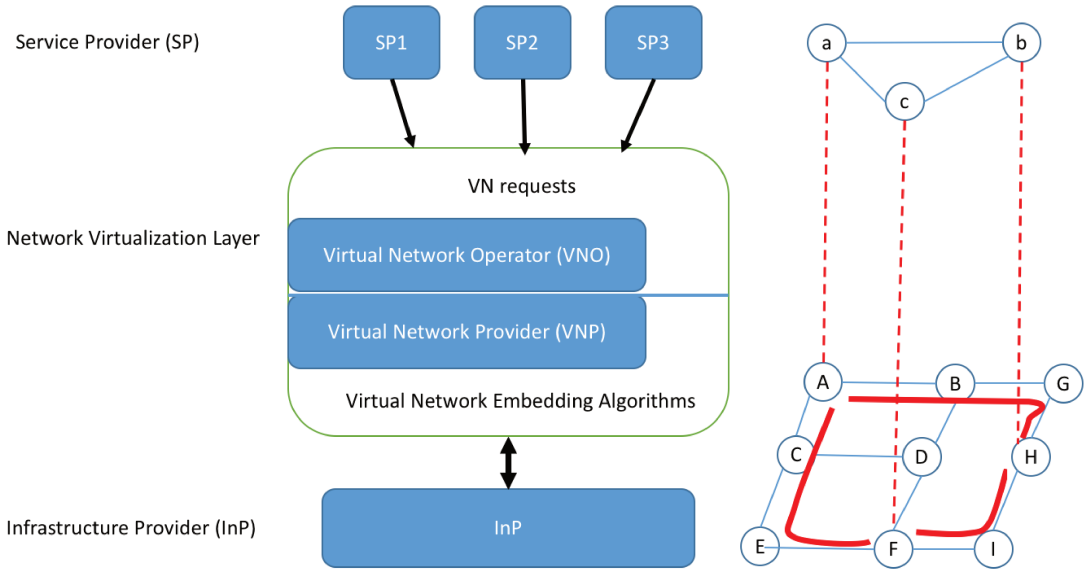


Fig. 2.1 Network virtualization architecture

allocating a part of the resources of their physical networks (physical nodes interconnected by physical links) to SPs. This is done by embedding a virtual network (virtual nodes interconnected by virtual links) on one or more physical networks. This approach allows each InP to manage its physical infrastructure independently while each SP focuses on providing and improving end-to-end services to end users.

Some authors [SWP+09] separate the management and business roles of the SP by identifying three main players (see Figure 2.1):

- *Virtual Network Provider (VNP)*, which is responsible for assembling virtual resources from one or multiple InPs into a virtual topology.
- *Virtual Network Operator (VNO)*, which is responsible for the installation and operation of a VN over the virtual topology provided by the VNP according to the needs of the SP, and thus realizes a tailored connectivity service.
- *Service Provider (SP)*, which uses the virtual network to offer his service. This can be a value-added service and then the SP acts as an application service provider, or a transport service with the SP acting as a network service provider.

2.2.4 Virtual Network Embedding (VNE)

The challenging problem of virtual network embedding (VNE) which consists to determine the best mapping of virtual networks to a substrate network, concerns the resource allocation [FBTB+13]. Through a dynamic mapping of virtual resources onto physical hardware, the benefit of existing hardware can be maximized. The optimal dynamic resource allocation, leading to the self-configuration and organization of future networks, will be more and more desired to provide customized end-to-end guaranteed services to end users. This optimality can be computed with regard to different objectives, ranging from QoS, economical profit, or survivability over energy-efficiency to security of the networks.

The hardware abstraction provided by the virtualization solution provides a common denominator, allowing any substrate resource to host virtual resources of the same type. Typically, a substrate resource is partitioned to host several virtual resources. For example, a virtual node can, in principle, be hosted by any available substrate node. Moreover, a single substrate node can host several virtual nodes. Thus, the mapping of virtual nodes to substrate nodes describes a (n : 1) relationship (a strict partition of substrate resources).

In some cases, substrate resources can also be combined to create new virtual resources. This is the case for a virtual link which spans several links (i.e. a path) in the substrate network. In this case, a virtual link between two virtual nodes v and w is mapped to a path in the substrate network that connects the substrate hosts of v and w. Each substrate link may then be part of several virtual links. As such, the mapping of virtual links to substrate paths describes a (n : m) relationship.

A virtual network embedding example is shown in the right side of Figure 2.1. A virtual network request is generated by service provider. This network is composed of three virtual nodes $\{a,b,c\}$ and 3 virtual links $\{a-b, a-c, b-c\}$. The substrate network managed by infrastructure provider contains 9 substrate nodes and 11 substrate links. The VNE solution is shown by the red dotted lines. Virtual nodes a, b, and c are mapped to A, H and F respectively. Virtual links $a-c$, $a-b$ and $b-c$ are mapped to $A-C-E-F$, $A-B-G-H$ and $H-I-F$, respectively.

2.3 VNE optimization

The VNE problem is basically a resource allocation problem. To solve this resource allocation problem, many framework/methods/algorithms are proposed. Most of the existing works are based on the graph models on which the objectives and constraints are formulated to build an optimization problem. This kind of optimization problem is often NP-hard. Optimization methods (exact and heuristics) are adopted to give the solutions.

In this section, we present a general network model and VNE formulation. The network model is adopted by most of the authors in existing works [CRB09], while the objectives and constraint may be different or more complex according to the requirement. We give some examples of optimization metric listed in Table 2.2. To solve the formulated problem, we briefly introduce some optimization tools.

2.3.1 Substrate network

The substrate network is modeled as a connected directed graph $G^S(N^S, L^S)$, where N^S is the set of substrate nodes and L^S is the set of substrate links. Each substrate node $n^s \in N^S$ is associated with various metric weights and constraints like CPU capacity $cpu(n^s)$ and geographic location $loc(n^s)$. Similarly, a substrate link l_s is associated with bandwidth capacity $C(l^s)$, failure probability P_{l^s}, etc..

2.3.2 Virtual network

The virtual network is also modeled as a connected directed graph $G^V(N^V, L^V)$, where N^V is the set of virtual nodes and L^V is the set of virtual links. Each virtual node $n^v \in N^V$ is associated with CPU capacity demand $cpu(n^v)$, geographic location $loc(n^v)$ and the distance $dis(n^v)$ specifying how far a virtual node n^v can be placed from its $loc(n^v)$. Each $l^v \in L^V$ is associated with metrics and constraints like bandwidth demand $bw(l^v)$. In addition, each virtual network G^V has a lifetime $t(G^V)$.

2.3.3 VNE Formulation

We describe the generic formulation of VNE that aims to embed a virtual network $G^V(N^V, L^V)$ on a substrate network $G^S(N^S, L^S)$. In this formulation, the objective function aims to optimize a function that combines both node and link metrics (node CPU and link bandwidth). Depending on applications, this objective function may take various forms.

Variables:

- $U_{n^s}^{n^v}$: cpu on substrate node n^s for virtual node n^v.
- $F_{l^s}^{l^v}$: flow on substrate link l^s for virtual link l^v.

Objective:

Minimize:

$$Obj(U_{n^s}^{n^v}, F_{l^s}^{l^v}) \tag{2.1}$$

Subject to:

Node location constraints:

$$distance(loc(n^s), loc(n^v)) \leq dis(n^v) \tag{2.2}$$

CPU capacity constraints:

$$\sum_{n^v \in N^V} cpu(n^v) U_{n^s}^{n^v} \leq Res(n^s), \qquad \forall n^s \in N^S \tag{2.3}$$

Flow conservation constraints:

$$\sum_{n \in N^S | \exists l^s = (m,n)} F_{l^s}^{l^v} - \sum_{n \in N^S | \exists l^s = (n,m)} F_{l^s}^{l^v} = \begin{cases} 1, & m = M(src(l^v)) \\ -1, & m = M(dst(l^v)) \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$
$$, \forall m \in N^s, l^v \in L^V$$

Bandwidth capacity constraints:

$$\sum_{l^v \in L^V} bw(l^v) F_{l^s}^{l^v} \leq Res(l^s), \qquad \forall l^s \in L^S \tag{2.5}$$

In our generic formulation, the constraints are flow conservation and resource capacities. More constraints are included according to the objective and context of problem.

The variables $U_{n^s}^{n^v}$ represent the CPU resources allocated to a virtual nodes n^v on a substrate node n^s. These variables are binary, since the virtual node need to be embedded entirely to a substrate node to perform its function. However, the variable $F_{l^s}^{l^v}$, which refers to the flow of a virtual link l^v on a substrate link l^s, can be either binary or continuous in $(0,1)$. When path splitting is allowed, the variables $F_{l^s}^{l^v}$ are continuous whereas they are binary for unplittable traffics. We note that path splitting allows to route the traffic of a virtual link on multiple paths whereas only one path should be used for the case of unsplittable traffic. The amount of flow routed on a substrate link l^s for the virtual link l^v is equal to $bw(l^v) \times F_{l^v}^{l^s}$.

Objective (2.1) is a function of variables $U_{n^s}^{n^v}$ and $F_{l^s}^{l^v}$. The objective function of VNE problem depends on the need of service provider. For example, we can combine node and link resource utilizations in the objective function so that the consummation of a VN is minimized. Other objectives (QoS, delay, failure) could be optimized in a similar manner.

Constraints (2.2) and (2.3) are node constraints. The node location constraint 2.2 aims to guarantee that the distance between the request location ($loc(n^v)$) and the location of n^s ($loc(n^s)$), on which n^v is mapped, does not exceed a given tolerant distance ($dis(n^v)$). The

capacity constraints 2.3 ensure that n^s has enough CPU resources to support virtual nodes that are mapped to n^s. The node constraints are used by mapping algorithms to determine a candidate set of substrate nodes before the node mapping step so that the research space is reduced.

Constraints (2.4) and (2.5) are link constraints. In constraints 2.4, $src(l^v)$ and $dst(l^v)$ retrieve respectively the source and destination of l^v. $M(n^v)$ retrieves the substrate node that n^v is mapped to. The flow conservation constraints 2.4 show us the establishment of substrate path on which a virtual link is mapped. The source nodes of the paths have egress flow (sum of the flow is equal to 1), while the destination nodes of the paths have ingress flow (sum of the flow is equal to -1). The ingress flows and egress flows on the intermediate nodes are equal. The bandwidth capacity constraints 2.5 ensure that each l^s has enough bandwidth resource to support virtual links that pass through l^s.

2.3.4 Examples of objective functions

2.3.4.1 CPU and bandwidth

In [YYRC08], the objective of VNE is modeled by the sum of node CPUs and link bandwidths,

$$R(G^V) = \sum_{l^v \in L^V} bw(l^v) + \sum_{n^v \in N^V} cpu(n^v) \tag{2.6}$$

where $R(G^V)$ is the revenue of G^v. The nodes are mapped with a greedy algorithm. In their solution, the available resource for a substrate node is given by its cpu resource and bandwidth resources of adjacent links,

$$H(n^s) = cpu(n^s) \sum_{l^s \in AL(n^s)} bw(l^s) \tag{2.7}$$

where $AL(n^s)$ is the set of all adjacent substrate links of n^s. Virtual nodes are mapped to substrate nodes with high available resources.

2.3.4.2 Delay

The delay of a virtual link is the sum of the delays of the links along the related path in the substrate network. In [ZCP12], the sum of the delay of all the virtual links in a VN is minimized,

$$Min \sum_{l^v \in L^V} \sum_{l^s \in L^S} D_{l^s} F_{l^s}^{l^v}$$

where D_{l^s} is the delay of a substrate link.

2.3.4.3 Penalty

If a service is violated due to a substrate resource failure, the service will be affected and subsequently result in penalties for the InP based on the level of frustration of the SP. Fast recovery methods can reduce this penalty. By applying the virtual network protection method, the penalty is calculated by the part of flow without protection (violated by the failure) [RB13],

$$\sum_{l^v \in L^V} S(l^v)[1 - \sum_{p \in P(v)} \frac{B(p, l^v)}{bw(l^v)}]$$

Where $S(l^v)$ is the monetary penalty function of l^v, $P(v)$ is the pre-selected protection path of l^v, $B(p, l^v)$ is the backup flow on the path p to protect l^v.

2.3.5 Optimization problem and methods

The VNE formulation that we presented in the previous section is often a linear programming (LP) problem. We briefly introduce the definition of LP problem, its methods and the software to solve the problem.

2.3.5.1 Linear programming (LP and ILP)

In a linear programming optimization problem, both the objective function and the constraints are linear functions. When the decision variables are both continuous (allowed to be fractional), we call it linear programming (LP), whereas integer linear programming (ILP) means that the variables are discrete. When the decision variables are both discrete and continuous, we are dealing with mixed integer programming problems (MIP). Hence, MIP models generalize LP and IP models.

Linear programming (LP) is one of the most satisfactory models of solving optimization problems because efficient exact algorithms such as *simplex* method can be adopted. The efficiency of the algorithms is due to the fact that the feasible region of the problem is a convex set and the objective function is a convex function. Then, the global optimum solution is necessarily a node of the polytope representing the feasible region. Moreover, any local optima solution is a global optimum.

However, ILP and MIP can not be solved in polynomial time. Solving these problems has improved dramatically the use of advanced optimization techniques such as relaxations and decomposition approaches, and cutting plane algorithms. Enumerative algorithms such as *branch and bound* may be used for small instances to give the exact solution. The size is not the only indicator of the complexity of the problem, but also its structure. Metaheuristics

are one of the competing algorithms for this class of problems to obtain good solutions for instances considered too complex to be solved in an exact manner.

2.3.5.2 Optimization methods

Simplex The simplex method is a method for solving problems in linear programming. The LP problem is first converted into a standard form. The linear inequalities of standard form defines a polytope as feasible solution region. The Simplex algorithm starts with a basic feasible solution.

Then, the algorithm walks along edges of the polytope to extreme points with greater and greater objective values. This continues until the maximum value is reached or an unbounded edge is visited, concluding that the problem has no solution. The algorithm always terminates because the number of vertices in the polytope is finite. Moreover, since we jump between vertices always in the same direction (that of the objective function), we hope that the number of vertices visited will be small.

Branch and bound Branch and bound is a method to solve the ILP and MIP problem. The solution area is usually a decision tree where each decision is represented by an edge. The leaves of this tree is the set of all possible solutions. Instead of checking each combination, the algorithm prune the tree, i.e. ignore completely sections of it which we know cannot have better results than the best one we have already found, without needing to fully calculate what results they achieve.

The feasible solution tree is partitioned into convex branch. Lower/upper bounds are found for each branch by relaxing the integer constraints which are making the problem hard to solve. The algorithm keeps searching the main tree for solutions better than what you have and prunes the sub-trees which is worse than the existing solution.

Branch and bound cannot guarantee short computation time as that depends on the degree of successful pruning which itself depends on the problem definition (values, costs etc). At worst, the entire tree need to be computed.

2.3.5.3 CPLEX

The IBM ILOG CPLEX Optimizer solves integer programming problems, very large linear programming problems using either primal or dual variants of the simplex method or the barrier interior point method, convex and non-convex quadratic programming problems, and convex quadratically constrained problems.

The CPLEX Optimizer provides interfaces to the C++, C#, python and Java languages. Additionally, connectors to Microsoft Excel and MATLAB are provided. Finally, a stand-alone Interactive Optimizer executable is provided for debugging and other purposes.

The CPLEX Optimizer is accessible through independent modeling systems such as AIMMS, AMPL, GAMS, OptimJ and TOMLAB. In addition to that, AMPL provides an interface to the CPLEX Optimizer.

An example of CPLEX optimization file can be found in Appendix B. In this thesis, we use CPLEX to to solve the optimization problem.

2.4 Single domain VNE

The basic VNE problem that we described above (in a single domain) is NP-hard. Different approaches are proposed. We present two stage solutions (node mapping and link mapping), one stage solution and the VN reconfiguration problem.

2.4.1 Two stage solutions

The VNE problem solution depends on the objectives. An intuitive objective of VNE consists to optimize the overall consumed resources of a VN, so that a maximum number of VNs can be accommodate on the SN. This single objective problem corresponds to the coordinated optimization of node resources (cpu) and link resources (bandwidth). Instead of joint optimization of the node and link resources, the 2 stage solutions are proposed.

2.4.1.1 Node mapping

In [ZA06], the network resources corresponds to the node/link stress. A greedy node mapping method is proposed to measure the potential of a substrate node by weighting the node stress. The later on a given node is determined by combining its residual resources and the distances to other substrate nodes. Virtual nodes with high degree [1] are mapped to substrate nodes with high resource availability.

For substrate node ranking, the Markov Random Walk algorithm is applied [CSZ+11]. The rank is determined by combining Formula (2.7) and the ranks of nodes that can be reached from a node u. The later is deduced by dividing the reachable nodes into two groups: (1) the nodes that are incident to the outgoing links from u and (2) the nodes that can be reached from u via multiple hops. In this way, virtual nodes with high ranks are mapped to substrate nodes with the same high ranks.

[1] the degree means the notion of degree of node in graph theory

In [CRB09], node mapping is preformed in a way that the mapping of virtual links to physical paths in the subsequent phase is facilitated. For this purpose, the physical network graph is extended by adding one meta-node for each virtual node. A meta-node is connected to a subset of substrate nodes with the use of links of infinite bandwidth. Then each virtual link with its bandwidth constraints is treated as a commodity consisting of a pair of meta-nodes. In this way, the problem is first formulated as mixed integer programming (MIP) before the relaxation of integer constraints to obtain a linear programming formulation that can be solved in polynomial time. Deterministic and randomized rounding techniques are used on the solution of the linear program to approximate the values of the binary variables in the original MIP.

The problem of *hidden hop* is pointed out in [BHFDM12]. The intermediate nodes in a SN path, acting as *hidden hop*, have a CPU expenditure because they have to be configured and they have to correctly forward the packets passing through the virtual link. The CPU resource that must be assigned to an intermediate node is a function of the virtual link demand. They formulate the problem with the objective of the sum of the spare bandwidth and spare CPU in the substrate network. Heuristic is proposed, where hidden hops are optimized in link mapping stage by minimizing the number of hops and accomplishing the bandwidth and CPU constraints.

[BHFDM12] also considers the case where some nodes and links do not have a constrained demand of resources. Each virtual node is mapped to a substrate node which has remaining CPU capacity. Each virtual link is mapped to a shortest path which has non-zero remaining resources on links and nodes.

Most of VNE described in the literature do not explicitly support the collocation of virtual nodes belonging to the same VN, but only collocate nodes of different VN. In [FSF13], the problem of virtual node collocation is addressed. Pre-cluster VN, which consists in transforming the original VN request to a VN where a single virtual node describes multiple virtual nodes, can be preformed in advance. Concretely, the proposed algorithm computes an optimal VN pre-clustering so that, given an estimation of the available resources in the physical network, the amount of network resources required for the VN embedding is minimized.

2.4.1.2 Link mapping

A single link mapping problem is basically a standard routing problem. Shortest path algorithm can be adopted to select the path with minimum distance [ZA06]. The difficulty of VN link mapping is that the VN request is a whole large unit. To utilize the flexibility of the small topology, the authors in [ZA06] propose to break up the VN into a number of

connected sub-virtual networks (subVN). Each subVN corresponds to a simple star topology. The connections between subVNs define the constraints relating substrate node selection.

In [YYRC08], the authors show that without path splitting, the VN link mapping problem corresponds to the Unsplittable Flow problem (UFP) that can be approximated by the k-shortest path algorithm. With path splitting, the link mapping problem can be reduced to the Multi-Commodity Flow Problem (MCF) which can be solved in polynomial time. Having multiple paths enables better resource utilization by harnessing the small pieces of available bandwidth, allowing the substrate to accept more VN requests. Furthermore, [YYRC08] enhances the path splitting in link mapping by adding a backtracking procedure which remaps one extremity node of the "bottleneck" substrate link. Since the multi-commodity flow is a widely used model for link mapping, we detail it below.

Multi-commodity flow A commodity is a flow demand from a source node to a sink node. Multi-commodity flow (MCF) problem is a network flow problem with multiple commodities between source and sink nodes. The goal of MCF is to find an optimal routine of all the flows which satisfies constraints including: link capacity, flow conservation on source, transit and destination nodes. These constraints correspond to the link constraints 2.4 and 2.5 shown in Section 2.3.3. As the virtual links are treated as commodities, we deduce that the VNE link mapping problem corresponds to MCF problem.

The optimal solution of MCF problem can be determined by linear programming (LP) in polynomial time. However, finding a feasible integer solution is NP-complete. Indeed, the integer solution to MCF corresponds to the solution of UFP that is NP-hard.

2.4.2 One stage solutions

Since 2-stage VNE solutions are lack of cooperation, some solutions mapping nodes and links in the same stage have been proposed.

In [LK09], a backtracking algorithm based on a subgraph isomorphism search method is proposed. After determining the candidate set of all the substrate nodes that can support each virtual nodes, a virtual node is first mapped to a candidate substrate node in its corresponding set. Next, an adjacent virtual node is mapped by searching for a substrate node that satisfies the path length and link capacity constraints. The path between the two nodes is determined and validated. The algorithm continues to map the next adjacent virtual nodes till all the virtual nodes are mapped. If any node or link constraint is violated and there exists no candidates, the algorithm does a backtracking step to the last valid mapping and tries to continue by selecting another candidate. The advantage of this single stage approach is that link mapping constraints are taken into account at each step of the mapping. When a

bad mapping decision is detected, it can be revised by simply backtracking to the last valid mapping decision, whereas the two stage approach has to remap all links which is very expensive in terms of runtime.

A similar approach is proposed in [CSZ+11]. After proposing a node rank algorithm, [CSZ+11] constructs a breadth-first search tree of VN request, where the root node is the virtual node with the largest node rank value. At each level of the search tree, nodes are sorted by their node rank values in non-increasing order. For each virtual node, candidate substrate node list is also ordered by node rank value. The same embedding procedure as [LK09] with backtracking is applied.

In [JK12], a Column Generation VNE reformulating the embedding problem in terms of Independent Embedding Configurations (IECs) is proposed. An IEC defines an embedding solution of one VN request. By allowing a small delay of VN setup, their embedding framework are done by small-batch at each planning period (each embedding procedure maps several VNs). Using Column Generation technique means that the VNE problem is decomposed into a master problem and a pricing problem. The master problem corresponds to the choice of a maximum of N IECs among the generated IECs, in order to maximize the overall revenue. the pricing problem corresponds to the problem of generating an additional column (IEC) for the constraint matrix of the current master problem. The master and pricing problem are solved by CPLEX.

In [WSW12], an alternate formulation of the virtual network mapping problem is provided. The VN is introduced by traffic matrices, implying fully connected VNs. The argument is that the internal topology of a virtual network should not matter as long as the mapped structure can provide connectivity at a sufficient level of service (e.g., bandwidth). [WSW12] formulates the traffic-matrix-based mapping problem as a p-Hub location problem, which seeks to obtain the optimized placement of hubs which minimizes the demand-weighted flow cost between the demand nodes in *hub and spoke networks*.

2.4.3 VN reconfiguration

The cost of reconfiguration includes both the computational cost and the service disruption cost. Computational cost refers to the expenses involved in recomputing the VN assignment and therefore is proportional to the frequency of VN reconfigurations. Service disruption cost is incurred because the normal operation of a VN is affected when it is switched from one assignment to another. Unlike flow rerouting where at most one path change happens per rerouting event, the reconfiguration of a single VN may result in more substantial changes involving both node switching and multiple path switching. Node switching happens when the assignment of a virtual node is changed from one substrate node to another. Path switching

happens when the assignment of a virtual link is changed from one substrate path to another. The disruption of the reconfiguration is therefore significant if all VNs are reconfigured periodically.

A selective reconfiguration scheme to perform efficient reconfiguration by giving higher priority to critical VNs is developed in [ZA06]. The key idea is to reconfigure only the critical VNs that allow the decrease of maximum stress.

In [YYRC08], path migration is proposed to reconfigure embedded VNs. At a given instant, a set of long duration VNs are selected. They are remapped by performing a MCF with the same paths. Path migration allows us to (periodically) treat the online embedding problem as an offline problem, to capitalize on the efficiency gains that are possible when handling a large collection of requests together.

In [SYAL13], the problem of dynamic VNs is addressed. Four possible changes are presented: (1) component deletion, (2)resource requirement decrease (3) components adding, and (4) resource requirement increase. (1) and (2) correspond to releasing the resources on substrate network, while (3) and (4) involve allocating new resources. If there are enough available resources on the substrate components that provide resources for these VN nodes or edges, resources are reallocated to them on demand to satisfy the new requirements. However, if there are not enough available resources on these substrate nodes or links, the proposed algorithm remaps these VN nodes or edges onto other substrate nodes or paths. New node or edge resources constraints should be satisfied and the remapping cost minimized.

2.5 Multi domain VNE

As some of the services are built over several infrastructures run by different operators, we expect that the VN requests concerns several domains. We present below the works treating the VNE in multi-domain networks.

2.5.1 Existing works

In [CSB10], a distributed multi-domain VNE architecture is proposed. Upon receiving a VN, an InP decides whether to reject or to accept the request according to its internal policies (e.g. intra-domain mapping). If the InP can only partially embed a VN, it forwards the rest of the request to other InPs. Instead of blindly disseminating the rest of the request, it uses geographic constraint as beacons to route the request to other possible InPs. In case of successful embedding of a VN request, the InP back-propagates the mapping and the price.

In [HLAZ11], the authors investigate the VN graph splitting problem in multi-domain. Each virtual node can have several candidates substrate nodes in multiple InPs. The problem of VN splitting is to determine the embedding InPs for all the virtual nodes so that the VN provisioning cost is minimized. The provisioning cost consists generally of the sum of node allocations to InPs and link allocations to intra-domain or inter-domain (e.g. if virtual node *a* and *b* are embedded to different domains, the virtual link between *a* and *b* is inter-domain). Note that the objective is to select embedding InPs for nodes without exactly embedding a virtual node to a substrate node. VN splitting across two InPs can be reduced to maxflow/mincut problem, while VN splitting across multiple InPs is formulated as an ILP and it can be solved by the branch and bound algorithm. After partitioning virtual nodes and links to the domains, [HLAZ11] proposes an exact embedding formulation that enables optimal and simultaneous node and link mappings.

In [DRP13b], the authors discuss the information disclosure in multi-domain. After examining the policies of InPs, they conclude that it is undesirable and inefficient to advertise the detailed intra domain information (e.g. substrate network topology and resource availability) out the InP. Only the costs of links interconnecting peering nodes are collected and transmitted to VNP. Meanwhile, the authors of [DRP13b] use traffic matrices to model the VN request and propose a VN segment framework, where each virtual node is mapped onto a particular peering node. The sub VN request (VN segment) for each InP is simplified, since traffic matrices aggregate the virtual links from a virtual node to the same InP. The problem is formulated as ILP.

An automated framework for VN embedding across multiple substrate networks is proposed in [DRP13a]. It comprises a realistic evaluation environment for VN request partitioning and intra-domain mapping algorithms. Furthermore, the proposed method can be used for the investigation of various VN embedding aspects, such as the impact of information disclosure on VN embedding efficiency and the suitability of different VN request descriptions (e.g., topology-based vs. traffic-matrix based VN requests). VN embedding insights gained by their framework can assist InPs and potential VNPs in configuring their information disclosure policies and adjusting their cost models.

To solve the VN segment problem mentioned in [DRP13b], a Particle Swarm Optimization (PSO) based heuristic is proposed in [GWQ+15]. Each particle has its position and velocity. The position of a particle is a possible solution. The velocity leads the particle to fly to a new position. The particles fly in the problem space by updating their positions and velocities. An appropriate-optimal solution of VN partitioning is generated through the evolution process of the particles.

<table>
<tr><td></td><td>VN partitioning</td><td>inter-domain connection</td><td>sub VN mapping</td><td>information model</td></tr>
<tr><td>Houidi [HLAZ11]</td><td>maxflow/Exact</td><td>VPN connection</td><td>Exact VNE</td><td>functional attributes</td></tr>
<tr><td>Dietrich [DRP13b]</td><td colspan="2">Exact</td><td rowspan="2">single domain VNE</td><td rowspan="2">peering link graph</td></tr>
<tr><td>Guo [GWQ+15]</td><td colspan="2">Particle Swarm Heuristic</td></tr>
<tr><td>Shen [SXYC14]</td><td>node mapping</td><td>single shortest-path</td><td>single domain link mapping</td><td>full mesh graph</td></tr>
</table>

Table 2.1 Multi-domain mapping method comparison

In [SXYC14], another information sharing scheme is proposed. The intra-domain link information is given by a length-based price for connecting any two nodes in the domain. This model allows VNP to have a general abstraction of overall substrate networks with estimated link costs. Subsequently, a single minimum-cost path on the abstraction graph is determined by VNP in order to select a single inter-domain substrate link for each virtual link. The intra-domain mapping is then performed by single-domain VNE.

2.5.2 Comparison of embedding methods

After presenting the existing works, we want to give a systematic analyse of the mapping method in terms of information exchange model and mapping procedure. This analysis will help us to identify the shortcoming of the existing works and guide us to our working direction. Table 2.1 shows the difference between the methods. Note that we focus only on the algorithms and computations ([HLAZ11][DRP13b][SXYC14][GWQ+15]), but not the architecture ([ARN16][CSB10][DRP13a]).

Multi-domain VNE is based on different information model shown in the last line of Table 2.1. With the partial information, multi-domain VNE consists in three major steps:

(i) Partitioning the VN request into each InP.

(ii) Establishing inter-domain connection (peering links) between InPs. In some methods, this step corresponds to VN segmentation.

(iii) Embedding each sub VN request in each InP using intra-domain algorithm.

2.5.2.1 Multi-domain information disclosure model

Because of policy and efficiency reasons, InPs cannot advertise their complete information to others, so it is critical to make clear the information disclosure policy.

The substrate resource information consists of functional and non-functional attributes [HLAZ11]:

- Functional attributes: define characteristics and properties of the substrate resources and including static parameters like node type (e.g. router, switch), node processor type and capacity (e.g. CPU, network processor), link/path type (e.g. VLAN, L3/L2 VPN), network interface type and number, geographic location, cost,etc..
- Non-functional attributes: specify criteria and constraints related to the substrate resources including dynamic (real-time) parameters like available node capacity (CPU), available link capacity (bandwidth), actual QoS parameters, geographic coordinates, etc..

InPs usually accept to publish functional attributes, but the non-functional attributes are not advertised since such dynamic attribute advertisements require real-time monitoring and increase the network load with the exchange of a large amount of information.

However, VNP can enhance their limited substrate network view with certain aspects of the substrate topology which are assumed as non confidential. InPs may disclose the information about their peerings. Such information enables a VNP to construct a more comprehensive view of the underlay network, including the location and connectivity of peering nodes [DRP13b].

In [SXYC14], three types of domain resource information are assumed to be provided to VNP:

- node: its location, available capacity and unit price;
- peering link: its vertices, available capacity and unit price;
- intra-domain link: a length-based price for connecting any two nodes in its domain.

With this information about intra-domain and peering links, an extended and approximated view of the intra-domain topology is determined. [SXYC14] assumes a full mesh topology for each domain, and thus each pair of substrate nodes within the same domain are connected via augmented links. With the given length-based unit prices of intra-domain links, VNP is able to compute the price of each augmented link.

2.5.2.2 VN partitioning

Various solutions are proposed in the literature for VN partitioning. We describe here the two main approaches. In the first approach [HLAZ11], a substrate domain is selected to support each virtual node such that the embedding cost is minimized. This approach assumes that the embedding costs of virtual nodes to domains are known and defined. This problem can be solved by maxflow for the case of 2 domains and by an exact ILP formulation for the other cases.

Another approach proposed in [DRP13b] and [GWQ+15] consists to combine the VN partitioning and inter-domain connections. On the peering link graph, the virtual nodes are first mapped to corresponding peering nodes, which correspond to the egress nodes of domains. Note that this node mapping is not final node mapping since the final node mapping will be performed next in sub VN mapping stage with consideration of other constraints (e.g. geographic constraint). In this way, the virtual nodes are first assigned to the domains so that virtual links can be segmented in the same stage.

2.5.2.3 Inter-domain connection (VN segmentation)

The objective of this step is to segment the virtual links so that the intra-domain and inter-domain parts of virtual links are separated. Two approaches are defined for VN segmentation, (1) determining the peering nodes from where the virtual links egress the domains, and (2) determining the inter-domain link used by the virtual links. As mentioned above, Dietrich [DRP13b] and Guo [GWQ+15] follows the first approach.Houidi [HLAZ11] and Shen [SXYC14] apply the second approach.

In [HLAZ11], each peering link is assumed as a single VPN (Virtual Private Network) connection. In [SXYC14], the peering link path is determined by shortest-path algorithm on the full mesh graph of VNP layer. The full mesh graph is an estimation of intra domain topology. Therefore, only the peering link availability is changed over time, since the intra domain model is the same all the time. As a result, shortest path based peering links have always the same path if the peering links are not overloaded. This phenomenon will result in difficulty of later sub VN mapping in all the domains. The sub VN mapping have to either choose a costly intra domain mapping, or reject the VN request if no mapping solution is found.

2.5.2.4 Sub VN mapping

In Dietrich [DRP13b] and Guo [GWQ+15], this stage is truly a single domain VNE problem (node and link mapping). The advantage is that one-stage VNE method (e.g. subgraph isomorphism) can be applied.

Houidi [HLAZ11] also proposes a one-stage exact VNE method. However, their difficulty is that, after intra-domain mapping, they need to map peering links (VPN connection), and then execute the proposed algorithm again to assign the inter-domain virtual links. Therefore, the mapping is coupled into 2 sub VN, which can result in under optimization.

In Shen [SXYC14], this step is a VN link mapping problem. Any VNE link mapping stage algorithm can be adopted. The part of intra-domain mapping on full mesh network are replaced by this link mapping with real topology and attributes.

2.5.3 Shortcoming of existing work

Existing solutions mainly focus on the partitioning of a multi-domain VN to each domain. One of the shortcomings in these frameworks is the lack of efficient link mapping method especially for the inter-domain connection. Substrate peering nodes and links on the path of inter domain virtual links need to be optimized, while inter-domain connection and sub VN mapping are independent step in the existing works.

The shortest path based inter-domain link selection does not well leverage the partial information of the domains, so peering links can always have the same path if the peering links are not overloaded. Since establishing peering links is a part of link mapping, the bad choice of peering nodes will probably influence the intra-domain paths. It will result in difficulty of later sub VN mapping in all the domains. The sub VN mapping have to either choose a costly intra domain mapping, or reject the VN request if no mapping solution is found.

2.6 Survivable VNE

A failure represents the condition in which the system deviates from fulfilling its intended functionality or the expected behavior. A network fails for various reasons, like bottlenecks, software attacks, natural disasters, etc.

The task of survivable VNE is to embed a virtual network that can deal with virtual and substrate network failures in a way, that after the failure, the virtual network is still operating. The failure and the fixing/recovery procedure should be transparent to the users of the virtual network.

Survivable VNE [HKA13] deals with failures of different types. The categories of failure can be summarized as follows,

- *Failure layer*: the failures occur on substrate and virtual layer. A failure on virtual layer is the dysfunction of a whole VN network. It often comes from the service provider. A failure on substrate layer results in the failure of several VNs since the resources are shared.
- *Network component*: the component of failure to be considered are link and node. Node failures are often due to maintenance, whereas link failures happen about ten times more than node failures.
- *Failure size*: both single and multiple failures can occur. The single failure case happens more often than multiple simultaneous failures.

To enhance the resilience of VN, pre-failure and post failure methods are proposed. Post failure methods react after the failure occurrence and start the backup restoring mechanism. When a failure occurs, the backup resources are allocated [2] and the affected network component is repaired progressively. However, some data loss is possible in the reactive case and the resource availability can not be guaranteed.

Pre-failure methods take into account the resilience before the failure and reserve resources for backup mechanism. In pre-failure methods, failure avoidance searches for a mapping of primary traffic with high reliability, while failure protection pre-computes backup path and reserves backup resource. The backup resources can be shared between failures or dedicated to a failure.

2.6.1 Architecture of survivable VNE

In [BHH+13], the authors introduce an architecture for resilient VN setup as well as the required enhancements to this architecture allowing sharing of the redundant resources among existing VNets. The VR requests are managed by an augmented RSVP-TE (Resource reSerVation Protocol Traffic Engineering) protocol. They introduce a new atomic activity to achieve the share protection. This activity takes as an input parameter the resources of different VNs. It is a sharing request to be sent from the VNO to the VNP, which can translate it to the InP. The InP evaluates the request, the results are returned to the VNP, which processes them and reports to the VNO. In case of shared protection, there is an additional information exchange required to determine which virtual protection paths are

[2] This method is also referred as *Restoration*.

allowed to share resources. Only protection paths belonging to disjoint working paths can share resources.

2.6.2 Node protection

In [YWK11], backup virtual servers are created dynamically and are pooled together to be shared between VNs to assure the requested reliability level. The higher the reliability level is, the higher number of needed backup nodes is. It is possible to share the backup nodes so that the total number of backup nodes is lower than the case where each VN separately has their own backup nodes. Every backup node can be a standby node for all other critical nodes.

With the Opportunistic Redundancy Pooling (ORP) mechanism, backup nodes can be shared between VNs as long as the reliability of every network is satisfied. The ORP shares these redundancies for both independent and cascading types of failures.

In [YAQS11], a two-step paradigm is adopted to fully restore a virtual network from any single facility node failure. It enhances a virtual network with backup virtual nodes and links requiring spare substrate resources, and then maps the enhanced virtual network to the substrate network. More specifically, [YAQS11] proposes two new approaches whereby an N-node virtual network is first enhanced to a 1-redundant and K-redundant virtual network with N+1 and N+K nodes, respectively, in addition to an appropriate number of redundant virtual links. The resource of redundant nodes are shared by protection. This method is referred as Failure Independent Protection (FIP), where the host nodes (1 or K) are assigned and dedicated to backup all working host (facility) nodes. That is, no matter which working host node fails, the affected task node will be migrated to backup host nodes, whereas other working hosts remain the same mapping.

In [QGH$^+$11][GQW$^+$14], an other approach called Failure Dependent Protection (FDP) is proposed. The differences between FIP and FDP are as follows. With FDP, each working host node can have a different backup host node under different failure scenarios. Upon a failure, the unaffected task node may be migrated from a working host node to its corresponding backup host node, as a result of re-embedding the entire task graph. In other words, FDP could provide more flexibility in survivable VN designing by allowing task nodes migrating freely after failure. Thus, FIP could be considered as a special case of FDP which is expected to use fewer resources at the cost of more task nodes migrations after a failure.

The ideal is that, when a node fails, its role may be replaced by any other nodes after a rearrangement of all the nodes (including the backup nodes) using graph transformation/decomposition and bipartite graph matching. A unique feature/requirement of FDP is that each virtual node may need to emulate multiple virtual nodes in the original VN, and

the virtual node for it to emulate depends on the failure. The disadvantage of this approach is that the large amount of possible migrations of working nodes after a failure makes the approach less applicable in large networks.

In [XWM+14], a recoverability-based VNE is used to allocate the VN with primary resources. When a substrate node fails, both the virtual nodes embedded on it and the virtual links whose hosting paths use it as an end or intermediate need to be remapped. The candidate backup node should have direct or indirect connections with the failure node within a certain geographical area. The recoverability is defined by the node and link resource in this area.

In VN primary mapping stage, the virtual nodes are embedded onto the substrate nodes with better recoverability to improve its node failure survivability. In the event of a substrate node failure, if the backup resources are not enough to remap all the affected virtual entities, the limited backup resources will be used to recover the VNs with higher penalties first, thus to reduce the total incurred penalty.

2.6.3 Link protection

2.6.3.1 Shared link protection

In [YSzXY11], the authors propose an algorithm, named RMap, for mapping of VNs with considering failures of substrate links. The substrate network is formulated as weighted graph by defining for each link its stress. Note that the latter quantifies the number of virtual links transiting through the substrate link. Afterwards, if a link stress is higher than a predefined threshold then RMap will compute its backup detour. When a link failure occurs, virtual links transiting over this substrate link will migrate to the backup links. Hence, the offered service will not be interrupted.

In [GWMT11], a shared link protection method is proposed. Each substrate link with primary flows is protected by a set of pre-configured bypass paths. After the node mapping, the primary link mapping and the protection of links are jointly computed by an ILP problem. The primary flows can be mapped to all the possible substrate links, whereas the backup paths are selected among the pre-configured paths.

Note that the primary and backup paths are both splittable. This means that upon a link failure, the upstream node needs to split the flow. This operation suffers from the out of order problem and is difficult to implement for an intermediate node on a path.

With the splittable and pre-configured backup path assumption, [GWMT11] also proposes an approach to compute statically the best primary and backup bandwidth separation before the arrivals of all the VNs.

2.6.3.2 Shared path protection

In [RB13], Rahman and al. propose a shared path protection method. A certain percentage of bandwidth resources on each substrate link is dedicated for backup purposes. Each virtual link is associated with a penalty. Before any VN request arrives, the InP pro-actively computes a set of possible backup detours for each substrate link. When a VN request arrives, [RB13] first compute the node mapping and primary link mapping by optimizing the primary resource utilization. After that, backup paths are selected through the optimization of virtual link penalty.

In [YAQ12], a shared path protection by node migration is proposed. If the link-disjoint backup path for a virtual link cannot be found, one end-node of this virtual link is relocated or migrated to another (backup) substrate node. In addition, all the virtual links connected with the migrated virtual node have to be remapped onto the substrate network. This method is effective to avoid bottlenecks, but difficult to implement.

In [WWW+14], backup topology simplification on substrate network is taken into consideration to reduce the total bandwidth constraints and actual resource consumption. A mininal backup topology for virtual network to protect against single link failure is found. The splittable formulation can be solved in polynomial time. For the unplittable case, a minimal tree based heuristic is proposed to solve the problem.

In [HZ14], a hybrid policy of survivable VNE is proposed. The profit is maximized and the service quality is improved. The authors define the meta-VN and propose a multi-embedding algorithm which embeds several meta-VNs instead of the original VN to improve the acceptance ratio and resource usage. Virtual links are reconfigured when they are damaged severely.

In [JK15], the p-cycle concept is applied to to protect against single node and link failure. Two approaches are proposed: (1) p-cycle-based path protection consists of protecting each embedding path by a disjoint path defined by a set of candidates p-cycles; (2) p-cycle-based virtual segment protection consists of transferring the protection of an embedding path into the protection of a set of virtual segments, where each virtual segment covers an intermediate node and two of its adjacent links used by this embedding path. To select the best set of p-cycle, a large-scale optimization approach based on the column generation optimization technique is proposed.

2.6.3.3 Dedicated path protection

In [ZPC11] [ZCP12], a dedicated path protection method is proposed. For each virtual link, two substrate disjoint paths are computed, one for primary mapping and the other for

backup mapping. Three objectives are taken in considerations: (1) The delay requirement is minimized; (2) The number of substrate nodes, which is used only by backup path, is minimized: in this way, the backup path are more likely to reuse the intermediate nodes used by primary links; (3) The authors avoid that a substrate link is used by two virtual links of the same virtual network so that a link failure affects less virtual links of a VN.

In [WWC14], a network coding based link mapping, which provides instantaneous recovery with the least redundant resources, is proposed. The network coding mechanism aims to find the maximum link-disjoint paths satisfying the bandwidth constraints. Assume that the endpoints of a virtual link have been mapped to u and v via multi-path (splitting). Another link disjoint path is reserved for backup. At the sender node, the coded data is transmitted additionally over the backup path. At the receiver node, if a link failure happens on one primary path, the missing data can be given directly by coding the information on the other primary paths, and taking the difference with the protection path information.

In elastic optical networks [LDZ+14] [YCL+15], the link disjoint paths are used to protect virtual links. The authors optimize the joint failure probability of the pair of primary and backup paths. A full-connected substrate graph is first constructed with the weight of joint path failure probability. For a virtual network request, virtual links are mapped one by one to the most reliable joint link path. All the possible mapping orders are computed, so it is difficult to apply it in large networks.

In [GDTV14], the virtual request is characterized by a reliability constraint. The objective is to optimize the CPU and bandwidth utilization and at the same time satisfy the reliability constraint of the nodes and links. Each substrate node and link is associated with a reliability value. In order to achieve the required reliability, the virtual graph is expanded by shared/dedicated backup method. The compound reliability of a substrate path for different methods is computed. The reliability constraints are added to the new VNE problem with the mapping of expanded VN graph.

2.6.4 Multiple failure protection

In [LML10], a probabilistic model is proposed to deal with Shared Risk Link Group (SRLG) failure. An SRLG is a set of links sharing a common physical resource (cable, conduit, etc.) whose failure affects all the links on the SRLG. The authors propose a probabilistic SRLG (PSRLG) that enables to effectively model correlated link failures. The authors develop mathematical formulations for the problem of finding a pair of paths with minimum joint failure probability. This approach enables the generalization of disjoint path protection schemes to the case of multiple probabilistic failures. The algorithms are based on linear approximations and Lagrangian relaxations, and are close to the optimums.

In [YQA+10], a single region failure recovery method is proposed. The failure of multiple adjacent substrate nodes/links is considered. If a substrate node initially allocated for a virtual node is within the failed region, another substrate node outside the region is pre-allocated to restore the affected node. By assuming that there are N region failures on the substrate network, their framework generates N+1 mapping solutions, one involving the initial primary mapping, and the others involving backup mappings (one backup solution for each region failure). The resources allocated to the solutions are shared since they consider the single region failure.

In [HMT16], two ILP formulations are proposed to provide content connectivity (CC). The first one guarantees the content connectivity in case of double-link failures (CC2) and the second one provides network connectivity against single-link failures and maintains content connectivity after double-link failures (NC1+CC2). The authors show that in case of single-link and double failures, content connectivity can be maintained with less network resources than network connectivity.

2.6.5 Failure avoidance

In [SFAM13], SN and VN are assumed to be composed of access nodes and backbone nodes. The virtual access routers and virtual links connecting them are first embedded. The remaining VN topology is subdivided into several star topologies that are embeded by Bee colony heuristic. The employed bees use the Dijkstra algorithm based on the metric which takes into consideration i) the residual resources and ii) the reliability of physical resources. The reliability metric is based on the age of physical nodes and links. Besides, for load balancing within the SN , the employed bees run a combinatorial multi commodity flow algorithm. The advantage of failure avoidance is that it does not allocate any backup resources for clients.

In [WLQL16], the historical failures statistics are taken into consideration to map the VN requests onto the part of substrate network which has fewer latent regional failures. A disaster-prediction scheme is proposed to evade large amounts of redundant backup resources against the multiple regional failures. Two mapping algorithms based on the regional failure model are provided.

In [GBM+16], relative disjoint paths are generated to consider the reliability of VN, as well as the bandwidth used by it. The algorithm focuses on varying the percentage (p) of disjoint path established between the source node and the set of destination nodes to find the best solution. $p = 0$ is the no redundancy case, which means that the topology will have a single path to each destination node. On the other hand, if $p = 1$, which is the full redundancy case, the topology will have two fully disjoint paths to each destination node.

Similarly, $p = 0.5$ is the case where half of links in primary path will be the basis for the secondary path.

To determine the relative disjoint path, The path definition algorithm allocates the shortest path with higher available bandwidth and lower failure risk. In this way, bandwidth resources are saved and the allocation of high risk components is avoided.

2.6.6 Post failure recovery

In [PLG+16], a progressive post-fault recovery schemes to judiciously schedule repair resources is proposed. The goal is to intelligently place node and link repair resources to rapidly recover failed VN loads and hence reduce service downtime/penalties. An initial multi-failure event occurs and is followed by a series of recovery stages. During these recovery stages, all failed nodes and links will either transition directly from an initial failed state to a fully-recovery state or through a partially-recovered state.

Based upon the above, all failed and partially-recovered nodes and links are deemed eligible to receive repair resources. The progressive recovery schemes first distribute available repair resources among failed nodes/links and then use any standard VNE algorithm to re-map failed demands. The scheme places node repair resource to eligible nodes which have at least one non-failed neighbour node. Similarly, link repair resources are only placed at failed or partially-recovered physical links with non-failed endpoints. In this way, the number of restored VN requests in each stage are maximized.

2.6.7 Analysis

The protection methods are classified in Table 2.2. This table summarize the survivable VNE frameworks in terms of concerned network components, the category of method to enhance the survivability, the type of failure to deal with and the optimization metrics.

A network is composed of physical nodes and physical links. Node migration principle is often used to protect nodes from failure. If a substrate node fails, the primary resources used on this node are migrated to another pre-selected substrate node. In other words, the pre-selected backup node replace the function of the failure node. Specifically, in VN environment, the virtual network topology is enhanced to include the backup virtual for each primary virtual node. The initial VN mapping problem is then transformed to the mapping of a new enhanced VN topology. This topology consists of not only the backup nodes, but also the backup links so that the connection between nodes will not be cut off.

Another component of the network is the links. A single substrate link failure results in the failure of several virtual links (networks). The characteristics of VNE lead to the following consideration of survivability:

- Since a virtual link is mapped to one or multiple substrate paths, the choice of the substrate links on each path is critical for the survivability. This is the original of the failure avoidance method.

- After the determination of primary path, it is easy to consider a backup path to protect the primary path against the link failure. Since each substrate link on the path can fail independently, the backup path can not pass through any substrate link of the primary path. This observation result in the disjoint path protection (the primary and backup path are disjoint).

- The shortcoming of the path protection is that if a substrate link on the path fails, the failure notification need to go back to the source node and the source node retransmits the flow. It result in a long recovery time, which is not a good phenomenon for some delay sensitive service. The link protection is proposed to deal with this problem. In link protection, each protection has a backup path. If the link fails, the flow on this link is immediately rerouted to its backup path from the source node of this link (instead of source node of the path in path protection). This method is also referred as local protection or fast recovery.

The mentioned survivability method are adapted to different kinds of failures. The single failure is the most common failure. Since there is only one failure on the network, the backup resources do not need to be active simultaneously. The share backup principle can be applied in this case. An amount of backup resource can be allocated to protect against the failure of multiple component since they do not fail simultaneously. If any one of these components fails, the backup resource is activated. After the repair of the failure component, the resource are switched back to primary component.

Multi failure is more complex and difficult than single failure. Failure avoidance method can be adopted to multiple failure, whereas protection methods are difficult to be applied. To protect against multiple failure, the failures need to be first identified. The multiple failure frameworks search for the backup solution for each multiple failure.

The optimization metric is also an important issue for survivable VNE. Node CPU resource and link bandwidth resource are most common objectives. The CPU and bandwidth are often associated with the long term profit of SP, so it is important to improve the profit of VN mapping. However, the failure of of VN will lead to the degradation or disruption of

service, which will result in some penalty to the profit. In survivable VNE environment, other metrics such as delay, probability of failure and penalty need to be taken into consideration.

By examining the existing work of survivable VNE, we find that there are few works concerning the optimization of failure probability, which is, on one hand, a good measurement of survivability level, and on the other hand, can be combined with the failure avoidance technique to easily configure a survivable virtual network. Moreover the combination of protection and avoidance is rare. In fact, protection is a high level survivability method with high resource consumption and complex configuration, whereas avoidance does not need extra configuration and can be a good supplementary method.

2.7 Conclusion

In this chapter, we introduce the background of this thesis: network virtualization, virtual network embedding, multi domain VNE and survivable VNE. The single domain VNE problem can be solved in two stage or one stage. Since the multi-domain VNE is more complex than single domain VNE, it is preformed in 3 steps. In survivable VNE frameworks, the backup resource is determined independently after the primary mapping (which is a single domain VNE).

We find that there are some shortcomings in existing works. We particularly notice, the efficient inter domain link mapping, the failure avoidance method with failure probability consideration, the verification of the existence of backup path and the combination of failure avoidance and protection. In the following chapters, we will try to propose some methods to solve these problems.

	component	category	type of failure	objective, constraints
Yeow [YWK11]	node	protection	single	cpu, bandwidth
Yu [YAQS11]	node	protection	single	cpu
Qiao[QGH+11] Guo[GQW+14]	node	protection	single	cpu
Xiao [XWM+14]	node	protection, avoidance	multiple	cpu, bandwidth
Yu [YSzXY11]	link	protection	single	cpu, bandwidth
Guo [GWMT11]	link	protection	single	bandwidth
Rahman [RB13]	link	protection	single	penalty
Yu [YAQ12]	link	protection	single	bandwidth
Zhang[ZPC11] [ZCP12]	link	protection	single	delay
Wang [WWC14]	link	network coding	single	bandwidth
Luo[LDZ+14] Yang[YCL+15]	link	protection	single	probability
Jarray [JK15]	node, link	protection	single	bandwidth
Lee [LML10]	link	protection	multiple	probability
Yu [YQA+10]	node, link	protection	multiple	bandwidth
Wang [WLQL16]	node, link	avoidance	multiple	
Soualah [SFAM13]	node, link	avoidance	single	age of equipment
Hmaity [HMT16]	link	protection	double	network connectivity
Wang [WWW+14]	link	protection	single	bandwidth
Hu [HZ14]	link	protection	single	bandwidth
Guerzoni [GDTV14]	link	protection	single	bandwidth with reliability constraints
Gomes [GBM+16]	node,link	protection, avoidance	multiple	bandwidth, reliability
Pourvali [PLG+16]	node, link	post-failure	multiple	cpu, bandwidth

Table 2.2 Protection method comparison

Chapter 3

Multi-domain VN Resource Allocation

3.1 Introduction

With the rapid development of cloud computing, services across multiple domain turn into an intuitive evolution. The network infrastructure is organized by many different administrative domains. Different domains managed by infrastructure provider (InP) on the same level are connected to each other's network via peering links.

Network virtualization is well explored in single domain, but few attentions have been paid to the creation of VN over multi-domain. Since the Internet is a multi-domain network, the implementation of VN on the Internet needs to consider the case of multi-domain. In addition, VN over multi-domain allows service provider (SP) to deliver enhanced VN-based services crossing multiple InPs. Consequently, efficient algorithm should be developed to provide and improve the overall VN mapping over multi-domain.

Establishment of multi-domain VN is more difficult than the one on single domain for at least two reasons:

- First, a single domain VNE problem is mainly solved by linear programming (LP). If we have a complete vision of all the domains, a multi-domain VNE could be considered as a single domain VNE with a very large domain, so computationally harder to solve.

- More importantly, for various reasons (technical, commercial, etc.), the acquisition of full information in multi-domain is costly and often not possible. Only limited information is exchanged between InPs via protocols like BGP. This kind of information is usually static over a long period since frequent exchange of real-time information would significantly increase the network load. As a result, single domain approach cannot be re-used and InPs should cooperate to meet the virtual request.

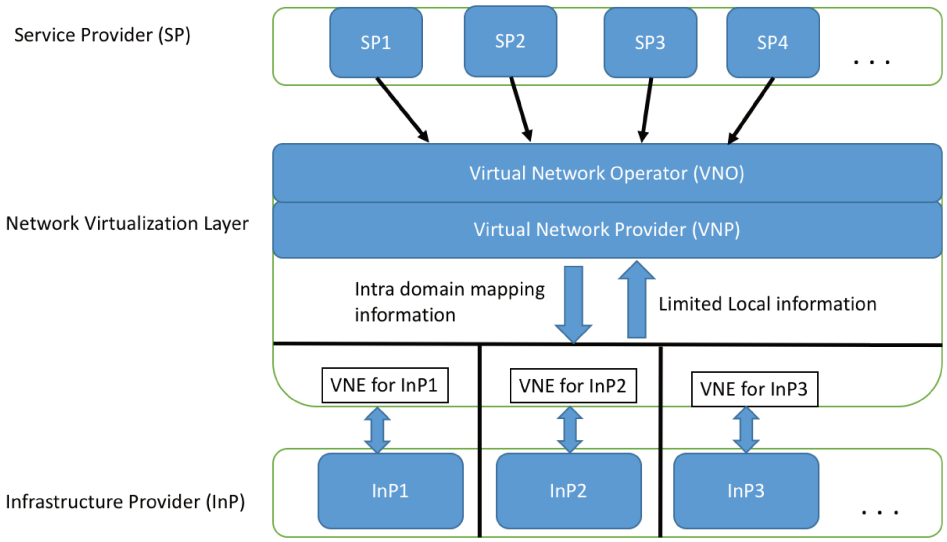


Fig. 3.1 Multi-domain network virtualization architecture

To address these challenges, multi-domain VNE frameworks are developed. The architecture of multi-domain network virtualization shown in Figure 3.1 is similar to the single domain architecture (Section 2.2.3). The key actor of multi-domain network virtualization is the VNP (*Virtual Network Provider*). The role of this VNP consists in assembling limited local information of the domains, decomposing VN and achieving the multi-domain VNE.

In order to embed a VN to multiple domains, there is usually a VN decomposition (partitioning) step followed by local sub VN mapping in each InP. A virtual node is mapped to one substrate node located in one of the domains in a way that the mapping cost is optimized. Since virtual nodes are mapped to different domains, some virtual links are *inter-domain virtual links* (cross at least 2 domains) and others are *intra domain virtual links*. The latter can be mapped by single domain VNE algorithm but the former becomes an issue. Since the inter-domain information transmitted by VNP to the hypervisor of each domain is incomplete, the inter-domain link mapping may not be optimized. In consequence, the overall link mapping optimization cannot be guaranteed.

Existing solutions mainly focused on the decomposition (partitioning) of a multi-domain VN to several intra-domain VNs. One of the shortcomings in these frameworks is the lack of efficient link mapping method especially for the inter domain virtual links. Substrate peering nodes and links on the path of inter domain virtual links need to be optimized.

To cope with these shortcomings, we describe and propose in this chapter an efficient method for link mapping in the context of multi-domain VNE. Our method jointly considers the mapping of intra and peering links to improve the link mapping. In our approaches, the

peering links are mapped simultaneously along with the intra domain links. It is easy to be deployed in current Internet architecture, since it only uses the information that is usually disseminated by classical routing protocols like BGP. In addition, as shown by the simulation, our method improves the utilization of substrate resources.

The rest of this chapter is organized as follows. Section 3.2 provides a general view of our method by comparing it with the existing methods. Section 3.3 describes the network model we adopt. Section 3.4 and section 3.5 present and detail our multi-domain VNE solution. The evaluation results are shown in section 3.6. Section 3.7 concludes this chapter.

3.2 Our method and existing solutions

In this section, we describe the basic operation principle of our method [LSC16b] and show the differences with the existing methods. We give a table (Table 3.1) and a comprehensive example (Figure 3.2-3.9) that shows the difference between the methods.

3.2.1 Position of problem

In Chapter 2, we affirmed that the classical mapping procedure is composed of 3 steps: VN partitioning, inter-domain connection and sub VN mapping. After analyzing the existing solutions, we found that in inter-domain connection stage, peering links always use the same path until some peering links are overloaded. This phenomenon will result in difficulty of later sub VN mappings in all the domains. The sub VN mappings have to either choose a costly intra domain mapping, or reject the VN request if no mapping solution is found.

In contrast, our approach [LSC16b] maps simultaneously the peering links along with intra domain links so that the phenomenon described above can be largely weakened. The inter-domain connection step and sub VN mapping step are jointly performed by our down-sizing algorithm.

Table 3.1 shows that our method [LSC16b] uses the same VN partitioning strategy as Shen's proposition [SXYC14]. However, inter-domain connection and sub VN mapping are merged together in our method. The information model we adopted to allow the link mapping computation consists in length based path costs.

3.2.2 Examples

Let us illustrate the operation of our proposition by examples. In Figure 3.2, three InPs are shown with their substrate nodes from A to P. Substrate node set $\{B,C,D,E,F,G,H,I,L,M,O,P\}$ are peering nodes. They are connected via 2 or 3 peering links. Intra substrate links are not

	VN partitioning	inter-domain connection	sub VN mapping	information model
Houidi [HLAZ11]	maxflow/Exact	VPN connection	Exact VNE	functional attributes
Dietrich [DRP13b]	Exact		single domain VNE	peering link graph
Guo [GWQ+15]	Particle Swarm Heuristic			
Shen [SXYC14]	**node mapping**	single shortest-path	single domain link mapping	full mesh graph
Li [LSC16b]		**downsizing mapping**		**length based path cost**

Table 3.1 Method comparison

depicted. We assume that a VN request of 4 nodes $\{a,b,c,d\}$ and 4 links $\{a-b,b-c,c-d,d-a\}$ arrives.

The virtual nodes are assigned to the domains in VN partitioning step as shown in Figure 3.3. Virtual nodes a and c are assigned to $InP1$ and $InP3$ respectively, whereas b and d are assigned to $InP2$. Virtual links $a-b$, $a-c$ and $c-d$ are inter domain virtual links, whereas $b-d$ corresponds to an intra domain link.

The information model used in Dietrich [DRP13b] and Guo [GWQ+15] corresponds to the peering link graph that is shown in Figure 3.4. This peering graph is composed of peering nodes, peering links ($B-F$, $D-L$, etc.) and augmented links between peering nodes ($D-E$, $G-H$, etc.).

The peering graph allows to compute the VN segment, which is shown in Figure 3.5. The virtual nodes a, b, c and d are first mapped to the peering nodes C, F, P and I respectively. The virtual peering links $a-b$,$d-c$ and $c-a$ are then deduced and mapped in our example to $C-F$, $I-H-P$ and $P-L-D-C$ respectively. In this way, all the egress nodes that are used to embed the inter domain virtual links are determined. For example, $a-c$ selects the egress node L whereas $c-d$ uses the egress node P.

After VN segment computation step, a single domain VNE problem is solved in each domain. The final mapping is shown in Figure 3.6, where $InP1$ maps a to A and establishes a substrate path between A and C, $InP2$ maps b and d to K and J respectively. Substrate paths $F-K$, $K-J$, $I-J$ are established. $InP3$ maps c to N and establishes the path $N-P$.

The information sharing scheme proposed in Shen [SXYC14] is shown in Figure 3.7. All the nodes belonging to the same domain are interconnected with the length-based shortest paths. For the mapping of virtual links interconnecting different domains, only one peering link is selected according to the shortest path algorithm. In this way, the multi-domain VNE

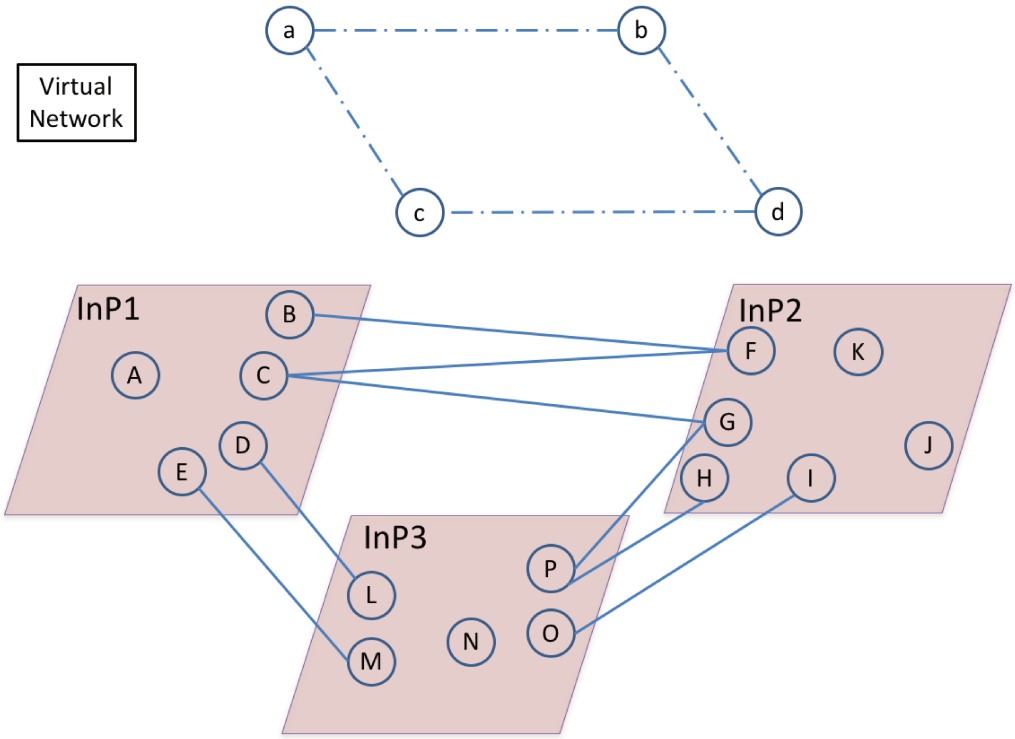


Fig. 3.2 Multi-domain and VN

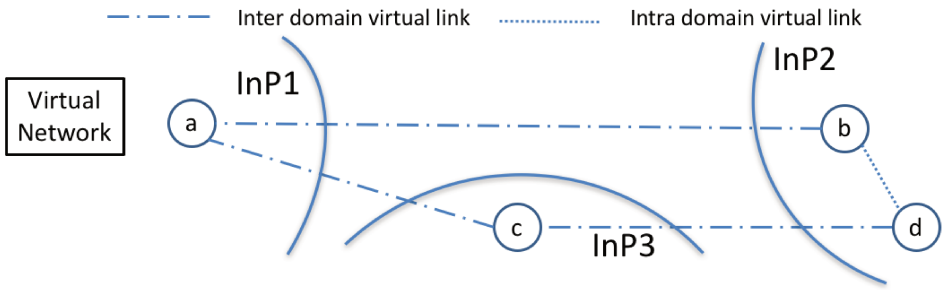


Fig. 3.3 VN partitioning

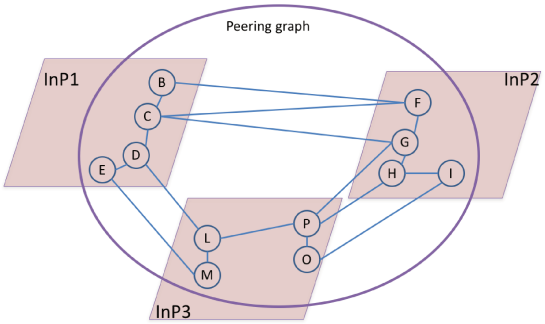


Fig. 3.4 Peering graph

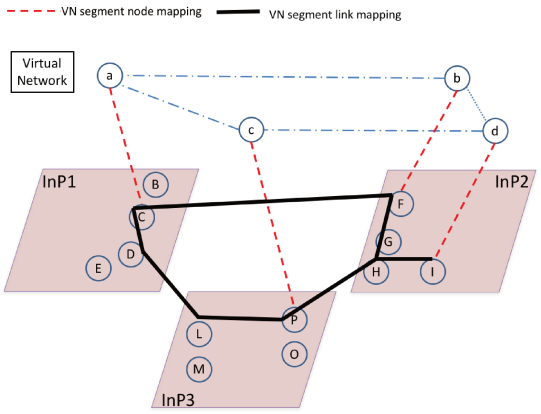


Fig. 3.5 VN segment

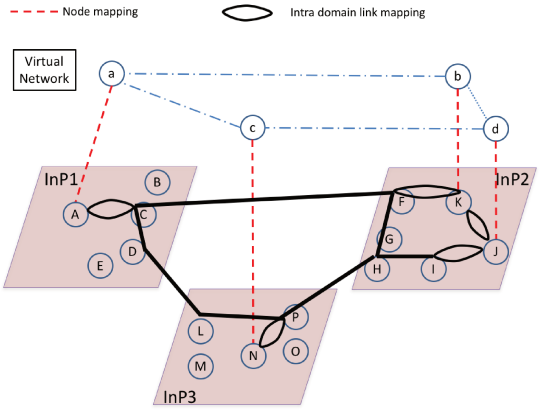


Fig. 3.6 Dietrich [DRP13b] and Guo [GWQ+15] link mapping

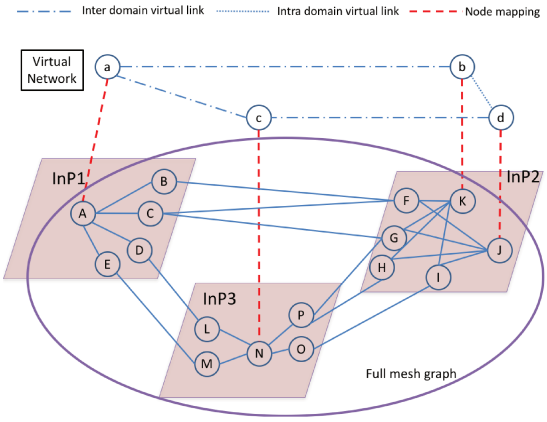


Fig. 3.7 Full mesh graph

is splitted onto various intra domain VNE instances. In Figure 3.7, the virtual link $a-c$ is mapped to the shortest path $A-D-L-N$. This path traverses one peering link $D-L$ connecting InP_1's border node D to InP_2's border node L. In this way, the virtual link $a-c$ is transformed into two sub virtual links: $a-c'$ in InP_1 where c' is mapped to border node D and $a'-c$ where a' is mapped to border node L.

After VN splitting, an intra domain VNE algorithm is performed in each domain to solve the problem. The mapping results are shown in Figure 3.6 and 3.8 where $A-C$, $A-D$, $N-L$,$N-P$, $K-F$, $K-J$ and $H-J$ are splittable intra domain link mappings.

In order to compare with existing methods, we show our solution in Figure 3.9. Different from the existing solutions, the intra-domain and peering link mapping can be both splittable. For example, $a-c$ is mapped to $A-D-L-N$ and $A-E-M-N$. The path $L-N$ and $M-N$ can also be multi-path. In addition, the peering links selected for link mapping

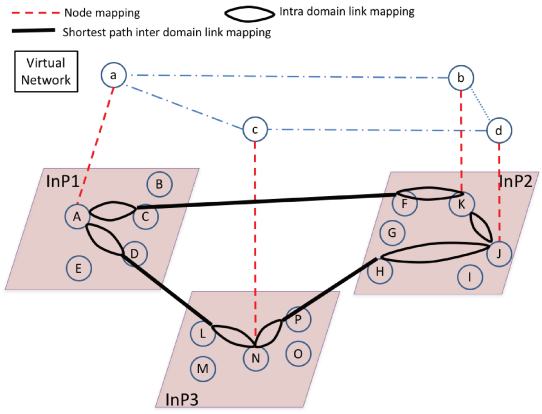


Fig. 3.8 Shen[SXYC14] link mapping

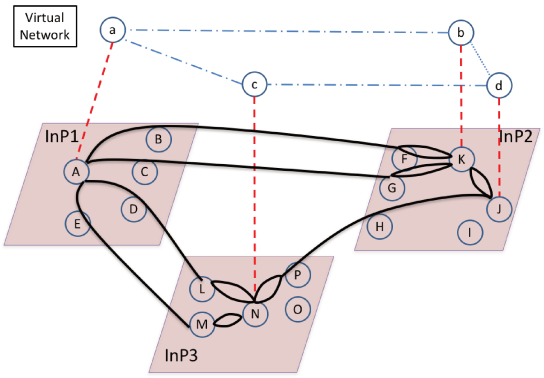


Fig. 3.9 Li[LSC16b] link mapping

are jointly determined with the intra-domain links so that the link resource utilization is improved.

3.3 Multi Domain Model

In order to handle multi domain VNE, we extend the substrate network model presented in Section 2.3. The virtual network model remains the same.

3.3.1 Substrate network

A domain InP_i is modelled as an undirected graph $G_i^S(N_i^S, L_i^S)$, where N_i^S is the set of substrate nodes in domain i, L_i^S is the set of internal (intra) substrate links. Each substrate node $n_i^s \in N_i^S$ is associated with a CPU capacity $cpu(n_i^s)$ and a geographic location $loc(n_i^s)$. Each substrate link $l_i^s \in L_i^S$ is associated with a bandwidth capacity $bw(l_i^s)$.

Assuming that the substrate network spans K domains, there are some peering nodes (border nodes) which have peering links with other domains. The notations are as follows:

- The peering nodes set: N_i^{SP}, $N_i^{SP} \subset N_i^S$;
- The peering links between InP_i (*i.e.* G_i^S) and InP_j (*i.e.* G_j^S): L_{ij}^S;
- the set of all of the peering links of InP_i:

$$P_i^S = \bigcup_{j=1}^{K} L_{ij}^S$$

- the set of all of the peering links P^S:

$$P^S = \bigcup_{i=1}^{K} P_i^S = \bigcup_{(i,j)\in(1...K)^2} L_{ij}^S$$

The complete substrate network $G^S(N^S, L^S)$ is thus obtained as follows:

$$\begin{aligned} N^S &= \bigcup_{i=1}^{K} N_i^S \\ L^S &= (\bigcup_{i=1}^{K} L_i^S) \bigcup P^S \text{ with } P^S = \bigcup_{i=1}^{K} P_i^S,\ P_i^S = \bigcup_{j=1}^{K} L_{ij}^S \end{aligned}$$

3.3.2 VNP layer model

VNP collects information provided by InPs. We assume that InPs provide exact information about their nodes, as well as the peering links. On the contrary, there is no exact information about the internal organisation of a domain. Similar to the solution in [SXYC14], we assume that this information is given by InP for each couple of <node, peering node>, as if there was a *pseudo* direct link between these two nodes. Denote the set of these links by $L_i^P = \{l_{mn} \,/\, m \in N_i^S, n \in N_i^{SP}\}$, InP_i provides to VNP the set of link cost C_i^P defined by

$$C_i^P = \{C(l_{mn}) \,/\, m \in N_i^S, n \in N_i^{SP}\}$$

where $C(l_{mn})$ represents a cost (distance, bandwidth, etc.) characterizing the link l_{mn}. This kind of information is actually what a routing protocol like BGP reports to other Autonomous System (AS).

Thus, the SN of an InP_i is *perceived* by VNP as a graph $G_i^P = (N_i^S, L_i^P)$. In this way, the whole substrate network that VNP perceives, referred as G^P, is defined as follows:

$$G^P = (\bigcup_i G_i^P) \bigcup P^S$$

i.e. the perceived vision of each domain and the exact vision of the inter-domain connections. With G^P, VNP can establish a kind of complete topology covering all the domains for achieving VN decomposition and link mapping.

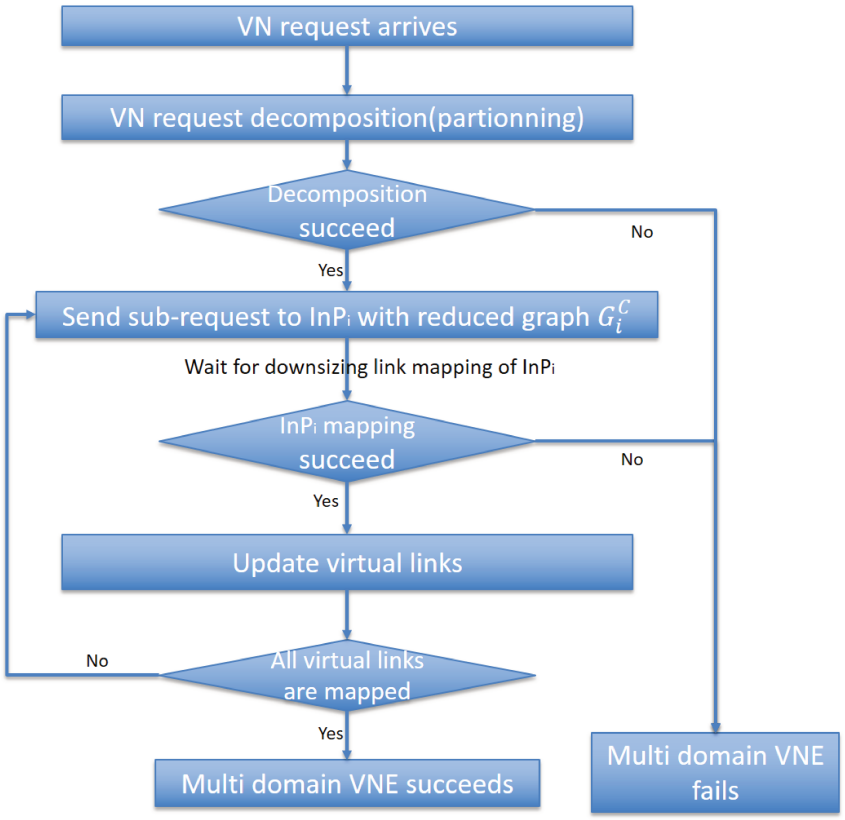


Fig. 3.10 VNP workflow to embed a VN

3.4 Our proposition

To solve VNE in the context of multi-domain, we propose a novel algorithm that maps jointly intra and peering links.

In our solution, we handle each VN request with a 2-step process:

- At the first step, VNP performs the node decomposition optimizing the node embedding.
- Subsequently, VNP performs a series of iterative downsizing VNE sub-solution, each of them optimizes both the intra and peering link mapping related to a domain.

The link mapping is determined, at each iteration, by the acting InP (called *mapper*). VNP is in charge of providing necessary information to the mapper. The generic work-flow of our algorithm is given by Figure 3.10. The details are explained as below.

3.4.1 Decomposition

Firstly, VNP decomposes the VN request with objective of minimizing the node mapping cost. In this stage, VNP associates each virtual node with a candidate set of substrate nodes that meet its location constraint $loc(n^v)$. VNP is free to use any multi-domain VN partitioning method (*e.g.* the methods presented in [HLAZ11][SXYC14]). At the end of this stage, virtual nodes are embedded to substrate nodes located in different domains.

An example of VN decomposition is shown in Figure 3.11. Three InPs are shown with their substrate nodes from A to P. They are connected via 2 or 3 peering links. Intra substrate links are not drawn. We suppose that a VN $\{a,b,c,d\}$ arrives. The VN decomposition step tells us that a, b, c and d are mapped to substrate nodes A, K, N and J, respectively. $\{a-b,a-c,c-d\}$ are virtual links which interconnect two different domains, while $\{b-d\}$ locates in only one domain.

3.4.2 An iterative downsizing VNE approach

Here we give a detailed presentation of the kernel of our proposal, which is formally given by Algorithm 1.

3.4.2.1 Rationale

After VN decompostition step, since there is no domain who knows the complete information of any other one, embedding the virtual links which interconnect two different domains becomes an issue.

We notice that, VNP can build, for each InP_i, a reduced vision (denoted by G_i^R) from G_i^P. This vision contains all the peering links/nodes, as well as the substrate nodes on which a virtual node is embedded. Formally, $G_i^R = (N_i^R, L_i^R)$ where

$$N_i^R = N_i^{SP} \bigcup \{n_i^S \in N_i^S \ / \ \exists n^v \in N^V, M(n^v) = n_i^S\}$$

i.e., N_i^R is the union of all the substrate nodes supporting virtual nodes on domain i and all of its peering nodes. In a similar way, we define L_i^R as follows;

$$L_i^R = \{l_{mn} \in L_i^P / n \in N_i^R, m \in N_i^{SP}\}$$

i.e., L_i^R is the subset of L_i^P between N_i^R and N_i^{SP} containing only the links interconnecting a peering node and a node supporting a virtual node.

In order to achieve an efficient and pragmatic operation mode, we prefer that VNP plays its role of coordinator: It is VNP who decides which of the InP should have the privilege to map its peering links with others. It is also VNP who provides to the chosen InP (that we refer as *mapper*) the topology of the rest of the network according to its perception. In other words, the chosen InP (the mapper) extends its view to the rest of the network, by using the vision provided by VNP, the only one who has a kind of comprehensive view on all domains. In this way, the mapper obtains an augmented graph on which it will perform link mapping, including both its intra and peering links.

This process continues, domain after domain, until all of the virtual links are set. The selection criterion is the link utilization, the InP has most stringent link utilization will be the first to map its peering links. The reason lies in that high link utilization denotes more constraints in the path choice.

3.4.2.2 Building of the augmented graph

Let InP_i be the chosen mapper. Formally speaking, the vision of the other domains provided by VNP is $G_i^C = \bigcup_{j \neq i} G_j^R$, i.e. the reduced perceived vision of all the other domains. We only need to consider the case where all the domains are adjacent to the mapper. The case of a domain not adjacent to the mapper but to which the mapper has virtual links can be reduced to the adjacent case. Actually, assume that the mapper has to establish a virtual link with a node $c \in InP_s$ and InP_s is not adjacent to the mapper. We have necessarily one of the two situations:

- The node c is known by *none* of the adjacent domain to the mapper, then we know that it is technically impossible to establish the virtual link, since there is no way to route traffic between the mapper and c. This leads to the failure of VNE embedding;
- There is at least one adjacent InP, say InP_j who knows c, it means that InP_j can include c among its *pseudo* direct link announcement, thus, c can be considered by VNP as being attached to InP_j and announced to InP_i through G_i^C.

VNP communicates G_i^C to the mapper (InP_i) so that the latter can create an augmented graph G_i^A defined as follows:

$$G_i^A = G_i^S \cup P_i^S \cup G_i^C$$

In other words, the mapper has a complete knowledge of its own domain (G_i^S) as well as its peering links (P_i^S). The augmented topology of the whole network is obtained by using information (G_i^C) communicated by VNP. This topology covers all of the accessible domains and can be used as a substrate graph on which the mapper performs VNE.

3.4.2.3 VN sub-request

VNP asks the mapper to perform a partial VNE, which concerns only the virtual links related to the mapper. We refer this partial VNE as a *sub-request* (L_i^{subv}). It is obtained from the current VN request by reducing it to virtual links related to the mapper.

InP_i deduces and transforms the inter domain VN request to a local VN request as follows:

- Any intra domain virtual link $x-y$, where x and y are mapped to substrate nodes belonging to InP_i are added to the local VN request. For instance, virtual link $b-d$ is added to the local VN request of InP_2 in Figure 3.13.
- Any inter domain virtual link $x-y$, where x and y are mapped to substrate nodes X and Y ($X \in InP_i$ and $Y \notin InP_i$), is replaced by one or several intra domain link $x-y'$ where y' is mapped to the extremity node of the peering link that is used to interconnect nodes X and Y.

3.4.2.4 An MCF-based link mapping

At this stage, the mapper gets an augmented vision of the whole substrate network, and a VNE sub request (L_i^{subv}), both from VNP. We have thus a classical VNE problem that we solve with the multi commodity flow (MCF) based mapping algorithm (line 6 of Algorithm 1).

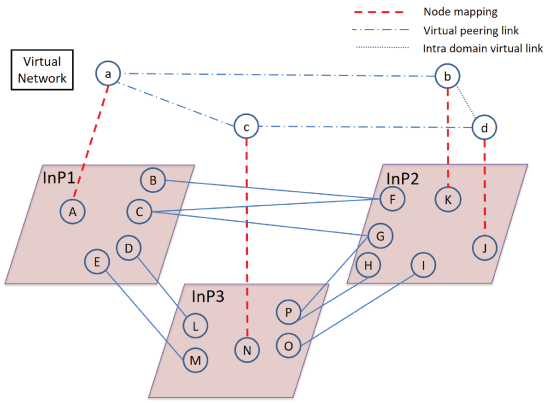


Fig. 3.11 VN decomposition

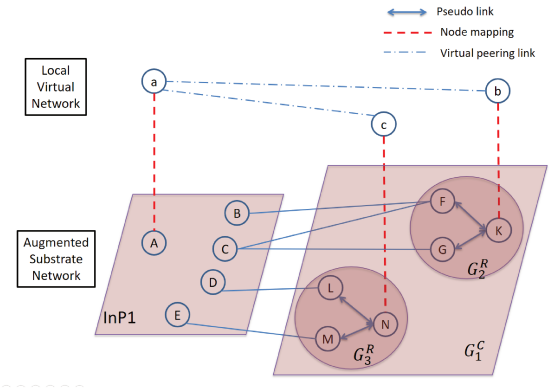


Fig. 3.12 downsizing link mapping by InP_1

At the end of this step, InP_i pre-allocates resources on the intra and peering links related to it and sends to VNP a virtual link update notification.

Let us illustrate the operation of our proposition (Algorithm 1) by our example. Assume in Figure 3.11 that InP_1 is chosen as the first mapper. VNP builds the VNP-level graph vision $G_1^C = G_2^R \cup G_3^R$ (see Figure 3.12) with $G_2^R = (\{F,\ G,\ K\}, \{F\text{-}K, G\text{-}K\})$ and $G_3^R = (\{M,\ N,\ L\}, \{M\text{-}N, L\text{-}N\})$. It builds also the sub-request $L_1^{subv} = \{a\text{-}b, a\text{-}c\}$ in which the virtual links b-d and c-d are pruned since they do not have any extremity node supported by a substrate node in InP_1. VNP then sends G_1^C and L_1^{subv} to InP_1. The latter builds the augmented graph G_1^A which includes G_1^S (all the nodes and links in InP_1), the peering links (B-F, C-F, C-G, D-L E-M), and G_C^1. InP_1 then applies the MCF-based algorithm to solve the embedding of L_1^{subv} on G_1^A.

3.4.3 Update and iteration

After each sub-request, the mapper (say InP_i) reports the results. In particular, it gives the results of the mapping of all its inter-domain virtual links in the following manner.

Let $l^v(u,v)$ be the virtual link between given nodes $u \in InP_i$ and $v \in InP_j$, with $bw(l^v(u,v))$ as the required bandwidth. The MCF algorithm maps $l^v(u,v)$ into one or several paths. Denote by N^F the set of the peering nodes of InP_j through which a fraction of $l^v(u,v)$ is mapped. After the link mapping of InP_i, the set of virtual links $\{l^v(w,v)\}_{w \in N^F}$ is equivalent to the virtual link $l^v(u,v)$ with bandwidth demand:

$$\sum_{w \in N^F} bw(l^v(w,v)) = bw(l^v(u,v))$$

It is to be pointed out that these links are totally inside inP_j and they replace $l^v(u,v)$.

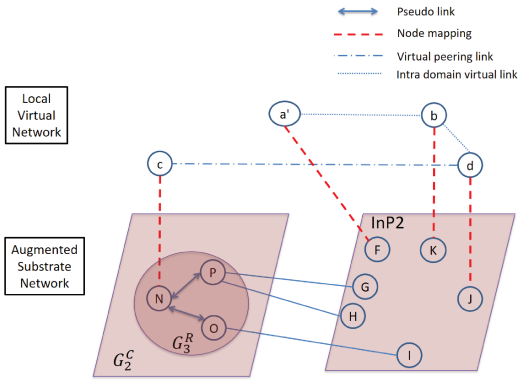


Fig. 3.13 downsizing link mapping by InP_2

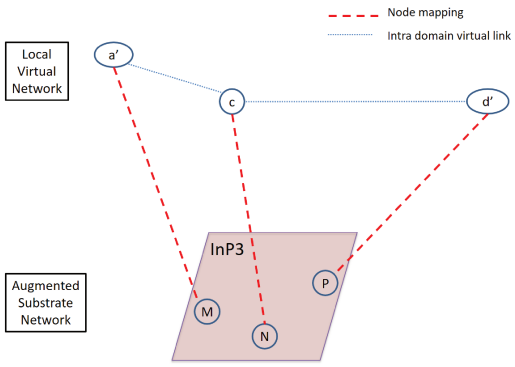


Fig. 3.14 InP_3, end of iteration

As the mapping of InP_i is achieved, it will no longer appear as domain in the subsequent problem which contains only the remaining domains. However, the achieved mapping concerns only the links related to InP_i (intra as well as peering), the part of inter-domain virtual links on the other domains still has to be mapped. Each of such inter-domain virtual link related to the mapper can be transformed into the above described equivalent set which will be added to each concerned domain. For the sake of reading simplicity, we prefer to give an informal description here, instead of a formal one, which would generate some more heavily-indexed notations.

In this way, we obtain a new VNE problem with:

- at the SN level, the retreat of InP_i and all the peering links related to it;
- at the VN side, the retreat of all the virtual links internal to InP_i and the replacement of all the inter-domain virtual links related to InP_i by their equivalent set which are added to corresponding domains.

This allows us to run iteratively the downsizing mapping described in Section 3.4.2. VNP repeats the process till its convergence which is *certain*, since the subset is reduced by at least one domain (the mapper) at each iteration.

In the example of Figure 3.11 and 3.12, assume that InP_1 has chosen link *F-K* to map virtual link *a-b*. After sending its results to VNP, this latter deduces and creates a new virtual link *a'-b* with node mapping *a'* equal to *F*. This virtual link *a'-b* replaces virtual link *a-b*.

Now, the new problem (Figure 3.13) contains only InP_2 and InP_3. Assuming that InP_2 is chosen as mapper, the same process continues and our problem is eventually reduced to a single domain (Figure 3.14) which is the last step of our algorithm.

Algorithm 1: Link mapping of InP_i as mapper

Input : sub request virtual links L_i^{subv}
Input : reduced perceived graph G_i^C
Output : virtual link update notification

```
begin
    if L_i^subv = NULL then
        return
    end
    create augmented substrate network G_i^A(N_i^A, L_i^A) ;
    solve single domain VNE MCF problem;
    foreach flow on substrate link l_mn do
        if l_mn ∈ L_i^S ∪ L_ij^S then pre-allocate resource on link l_mn ;
    end
    send virtual link update notification;
end
```

3.4.4 The MCF-based sub VNE problem

As aforementioned, the mapper (say InP_i) has to solve a reduced VNE problem, with the augmented graph $G_i^A(N_i^A, L_i^A)$ as substrate network, and the sub-request L_i^{subv} as a VN to embed. Recall that only the links need to be mapped.

We formulate it as a multi-commodity flow (MCF) linear programming optimization problem, referred as *LA_MCF*, to determine the flows of intra and peering domain links. The *LA_MCF* problem is formulated as follows.

Variables:

- $F_{l^s}^{l^v}$: A flow variable denoting the total amount of flow on l^s for the virtual link $l^v \in L_i^{subv}$.

Objective:

Minimize:

$$\sum_{l^v \in L_i^{subv}} bw(l^v)\Big(\sum_{l_s \in L_i^S \cup P_i^S} \frac{F_{l^s}^{l^v}}{Res(l^s)+\delta} + \sum_{l_s \in L_i^C} C(l_s) F_{l^s}^{l^v}\Big) \tag{3.1}$$

Subject to:

$$\sum_{n \in N^S | \exists l^s=(m,n)} F_{l^s}^{l^v} - \sum_{n \in N^S | \exists l^s=(n,m)} F_{l^s}^{l^v} = \begin{cases} 1, & m = M(src(l^v)) \\ -1, & m = M(dst(l^v)) \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$
$$, \forall m \in N^s, l^v \in L^V$$

$$\sum_{l^v \in L_i^{subv}} bw(l^v) F_{l^s}^{l^v} \leq Res(l^s), \ \ \forall l_s \in L_i^S \cup P_i^S \tag{3.3}$$

$$0 \leq F_{l^s}^{l^v} \leq 1 \tag{3.4}$$

- The objective function (3.1) minimizes the total embedding cost with balance between the loads of substrate links. $Res(l^s)$ is the residual bandwidth of substrate link l_s. By dividing the residual bandwidth $Res(l^s)$, we prefer to use the intra and peering links which have more residual bandwidth. δ is a small positive value to avoid dividing by zero. For pseudo direct links l_s in L_i^C of G_i^C, we adopt the cost $C(l_s)$ which corresponds to the information provided by VNP.
- Equations (3.2) are the flow conservation constraint. $M(p)$ gives the substrate node on which the virtual node p is mapped.
- Constraints (3.3) are the capacity constraint. The capacity constraint concerns only intra and peering links $L_i^S \cup P_i^S$. The *pseudo* direct links are assumed to have infinite capacities. This simply means that each domain should be able to realize the physical link for every *announced* pseudo direct link.
- Constraints (3.4) allow the path splitting for virtual link mapping.

3.4.5 Reject of virtual request

The resources are definitively allocated only if all the computation on different domains succeed. A COMMIT message is then sent by VNP to InPs so as to validate the resource reservation. Should a mapper report a failure, a DEALLOC message would be sent by VNP, which stops the process (VNE failure) and allows each domain to deallocate pre-allocated resources.

3.5 Reinforcement of our method

As mentioned at the end of Section 3.4.2.1, we choose the link utilization criterion to determine mapping sequence. This choice simplifies the algorithm, but may fail to get the optimal solution. In this section, we propose a reinforcement of our algorithm, which waives the constraint of sequence selection.

Fundamentally, domains are peers. From a domain's point of view, there are actually 2 "domains" : a single domain (itself) and an outside domain (others). Using our downsizing algorithm, a domain tries its best to map its intra and peering virtual links, but how does

the outside domain (others) map the remaining virtual links? We notice that after the first downsizing mapping, the problem is reduced to a multi-domain VNE on the outside domain (others) because the first mapper has mapped its own intra and peering virtual links. Following the downsizing logic, the problem will finally be reduced to a 2 domain VNE, on which a better solution can be easily found. Therefore, we first study the case of only 2 domains, and then we move to VNE in K domains.

3.5.1 Two domain basic method

First, we consider the case of 2 domains. Assuming that the multi-domain VNE problem consists of 2 domains (denoted by InP_1 and InP_2). There are obviously 2 possible mapping sequences.

- First Solution $S_{1->2}$: InP_1 starts up the mapping processus as mapper and then InP_2 solves a single domain VNE.
- Second solution: $S_{2->1}$: InP_2 starts up the mapping processus as mapper and then InP_1 solves a single domain VNE.

These two solutions are sent to VNP, which compares the embedding cost of these solutions. The final solution of the 2 domain basic multi-domain VNE (denoted by $MDVNE(2)$) is the better one among the 2 solutions above:

$$S_{MDVNE(2)} = min\{S_{1->2}, S_{2->1}\}$$

3.5.2 Towards K domain solution

From the 2 domain basic method, K domain multi-domain VNE (denoted by $MDVNE(K)$) can be determined by a recursive algorithm. The detail of $MDVNE(K)$ is shown in Algorithm 2.

The multi-domain is fundamentally divided into 2 elements, a mapper and the others. The former is mapped using our downsizing method (line 3 in Algorithm 2), and the later is reduced to a K-1 domain problem (line 6 in Algorithm 2), until the basic 2 domain problem. The cost of the candidate is the sum of cost of the 2 elements (line 13 in Algorithm 2). The minimum cost candidate will be adopted as the mapping solution.

Algorithm 2: $MDVNE(K)$

```
Input  : VN request
Output : embedding cost
Output : mapping solution
1  begin
2      foreach InP_i do
3          link mapping of InP_i as mapper;
4          get embedding cost C(mapper);
5          if K > 2 then
6              solve MDVNE(K − 1) ;
7              get embedding cost C(others);
8          end
9          else
10             solve MDVNE(2);
11             get embedding cost C(others);
12         end
13         InP_i solution cost C(i) = C(mapper) + C(others) ;
14     end
15     return minimum C(i) and correspond solution;
16 end
```

3.6 Performance Evaluation

We implemented a discrete event simulator to evaluate the performance of our method. The optimization problem is solved by IBM CPLEX library. Since we are basically interested by the link mapping, all the evaluated methods work with the same node decomposition by using the greedy algorithm in [YYRC08].

3.6.1 Evaluation Environment

The simulation framework is shown in Figure 3.15. We use GT-ITM (see Appendix A) to generate substrate networks and virtual networks. The arrival and departure of each VN are added to the events queue. The arrival events call the embedding algorithms, whereas the departure events trigger the resource liberation. After each event, we compute the values of the metrics. In our simulation, we use 4 domains with different configurations. Scenario 1 evaluates the performances in real networks retrieved from SNDlib [OPTW07] (see Appendix A), while scenario 3 generates substrate networks with GT-ITM. Scenario 2 changes the peering link configuration (β) with the same substrate networks. In all the scenarios, the substrate resources are generated in the same manner. The CPU capacity of each node is

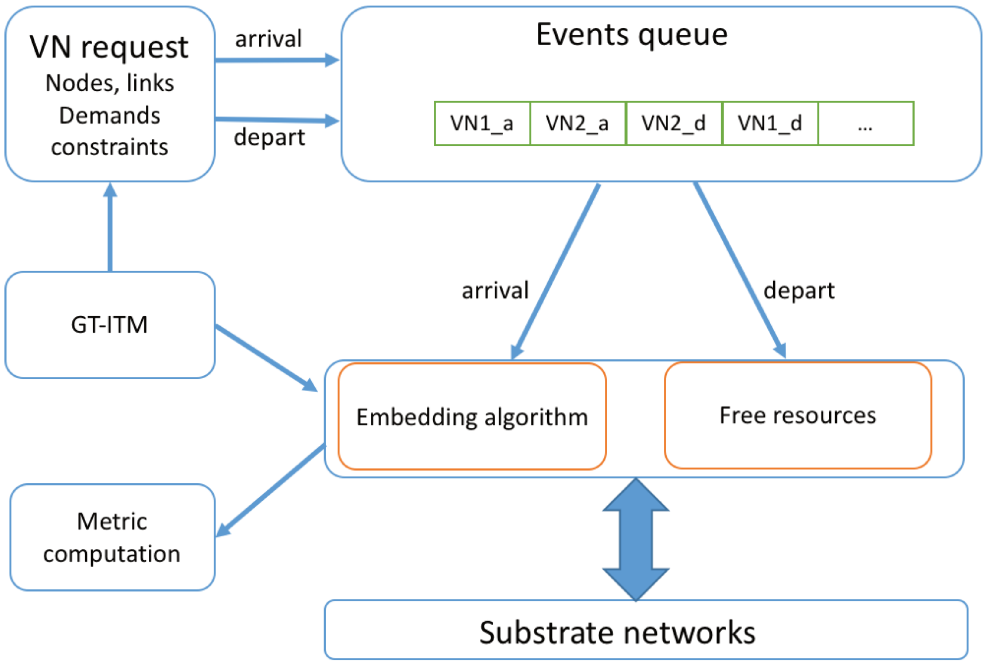


Fig. 3.15 Simulation procedure

chosen in [100,150]. The bandwidth capacity is selected in [100,150] for intra links and in [200,300] for peering links.

The number of virtual nodes of each VN follows a uniform distribution between 3 and 8. The virtual nodes are interconnected with probability 0.5. The CPU demands are uniformly chosen in [0,20]. The effect of different bandwidth demands is simulated in scenario 4.

The VN request arrival process is Poisson with arrival rate λ requests per 100 time units. Scenario 1 and 3 shows the performance of different λ. The life time of each VN follows an exponential distribution with an average of 1000 time units. Each simulation lasts for 30000 time units.

3.6.2 Random peering link model

In our simulation, we adopt a common random graph model proposed by Waxman [Wax88] to generate the peering links between domains. The probability of an edge from u to v is given by:

$$P(u,v) = \alpha e^{-d/(\beta L)} \tag{3.5}$$

where $0 < \alpha, \beta \leq 1$, d is the Euclidean distance from u to v and $L = \sqrt{2} \times scale$ is the maximum distance between any two nodes. An augmentation of α increases the number of

edges in the graph, while an augmentation of β1 increases the ratio of long edges relative to short edges. To simulate the real network situation, we set a big α and a quite small β, so that the boundary nodes are more likely to be connected by peering links.

3.6.3 Compared methods

Our method, called *ciplm* (Coordinated intra and peering link mapping), is compared with the following methods:

(i) *ideal*: The MCF link mapping is done with full information. This means that the multi-domain network is treated as a single domain. This is not feasible in practice.

(ii) *ciplm_up*: It uses the real time attributes (residual bandwidth in particular) to compute the pseudo direct link cost. This method needs to update the residual bandwidth of each link in VNP every time a VN is accommodated. This method is not practical since it requires the exchange of a large amount of information and it claims a less strict information disclosure policy.

(iii) *ciplm_r*: The reinforced method proposed in Section 3.5, which uses the recursive method to select the best mapping sequence. The performance of this method are only measured in Scenario 5.

(iv) *shen* [SXYC14]: This approach computes separately intra and peering links. The latter is determined according to Dijkstra's algorithm.

3.6.4 Metrics

We used the following metrics for comparison:

- *VN request acceptance ratio (AR)*: the ratio of the accepted VN request over the total arrived VN requests. It is formally computed as follows:

$$AR = \frac{acc\_num}{tot\_num}$$

where *acc_num* is the accepted VN number and tot_num is the total VN number.

- *Average link utilization*: the link utilization corresponds to the total allocated link resources over the total substrate resource,

$$LU = \frac{\sum_{l^s \in L^S} Ocp(l^s)}{\sum_{l^s \in L^S} Cap(l^s)}$$

  where $Ocp(l^s)$ is the occupied bandwidth on l^s and $Cap(l^s)$ is the bandwidth capacity of l^s.

- *Total revenue*: The revenue of a VN as the weighted sum of bandwidth and CPU:

$$\mathscr{R} = \theta \sum_{n^v \in N^V} cpu(n^v) + \eta \sum_{l^v \in L^V} bw(l^v)$$

  where θ (resp. η) is the unit revenue for CPU (resp. bandwidth).

- Cost/revenue: The cost is the total mapped substrate resources. The revenue is the total mapped virtual demand. Therefore, this ratio denotes the average mapped substrate resources per unit of virtual demand.

3.6.5 Scenario 1: real substrate networks

In the first scenario, 4 real networks are retrieved from SNDlib: india35 (35 nodes, 80 links), pioro40 (40 nodes, 89 links), germany50 (50 nodes, 88 links), zib54 (54 nodes, 81 links). For each pair of networks, peering links are generated by Waxman with $\alpha = 0.8$ and $\beta = 0.1$. We compare the methods with different arrival rates. The arrival rates determines the load of the network.

Before presenting our results, we would like to make the following qualitative analysis. It is obvious that the *ideal* method provides the best results, thanks to the knowledge of the complete set of information. As its name suggested, it fixes the upper bound. As the *ciplm_up* has richer and updated information, it should provide a better performance than the basic *ciplm* method. It is of course our expectation that the latter yields better results than *shen*, due to the expected impact of joint consideration of both intra and peering links.

The simulation results are shown in figure 3.16. The VN request acceptance ratio, the mapped revenue, the link utilization and the cost revenue are shown in figure 3.16a, 3.16b and figure 3.16c and 3.16d, respectively. We got the following observations:

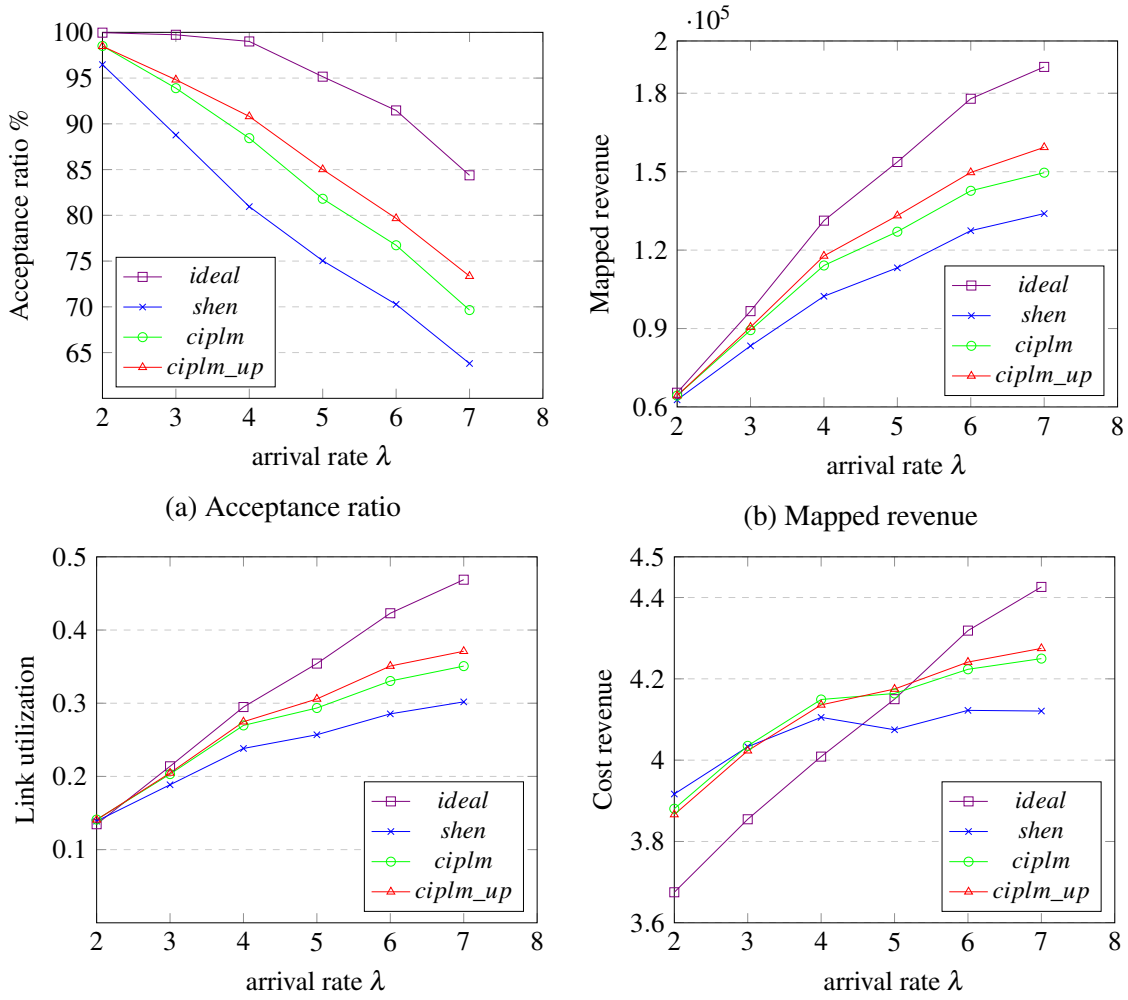


Fig. 3.16 Real substrate networks - arrival rates

- The performance ordering is actually what we expected, i.e. *ideal* is the best, followed by *ciplm_up*, and our basic method *ciplm* outperforms clearly *shen* over all the three metrics.
- The difference between *ciplm* and *shen* is always significant.
- The difference between *ciplm_up* and *ciplm* is not always significant, this is true in particular for the case of link utilisation (figure 3.16c).

3.6.6 Scenario 2: peering links

In this scenario, we study the impact of peering link number on the performances of the compared method. The peering links between each pair of substrate network are determined according to Formula 3.5. Here, $\alpha = 0.9$ and β varies from 0.06 to 0.12. As β increases, the number of peering links increases. For instance, when $\beta = 0.04$, each pair of domains has only one peering link. When $\beta = 0.12$, each domain has 30 to 40 peering links connecting with all the other domains. We use the same real substrate networks as scenario 1. the arrival rate λ is 3.

The simulation results are shown in Figure 3.17. The VN request acceptance ratio, the mapped revenue, the link utilization and the cost revenue are shown in figure 3.17a, 3.17b and figure 3.17c and 3.17d, respectively.

For small β (few peering links), the methods have few choice of peering links, so the advantage of our methods is not obvious. As β increases, the acceptance ratio and mapped revenue shows us that our methods have better performances than *shen*. Our methods take advantage of different peering links to perform an efficient peering link selection and load balancing, whereas *shen* always uses the same peering links which are determined by the shortest path algorithm.

3.6.7 Scenario 3: random substrate networks

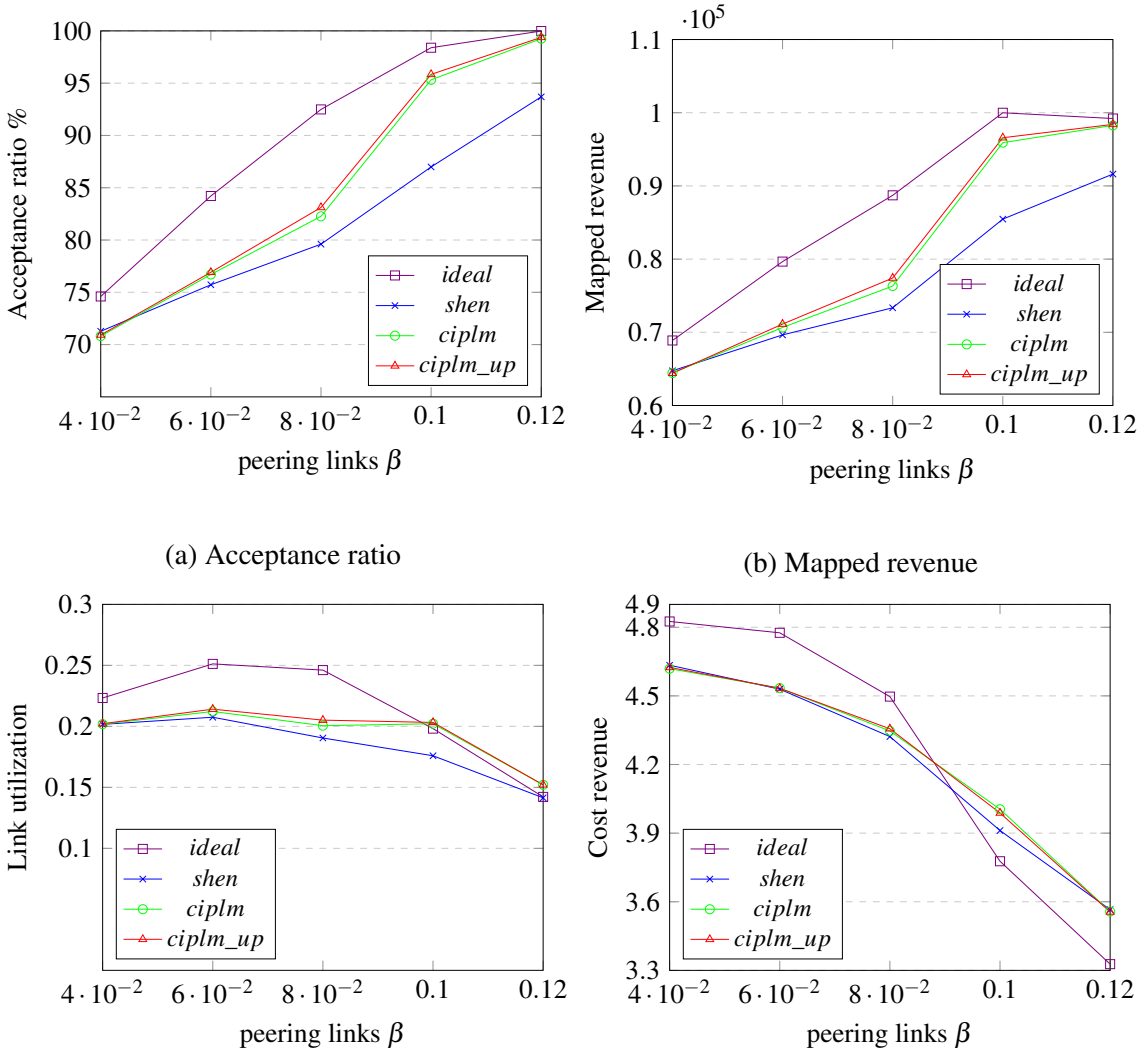
To evaluate the performance of the compared methods on different substrate networks, we generated 4 substrate networks with GT-ITM. Each substrate network is composed of 30 nodes and about 50 links. The curves are the average of 10 runs.

The simulation results are shown in figure 3.18. The VN request acceptance ratio, the mapped revenue, the link utilization and the cost revenue are shown in figure 3.18a, 3.18b and figure 3.18c and 3.18d, respectively.

On random substrate networks, we have the same observation as scenario 1. $ciplm$ has a better performance than *shen*. The difference between $cilpm$ and $cilpm\_up$ is small.

3.6.8 Scenario 4: virtual link demands

In order to illustrate the effect of requested bandwidth, we measure the performance of the compared methods with different virtual link bandwidth demands. The virtual network topology are generated in the same manner as the previous scenarios, but the link bandwidth demand interval varies with [0,20], [10,30], [20,40], [30,50] and [40,60]. α, β and λ are 0.8, 0.12 and 3 respectively.



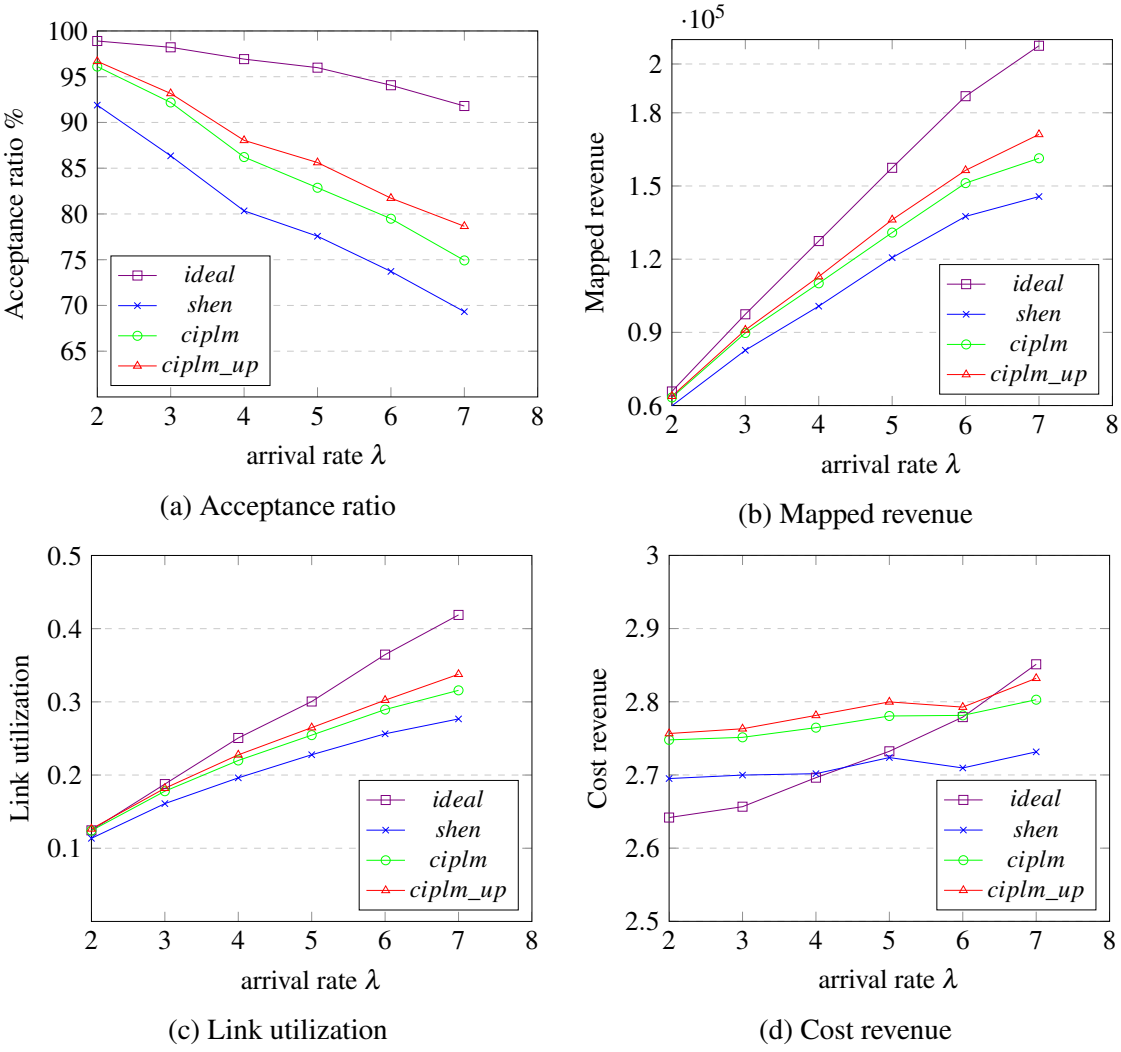
(a) Acceptance ratio

(b) Mapped revenue

(c) Link utilization

(d) Cost revenue

Fig. 3.17 Real substrate networks - peering links



(a) Acceptance ratio

(b) Mapped revenue

(c) Link utilization

(d) Cost revenue

Fig. 3.18 Random substrate networks - arrival rates

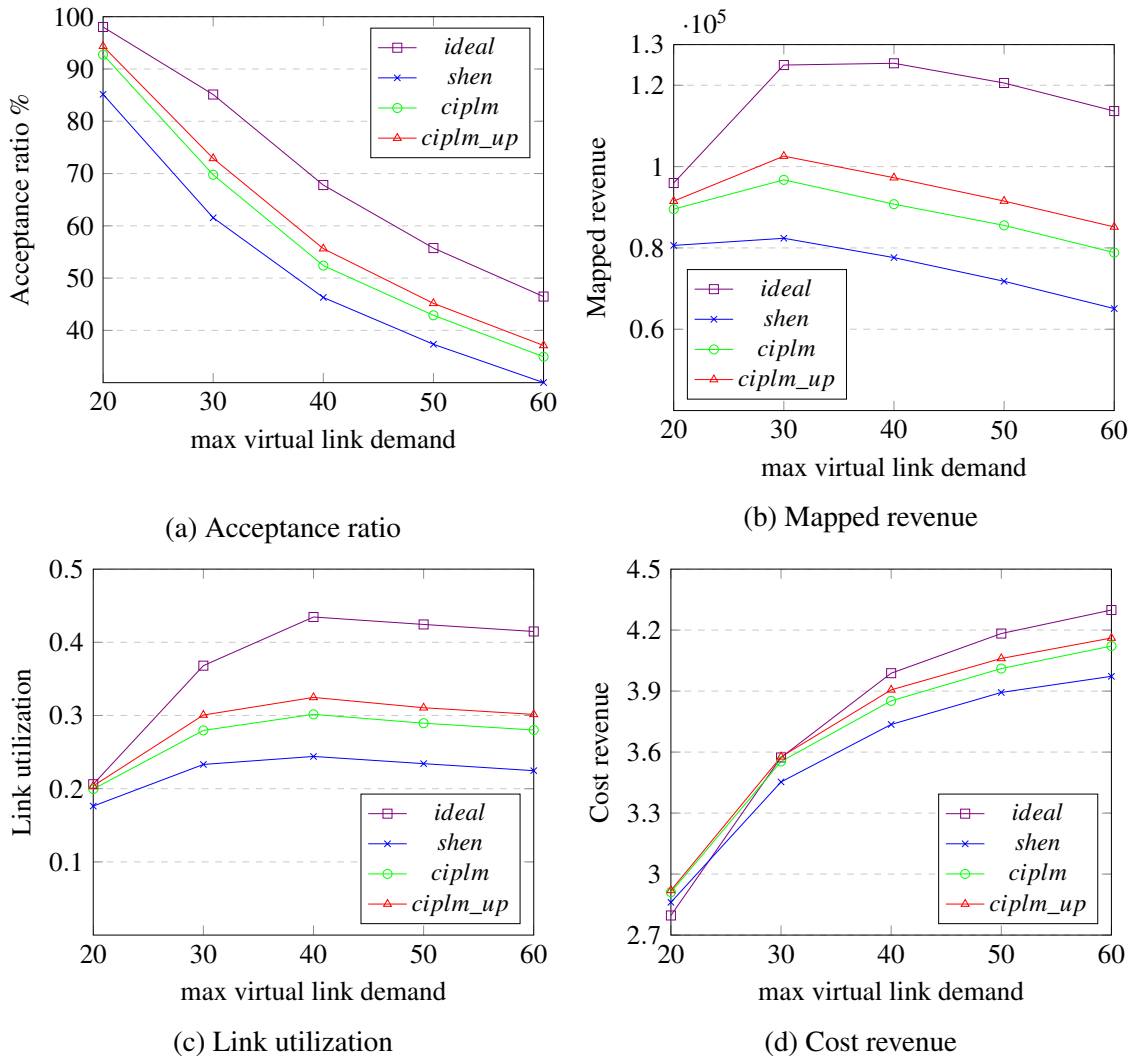


Fig. 3.19 Random substrate networks - bandwidth demand

The simulation results are shown in figure 3.19. The VN request acceptance ratio, the mapped revenue, the link utilization and the cost revenue are shown in figure 3.19a, 3.19b and figure 3.19c and 3.19d, respectively.

The acceptance ratio in Figure 3.19a decreases with the increase of maximum virtual link bandwidth demands, but the difference between the methods remains nearly the same. For different size of bandwidth demands, our methods always have a better resource utilization than *shen*.

3.6.9 Scenario 5: reinforced method

In this scenario, we evaluate the performance of the reinforced method (see Section 3.5). Three domains are generated by GT-ITM. Each of them is composed of 50 nodes and 120 links. The virtual nodes of each VN follow a uniform distribution between 5 and 10. α is 0.9 and β is 0.09.

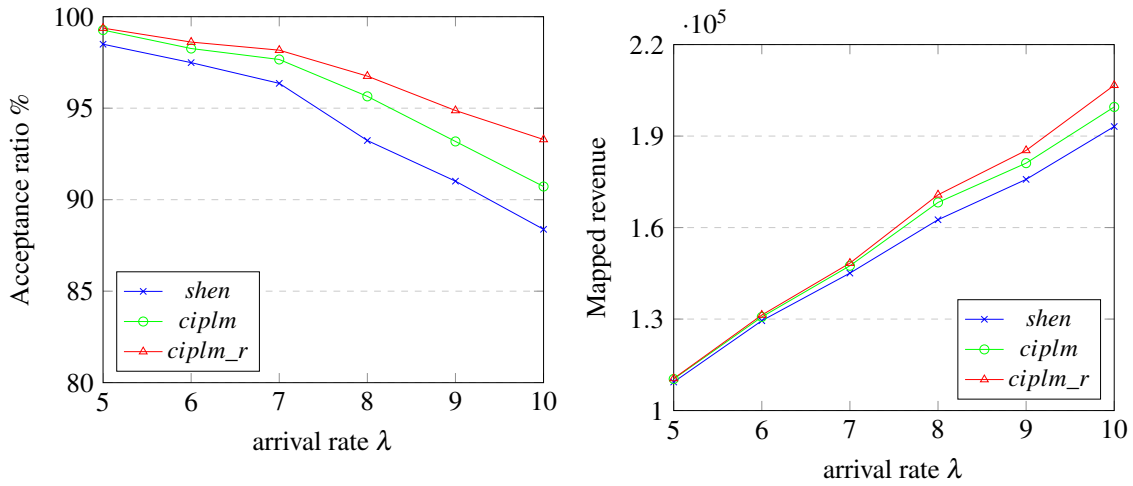
The simulation results are shown in figure 3.20. The VN request acceptance ratio, the mapped revenue, the link utilization and the cost revenue are shown in figure 3.20a, 3.20b and figure 3.20c and 3.20d, respectively. We got the following observations:

- *cilpm_r* is the best, followed by *ciplm*, and then *shen* over all the three metrics.
- The difference between *ciplm* and *shen* is always significant.
- The difference between *ciplm* and *ciplm_r* is often small.

3.6.10 Conclusion of simulation

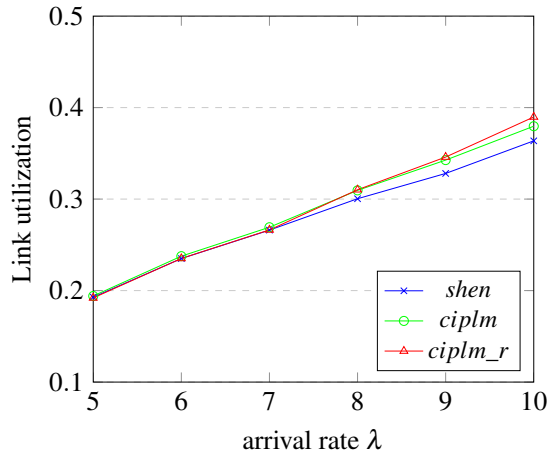
To summarize our 5 scenarios:

- Our approaches are better than that of *shen*. Indeed, mapping jointly intra and peering links enhances the resource utilization by selecting the best intra and peering links. Our proposition improves the performance in particular under heavy loads. In these cases, traffic is splitted and sent to less loaded links, achieving in this way a better utilization of the overall residual bandwidth.
- The comparison between *ciplm* and *ciplm_up* shows that the out-performance of the latter maybe be small. Considering the over-cost of deploying *ciplm_up*, in terms of information exchange and requirement on the information disclosure, we think that the *ciplm_up* does not offer a good trade-off between cost and performance.
- The out-performance of *ciplm_r* is small compared to *ciplm*. *ciplm* ensures a cost-efficient mapping solution for every VN. The peering links are mapped jointly with the right domain, which leads to a better resource allocation. However, the peering links in *cimplm_r* are sometimes not perfectly mapped because link utilization does not always give the best mapping sequence. In this case, exploring all the possible sequences by *ciplm_r* gets better performances.

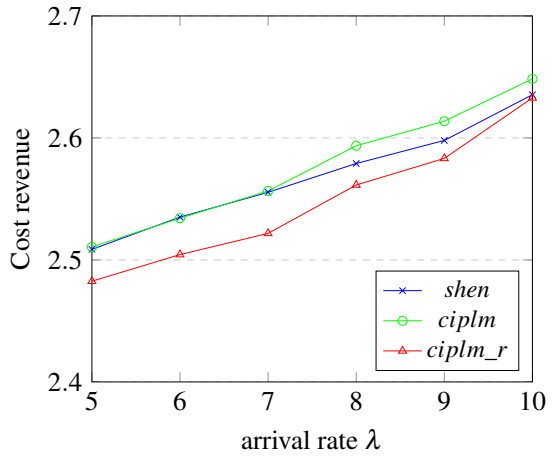


(a) Acceptance ratio

(b) Mapped revenue



(c) Link utilization



(d) Cost revenue

Fig. 3.20 3 Random substrate networks - arrival rates

3.7 Conclusion

Network Virtualization is regarded as an important technique of future network, since it allows the (dynamic) building of a network suited to end-users need, without modifying the underlay infrastructures. Part of them will be built over several infrastructures run by different operators.

The virtual network embedding, which aims at establishing the optimal virtual networks on substrate networks, is a key issue in network virtualization. In multi-domain context, the partial knowledge of routing information makes the multi-domain VNE quite different from the single-domain VNE and this problem remains a challenge. Some multi-domain VNE solutions have been proposed in literature. Most of them focus more on VN decomposition into sub VN requests for each domain, so that the single-domain VNE can be applied subsequently. Few attention has been paid on the mapping of peering (inter-domain) links.

In this chapter, we proposed a novel multi-domain VNE algorithm which aims to jointly optimize the intra and peering link mapping. The VNP layer has the privilege to get a comprehensive vision of all of the domains as well as the peering links. It performs VN decomposition, then coordinates the optimized mapping of both intra and peering links, domain after domain, in an iterative and converging manner. The optimization is achieved by applying the MCF algorithm on an augmented graph related to each domain. Simulation shows that our approach improves the substrate resource utilization compared to existing methods.

Chapter 4

VN Reliability Enhancement

4.1 Introduction

A network fails for various reasons, like bottlenecks, software attacks, natural disasters, etc. A failure of a single physical element (node, link) may result in major disruption involving several VN services. To avoid such scenarios which induce severe penalties (monetary and reputational) for the service provider (SP), this latter should provide reliable VNs capable to deal with failures [CMDTM16].

Various techniques are developed for reliability. These techniques can be grouped in two categories: (1) failure-recovery and (2) failure-avoidance.

Failure-recovery techniques [JK15][HKA13] consist in repairing the affected VNs after the failure occurrence by determining a backup routing. *Protection* mechanisms pre-compute backup paths before the failure occurs, whereas *restoration* performs backup path computation upon failure occurrence. Generally, protection pre-reserves resources for the backup paths to guarantee enough resources upon failure and to ensure service continuity.

Failure-avoidance techniques [SFAM13] try to provide a primary routing which is determined in a way that failures affect as little as possible the network. Such techniques generally determine routes which bypass the network components which are the most vulnerable to failures.

In this chapter, we aim at proposing a novel failure-avoidance oriented approach for VN embedding. Considering that several VNs can share a single SN component, the objective is to set up VNs which are *natively* more reliable, and thus minimize the activation frequency of failure recovery mechanisms. Our solution improves the network reliability without extra configurations and with slight extra resources. It is complementary to, and compatible with, failure recovery mechanisms (restoration/protection).

In this chapter, we propose and describe two failure-avoidance based approaches to improve the virtual network resilience. Both the two approaches are based on a primary VNE. The first one happens when the physical resources are much larger than the logical requests. We model this VNE problem as a classical SMT (Steiner Minimal Tree) problem. This starting point allows us to design a second approach that considers the more realistic scenario of VNE with limited bandwidth links. With the second approach, we improve the resilience by formulating the VNE problem as an Integer Linear Program (ILP) problem for which we propose SMT-based heuristics.

The rest of this chapter is organized as follows. Section 4.2 presents the problem and our direction. Section 4.3 and 4.4 present in more details our solutions. The evaluation results are shown in Section 4.5. Section 4.6 concludes this chapter.

4.2 Avoiding the failures

4.2.1 Position of problem

With the following example depicted in Figure 4.1, we show that different VNE strategies lead to different uses of SN resources and, consequently, different levels of reliability. The virtual network consists of 3 nodes $\{A,B,C\}$ and 2 virtual links $\{A-B,B-C\}$ (Figure 4.1(a)). The virtual links $A-B$ and $B-C$ ask for 5 and 10 units of bandwidth, respectively. The substrate network (Figure 4.1(b)) is composed of 8 nodes and 10 links. Each link in Figure 4.1(b) is labeled with a pair of values corresponding respectively to the residual bandwidth and the link failure probability.

We assume that the node mapping is: $b \rightarrow A$, $f \rightarrow B$ and $c \rightarrow C$. If the main criterion is bandwidth optimization, a likely mapping is given by the dotted line. A non optimal mapping that reduces the failure probability is given in Figure 4.2. The two virtual links in Figure 4.2 cannot cross the substrate link $\{b-e\}$ at the same time because of bandwidth constraint $(10+5>12)$. $B-C$ is embedded to $\{c-d-f\}$, whereas $A-B$ is mapped to $\{b-e-f\}$. The VN failure probability of the former mapping in Figure 4.1(b) is equal to $1-(1-10^{-4})^6$, which is higher than the failure probability of the latter mapping $1-(1-10^{-4})^3(1-10^{-5})$ in Figure 4.2.

This example suggests that the failure of SN components should be specifically taken into account in VNE problem for reliability. This is the focus of this section. Actually, the second mapping is not the optimal one. We show and propose efficient heuristics determining better solutions.

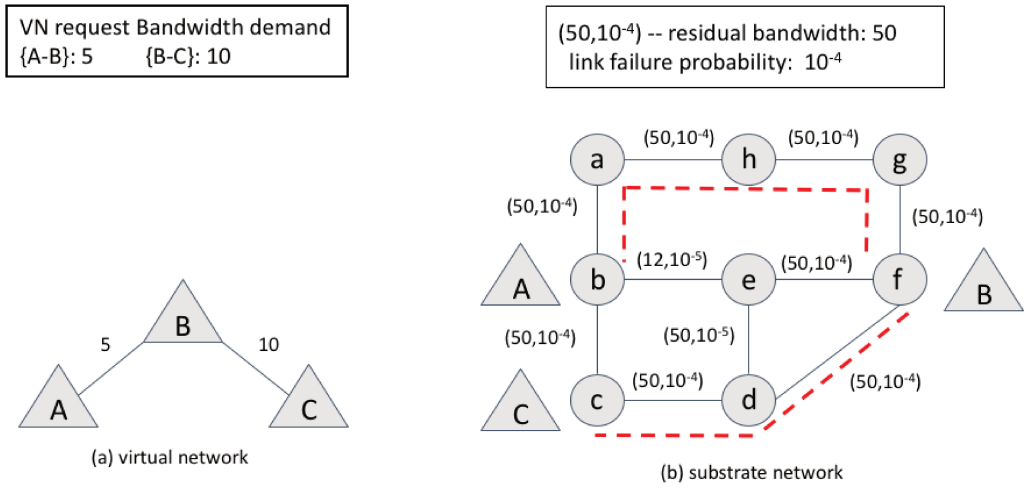


Fig. 4.1 Virtual network embedding without probability consideration

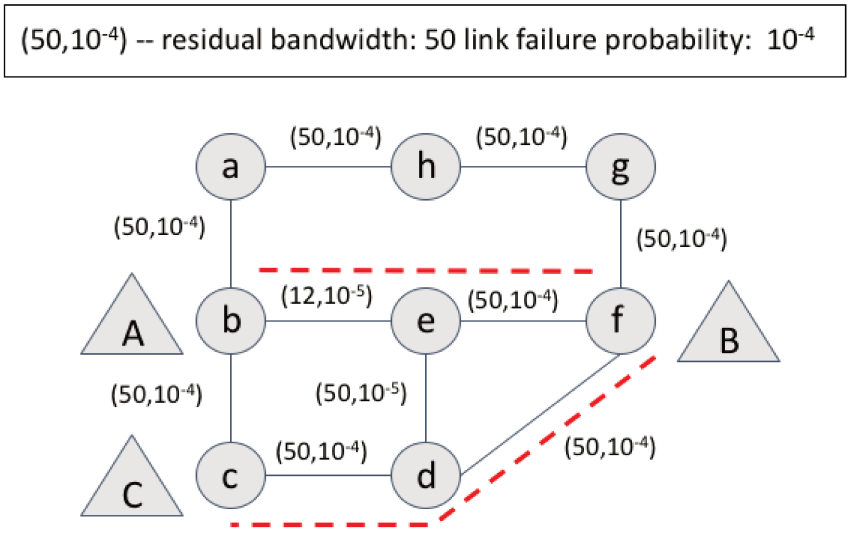


Fig. 4.2 VNE with probability consideration

4.2.2 Our direction

As described in Chapter 2, existing VNEs mainly focus on the optimization of bandwidth. Only few works deal with the failure probability though its importance especially for real time network applications.

Note that failure probability of a network component gives us a good measurement of reliability. In Table 2.2, an existing work which optimizes bandwidth with reliability considerations [GDTV14] is described. In this work, the reliability is treated as a QoS constraint.

The failure probability of network is more critical than bandwidth for various types of network applications. It is computed and approximated by combining various real parameters of network equipment (e.g. age and type of equipment [SFAM13]). In litterature, the optimization of failure probability has been considered to treat some specific types of failures, e.g. SRLG [LML10] and elastic optical network [YCL+15], but there is no work on the optimization of a virtual network failure probability.

Since a VN is composed of several virtual links, the optimization of VN failure probability consists in minimizing the overall failure probability of all the virtual links (i.e. minimizing the probability that at least one virtual link fails). Since failure probability of a virtual link depends on its substrate path, we conclude that the failure probability is minimized for VNs using reliable substrate links (i.e. avoiding the vulnerable substrate links). This is the main idea of the failure avoidance oriented approaches.

In the following sections, we first consider the failure probability optimization without bandwidth constraint. The goal is to find a basic probability-based and failure-avoidance-oriented solution. It it typically useful for the substrate resources much larger than the VN requests. After examining the bandwidth constraint, we extend the basic solution and propose a generic solution for failure probability optimization.

4.3 Solution for infinite bandwidth links

As we are solely interested in VN failure frequency (i.e. recovery process activation frequency), we assume that a VN survives if none of its substrate links is affected by a failure. As our method is compatible with restoration/protection, we assume that failures are eventually recovered, so that the affected VNs are not withdrawn. Besides, we focus on the failure of links. Actually, the joint consideration of both node and link mappings would create an additional difficulty degree that we do not treat here.

Below, we introduce our objective function for VN failure probability minimization. We show how to linearize the objective before solving the problem for the case of infinite bandwidth links.

4.3.1 Objective function

Given a VN request G^V, we define a set of binary variables X_{l^s} denoting whether l^s is used to embed G^V:

$$X_{l^s} = \begin{cases} 1, & l^s \in L_M^S \\ 0, & l^s \notin L_M^S \end{cases}, \quad \forall l^s \in L^S \tag{4.1}$$

The surviving probability $PS(X)$ of a VN is given by:

$$PS(X) = \prod_{l^s \in L^S} (1 - P_{l^s} X_{l^s}) \tag{4.2}$$

Where P_{l^s} is the failure probability of substrate link l^s.

To optimize the reliability, (4.2) should be maximized. Instead of maximizing the non linear function (4.2), it is more easy to look for a new linear function that is optimal for the same solution X^* as (4.2).

To determine such linear function, we apply logarithm function to (4.2):

$$\begin{aligned} log\ PS(X) &= \prod_{l^s \in L^S} (1 - P_{l^s} X_{l^s}) \\ &= \sum_{l^s \in L^S} log(1 - P_{l^s} X_{l^s}) \\ &= \sum_{l^s \in L^S} log(1 - P_{l^s}) X_{l^s} \end{aligned} \tag{4.3}$$

as $log(1 - P_{l^s} X_{l^s})$ is equal to $log(1 - P_{l^s}) X_{l^s}$ for any l^s. Recall that (4.2) and (4.3) are maximized for the same solution X^*. Instead of maximizing (4.3), we chose to minimize $-log\ PS(X)$. The final objective function is described below:

$$\min \sum_{l^s \in L^S} [-log(1 - P_{l^s})] X_{l^s} \tag{4.4}$$

By associating a cost $C_{l^s} = -log(1 - P_{l^s})$ to each link l^s, we deduce that the optimal solution of (4.4) corresponds to the least cost sub-network (in terms of C_{l^s}) which spans all the substrate nodes supporting virtual nodes.

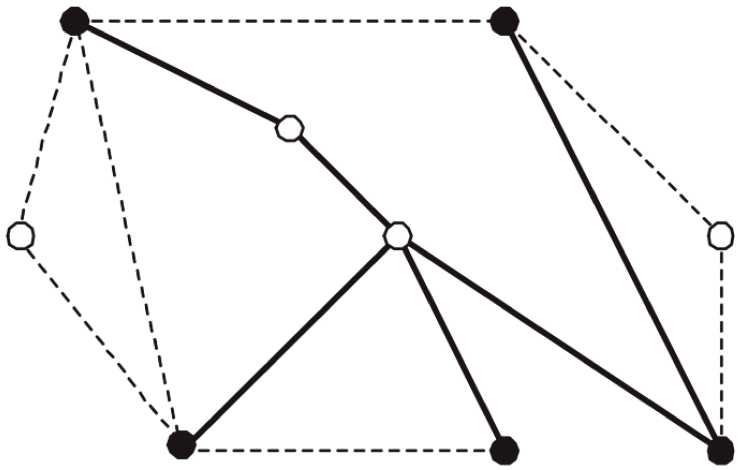


Fig. 4.3 Steiner minimal tree

4.3.2 Steiner minimal tree solution

At VNE link mapping stage, we recall that the substrate nodes (N_M^S) which support virtual nodes via a node mapping function M are known. N_M^S is a subset of N^S. For the case of substrate links with infinite bandwidth capacities, the objective can be reduced to find a connected sub-graph spanning N_M^S with minimal cost C_{l^S}. This is a classical NP-hard problem, called *Steiner minimal tree problem in graph (SMT)*, where N_M^S is the set of Steiner nodes.

4.3.2.1 Steiner minimal tree

Here, we recall the Steiner tree theorem. In graph theory, the Steiner minimal tree problem (SMT) is to find a tree that span a subset of the vertices with minimal total edge weight. The SMT problem is different from the spanning tree problem. A spanning tree spans all vertices of a given graph, a Steiner tree spans a given subset of vertices.

The vertices are divided into two parts: terminals (Steiner nodes) and non-terminal vertices. The terminals are the given vertices which must be included in the solution. The cost of a Steiner tree is defined as the total edge weights. A Steiner tree may contain some non-terminal vertices to ensure the connectivity of the tree. Thus the objective is to find a connected sub-graph spanning all the terminals with minimal total link cost. Since the distances correspond to failure probability in our case, the solution is a tree structure.

An example of Steiner minimal tree is shown in Figure 4.3 where all the link costs are similar. The black points are terminals and the white points are non-terminals. The solid lines depict Sterner minimal tree.

KMB Tree The problem of finding a Steiner minimal tree is NP-complete. Some efficient heuristics have been proposed. Among them we cite KMB [KMB81]. Given a Graph G and a Steiner node set S, KMB's algorithm determines an approximated Steiner tree as follows,

1. construct a complete graph $S'$of Steiner nodes where the links correspond to the shortest paths in G;
2. find the minimal spanning tree on S';
3. replace the links on S' by the corresponding shortest path in G, this graph is called G_s;
4. find the minimal spanning tree T_s of G_s;
5. construct a Steiner tree T_h from T_s by deleting edges in T_s, if necessary, so that all the leaves in T_h are Steiner nodes.

The KMB tree is at worst twice more costly than the Steiner tree.

4.3.2.2 Example

In any tree (Steiner tree, KMB tree, etc.), each couple of nodes are inter-connected by one and only one substrate path. To minimize the failure probability of a VN, we should thus determine a Steiner tree that optimizes the cost $C_{l^s} = -log(1-P_{l^s})$. In our example in Figure 4.1, Steiner tree solution is shown by the dashed lines in Figure 4.4. Concretely, the VN is mapped to $\{c-b, b-e, e-f\}$. The failure probability is $1-(1-10^{-4})^2(1-10^{-5})$.

Note that this mapping is not applicable for the example of Figure 4.1 since the bandwidth constraints are not respected on link $b-e$.

4.4 Solution for limited bandwidth links

In this section, we consider links with limited bandwidth capacities and apply the admission control before link selection. Since failure probability and bandwidth are often orthogonal criteria, we try to optimize the failure probability of VNs without breaking the bandwidth constraint.

The problem of finding a VNE that optimizes the failure probability is also NP-hard for the case of limited bandwidth links. Indeed, for a VN request with a small bandwidth demand (i.e., $\forall$, $l^s \sum_{l^v \in L^V} bw(l^v) \leq C(l^s)$), the optimal solution corresponds to a Steiner tree. Note that generally the solution to the limited bandwidth link version is not a tree because of admission control on links. Below, we formulate the problem as an ILP and then give some efficient heuristics to solve it.

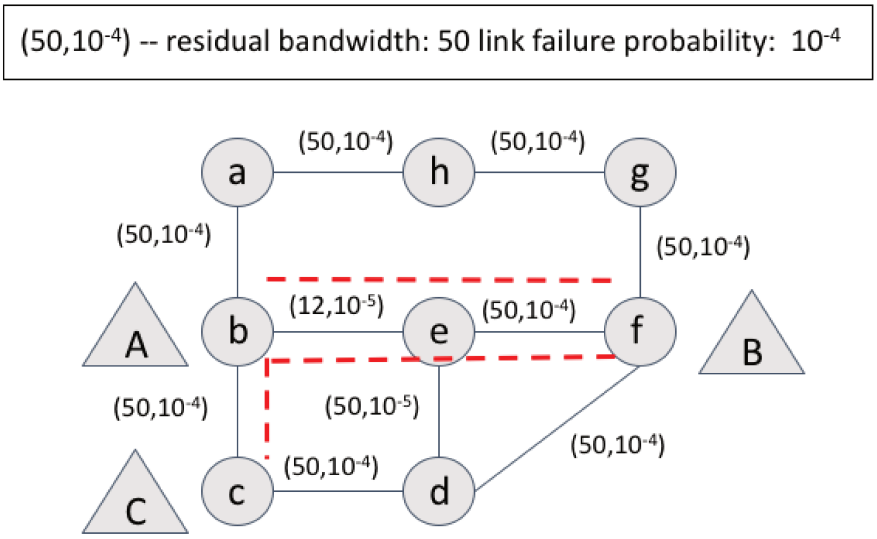


Fig. 4.4 Steiner tree solution

4.4.1 Exact ILP formulation

As the optimal solution is no longer a tree topology, the determination of the optimal substrate links which receive the flows is not sufficient to configure a VN. Indeed, with an optimal substrate network including cycles, several paths can embed a virtual link, resulting in a difficulty of VN setup. On the contrary, determining the flows on substrate links for each virtual link allows not only to configure a VN but also to easily deduce the used substrate links for failure probability computations.

Thus, in our formulation which prohibits the flow splitting, we associate to each substrate link l^s two types of binary $F_{l^v}^{l^s}$ and X_{l^s}. The first variable $F_{l^v}^{l^s}$ is set to 1 only if the virtual link l^v is mapped on a path including the substrate link l^s. The second variable X_{l^s} is set to 1 to indicate that at least one virtual link uses the substrate link l^s for its embedding (see Equation (4.1)). As flow splitting is prohibited in our formulation, we conclude that the amount of flow routed on a substrate link l^s for the virtual link l^v is equal to $bw(l^v) \times F_{l^v}^{l^s}$.

Below, the complete formulation of the VNE which minimizes the failure probability of the VN links is presented. Recall that we assumed independent link failures.

Variables:

- $F_{l^s}^{l^v}$: A binary variable denoting that virtual link l^v is embedded on the substrate link l^s.
- X_{l^s}: See Equation (4.1).

Objective:

Minimize:

$$\sum_{l^s \in L^S} [-log(1-P_{l^s})] X_{l^s} + \varepsilon \sum_{l^s \in L^S} \sum_{l^v \in L^V} F_{l^s}^{l^v} \tag{4.5}$$

Subject to:

$$\sum_{n \in N^S | \exists l^s = (m,n)} F_{l^s}^{l^v} - \sum_{n \in N^S | \exists l^s = (n,m)} F_{l^s}^{l^v} = \begin{cases} 1, & m = M(src(l^v)) \\ -1, & m = M(dst(l^v)) \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$
$$, \forall m \in N^s, l^v \in L^V$$

$$\sum_{l^v \in L^V} bw(l^v) F_{l^s}^{l^v} \leq Res(l^s), \ \ \forall l^s \in L^S \tag{4.7}$$

$$\sum_{l^v \in L^V} F_{l^s}^{l^v} \leq \beta X_{l^s} \tag{4.8}$$

- The objective function (4.5) minimizes the failure probability of virtual network (Formula 4.4). The second term avoids the path repetition. It has an effect of choosing the optimal solution with minimal number of embedding links. ε is a small constant.
- Equations (4.6) correspond to the flow conservation constraint.
- Inequalities (4.7) correspond to the capacity constraint.
- Inequalities (4.8) guarantee that if there exists an amount of flow on l^s, X_{l^s} would be 1. β is a constant greater than the maximum number of VN links.

For the mapping problem described in Figure 4.1, our ILP determines the optimal solution. This solution minimizes the VN failure probability while ensuring the respect of the bandwidth constraints.

4.4.2 Failure avoidance based heuristics for limited bandwidth links

As the VNE minimizing the failure probability is NP-hard, the ILP formulation presented in previous section is not scalable. To accelerate the computations, we propose here efficient heuristics inspired by KMB tree. In our heuristics, each virtual link is mapped to a shortest path minimizing the failure probability. The resulting paths are then combined to form an approached KMB tree and reduce the overall VN failure probability. Because of admission control, the sequence of virtual link mappings impact the final solution. Indeed, the selection of a path traversing a given link can lead to exclusion of such link in the next path computation,

Algorithm 3: baseline heuristic

```
   Input  : virtual network request G^V(N^V, L^V)
   Output : link mapping solution
 1 begin
 2     foreach l^v in L^V do
 3         compute the cost of the probability-based shortest path for l^v without verifying
             the bandwidth constraint;
 4     end
 5     foreach l^v in ascending order of cost do
 6         determine a probability-based shortest path π for l^v. All the substrate links in π
             must verify the constraints of bandwidth
 7         if π = NULL then
 8             free pre-allocated resources;
 9             return no solution;
10         end
11         else
12             foreach l^s in π do
13                 pre-allocate bandwidth resource;
14                 set probability cost C_{l^s} of link l^s to 0;
15                 add π to the solution;
16             end
17         end
18     end
19     return solution;
20 end
```

particularly when its residual bandwidth decreases to zero. Below we propose 2 heuristics improving the reliability of VNs.

4.4.2.1 Baseline heuristic

With our baseline heuristic (see Algorithm 3), we sort the virtual links according to their failure probability costs, without taking into account the bandwidth constraint (line 2-4). Recall that the failure probability based link cost is $C_{l^s} = -log(1 - P_{l^s})$. After that, virtual links are mapped one by one in their ascending order of costs (line 5) to shortest substrate paths. These substrate paths are computed so that they minimize the failure probability-based cost while verifying the bandwidth constraint (line 6). If no substrate path is determined to accommodate the traffic of a virtual link, the heuristic returns no solution (line 7-9).

To decrease the failure probability, we avoid as much as possible the creation of loops by reusing the embedding substrate links. Once a virtual link is mapped to a substrate path, the

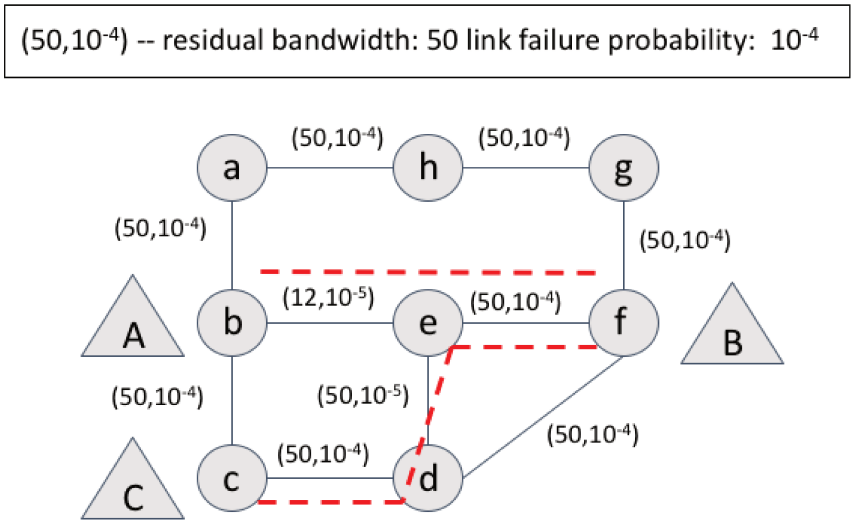


Fig. 4.5 Baseline heuristic solution

link costs on this path are set to zero (line 14) so that the next mappings will privilege the reuse of these links.

After mapping all the virtual links to a set of substrate paths, this later is returned as the final solution.

Note that the complexity of our basic heuristic is $O(|L^V|\ |N^S|log|N^S|)$, where $|N^S|$ is the cardinality of substrate nodes set and $|L^V|$ is the cardinality of virtual link set.

In our example of Figure 4.1, the failure probability based shortest path for $A-B$ and $B-C$ is $\{b-e-f\}$ and $\{c-d-f\}$, respectively. The cost of path $b-e-f$ is smaller than that of $c-d-f$. As a result, the mapping order is $A-B$ and then $B-C$. The baseline heuristic solution is shown in Figure 4.5. After mapping $A-B$ to $\{b-e-f\}$, the cost $C(e-f)$ is set to zero. $B-C$ reuses $e-f$ to determine its substrate path $\{c-d-e-f\}$. Finally, virtual link $B-C$ is mapped to the failure probability based shortest substrate path $c-d-e-f$ which verifies the bandwidth constraints (substrate link $b-e$ cannot be used as it cannot provide enough bandwidth to satisfy the demand). The overall solution is shown by the dashed lines in Figure 4.5. Its failure probability corresponds to the optimum $1-(1-10^{-4})^2(1-10^{-5})^2$ and the total consumed bandwidth is $5\times 2+10\times 3=40$.

4.4.2.2 Reinforced heuristic

The mapping performed by the baseline heuristic is only driven by the minimization of the failure probability which can lead to the reuse and overload of some links. Such heuristic

Algorithm 4: reinforced heuristic

Input : virtual network request $G^V(N^V, L^V)$
Output : link mapping solution

begin
  $UL^V \leftarrow L^V$;
  while $UL^V \neq \emptyset$ **do**
    $cost \leftarrow \infty$;
    foreach l^v *in* L^V **do**
      determine a failure probability-based shortest path π for l^v. All the substrate links in π must verify the constraints of bandwidth
      if $\pi = NULL$ **then**
        free pre-allocated resources;
        return no solution;
      end
      if $C(\pi, l^v) < cost$ **then**
        $cost = C(\pi, l^v)$;
        $\pi^* = \pi$.
        $l^* = l^v$;
      end
    end
    foreach l^s *in* π^* **do**
      pre-allocate bandwidth resource;
      set probability cost C_{l^s} of link l^s to ε;
    end
    delete l^* from UL^V;
    add (l^*, π^*) to solution;
  end
  return solution;
end

can thus result in some substrate paths which consume more additional bandwidth resources compared with the bandwidth optimization oriented VNEs.

To cope with the precedent issue, we slightly modified the baseline heuristic and developed a novel reinforced heuristic (see Algorithm 4). With the reinforced heuristic, we change the cost function to include bandwidth optimization parameters. Besides, the link mapping order is determined step by step. Hereafter, the cost function $C(\pi, l^v)$ of mapping a virtual link l^v on a substrate path π is:

$$C(\pi, l^v) = C(\pi) - \mu \frac{bw(l^v) - \overline{L^V}}{\overline{L^V}}$$

$\overline{L^V}$ is the average of virtual link bandwidth demands and μ is a positive constant controlling the bandwidth optimization level at the cost of VN reliability.

Note that the novel cost is inversely proportional to the bandwidth demand. The fist part of the cost function allows to select the substrate links minimizing the failure probability whereas the second part of the function participates to the determination of the order of virtual link mappings. Larger μ is, higher is the probability of saving the bandwidth resources. For instance, with a high value of μ, the reinforced heuristic first maps the virtual links with the highest bandwidth demands on shortest paths. Then, virtual link mapping requests with small bandwidth demands are satisfied by mapping them on paths that are statistically more longer than the shortest ones. Indeed, the reuse of links with nil costs increases the number of links in paths. As the resource-intensive paths are statically short, the overall VN consumption is expected to decrease compared to the baseline heuristic.

The operation of our reinforce heuristic is detailed in Algorithm 4. $|L^V|$ steps are required to determine the final solution. At each step, a failure probability based shortest path which verifies the constraints of bandwidth is determined for each virtual link that is not mapped yet. The best couple (l^s, π) which minimizes the cost function is then selected and added to the final solution. To decrease the failure probability while balancing the load on links, we modify and set the link costs of the selected path π to a very small value ε (instead of 0) before the starting of the next step.

In the example of Figure 4.1, the solution that is determined by the reinforced heuristic is shown by the dashed lines in Figure 4.6. In the first step, link $B-C$ is mapped to the failure probability based shortest path $f-d-c$. After assigning the cost ε to the substrate links $f-d$ and $d-c$, the second virtual link $A-B$ is mapped to the failure probability based shortest path $b-e-d-f$. This gives us the final solution with a failure probability $1-(1-10^{-4})^2(1-10^{-5})^2$ and the total consumed bandwidth is $10\times 2+5\times 3=35$.

The reinforced heuristic decrease the bandwidth allocation by 5 units without failure probability increase. Generally, the reinforced heuristic allows a trade-off between the failure probability decrease and the bandwidth allocation reduction. The complexity of our reinforced heuristic is $O(|L^V|^2\ |N^S|log|N^S|)$.

4.5 Performance Evaluation

We extend our simulator described in Chapter 2 to evaluate the performance of the methods. In addition of embedding modules, we add new modules to deal with the failure (see Figure 4.7) in our simulator:

- the link failures are generated by a simplified Weibull distribution;

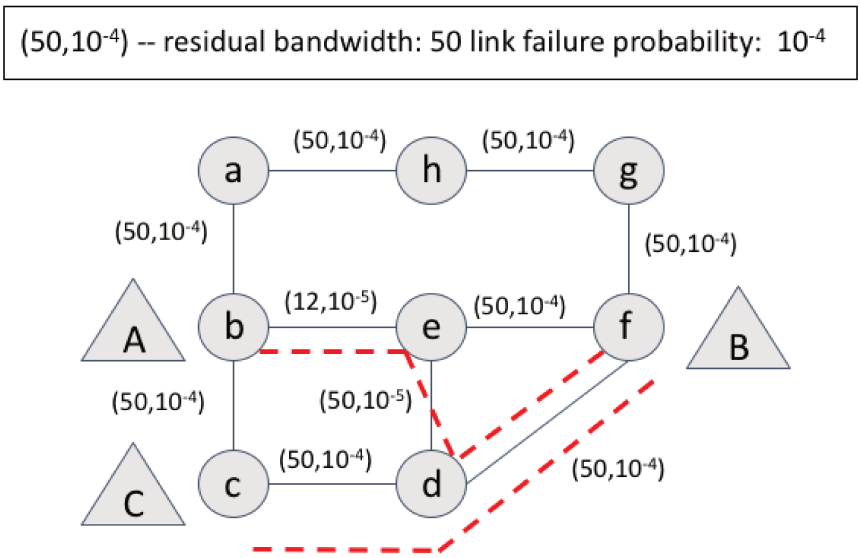


Fig. 4.6 Reinforced heuristic solution

- the failure events are pushed into the events queue along with VN arrival and depart events;
- when a link failure event occurs, the failure based metrics (affected VN, affected revenue, etc.) are computed. The virtual and substrate network are then repaired before treating the next events. The repair operation is out of the scope of our discussion.

Since we are basically interested in link mapping, all the evaluated methods work with the same node mapping described in [YYRC08].

4.5.1 Lifetime model

The Weibull distribution is one of the most widely used lifetime distributions in reliability engineering. It is a versatile distribution that can take on the characteristics of other types of distributions, based on the value of the shape parameter β. The 3-parameters Weibull probability density function (pdf) is given by:

$$pdf(t) = \frac{\beta}{\eta}(\frac{t-\gamma}{\eta})^{\beta-1}e^{-(\frac{t-\gamma}{\eta})^{\beta}}$$

where:

- η is the scale parameters, or characteristic life;

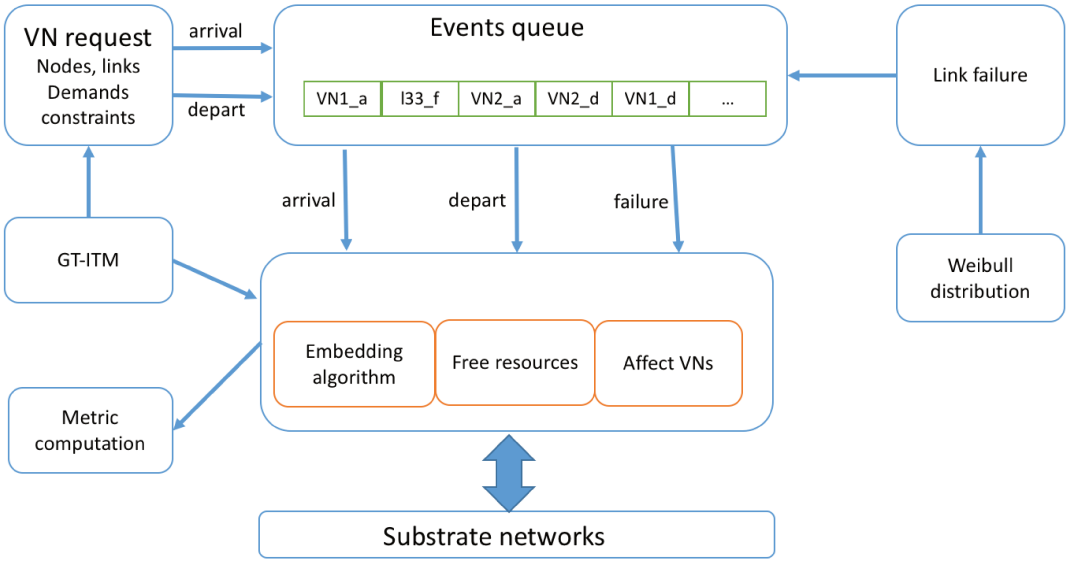


Fig. 4.7 Simulation procedure with link failure

- β is the shape parameter (or slope);
- γ is the location parameter (or failure free life).

A change in the scale parameter η has the same effect on the distribution as a change of the abscissa scale. Increasing the value of η while holding β constant has the effect of stretching out the pdf. Since the area under a pdf curve is a constant value of one, the "peak" of the pdf curve will also decrease with the increase of η.

The Weibull shape parameter β, is also known as the slope. This is because the value of β is equal to the slope of the regressed line in a probability plot.

- $\beta < 1$ indicates that the failure rate decreases over time. This happens if there is significant "infant mortality", or defective items failing early and the failure rate decreasing over time as the defective items are weeded out of the population;
- $\beta = 1$ indicates that the failure rate is constant over time. This might suggest random external events are causing mortality, or failure. The Weibull distribution reduces to an exponential distribution;
- $\beta > 1$ indicates that the failure rate increases with time. This happens if there is an "aging" process, or parts that are more likely to fail as time goes on.

The location parameter γ, as the name implies, locates the distribution along the abscissa. Changing the value of γ has the effect of sliding the distribution and its associated function either to the right ($\gamma > 0$) or to the left ($\gamma < 0$).

The parameters should be decided by the manufacturer a priori from past experience or tests. In our simulation, we set $\gamma = 0$, $\beta = 1$ and $\eta = \frac{1}{P_{l^s}}$ for each substrate link l^s. In this case, the failure rate is constant over time and the failure of l^s is determined by P_{l^s}.

4.5.2 Compared methods

Through the numerical studies, we mainly want to assess the advantage as well as shortcoming of our methods compared to the traditional bandwidth-oriented VNE methods. In our simulation study, we compared the following methods:

(i) *baseline*: our baseline heuristic (Section 4.4.2.1),

(ii) *reinforced*: our reinforced heuristic (Section 4.4.2.2),

(iii) *exact*: the ILP solution provided by CPLEX which acts as the bottom line reference,

(iv) *bw*: a basic shortest path method for virtual network embedding which optimizes the bandwidth. It is our comparison reference.

4.5.3 Metrics

All the metrics presented in Chapter 3 remain available for the comparison. In addition, since we are interested in the survivability of VNs, we introduce some new metrics related to the survivability.

- *VN failure probability*: a VN survives if none of its virtual link fails. We deduce the failure probability P_{G^v} of a VN (G^V) :

$$P_{G^v} = 1 - \prod_{l^s \in L^S_{G^V}} (1 - P_{l^s}), \tag{4.9}$$

  where $L^S_{G^V}$ is the set of substrate links on which G^v is mapped.

- *Average number of affected VNs*: a single substrate link failure affects all of the virtual networks using the failed link. This metric counts the number of affected VNs for each failure event.

- *Average affected revenue*: for each VN affected by a failure event, we compute its revenue $\mathscr{R}(G^v)$ by Formula 3.6.4. The affected revenue (AFR) is the revenue sum of all the affected VNs.

$$AFR = \sum_{G^v \in G^V_f} \mathscr{R}(G^v) \tag{4.10}$$

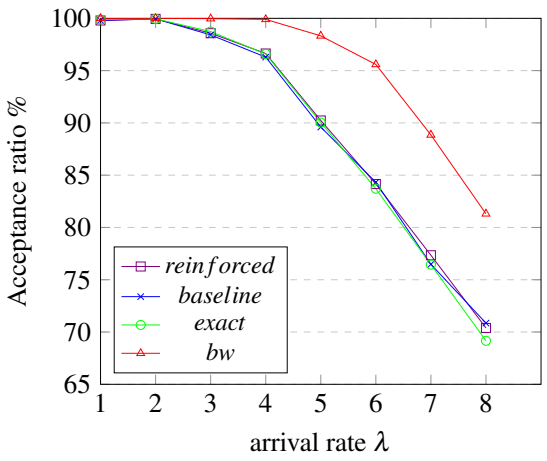


Fig. 4.8 Acceptance ratio

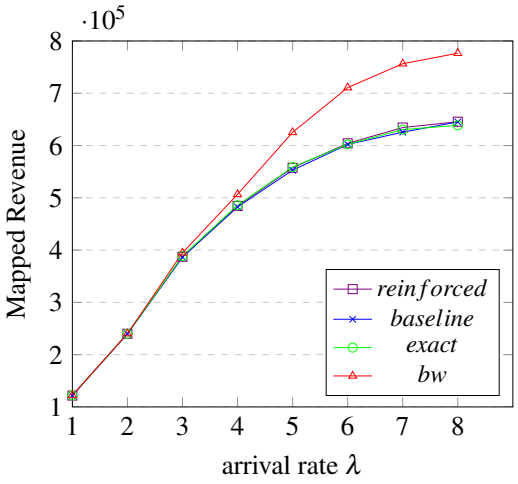


Fig. 4.9 Mapped Revenue

where G_f^V is the set of VNs affected by a failure event f. The average affected revenue is the mean value of AFR for all the failure events.

- *Average ratio of affected VNs*: since the compared methods do not have the same acceptance ratio, it is useful to compute the ratio of affected VNs instead of their numbers. This metric computes the ratio as the number of affected VNs over the numbers of accepted VNs.

In our scenarios, we keep the VN acceptance ratio metric (in Section 3.6.4) to show the primary mapping performances. Apart from acceptance ratio, we mainly focus on the metrics related to the survivability.

4.5.4 Scenario 1: arrival rate

4.5.4.1 Configuration

The substrate network (50 nodes, 120 links) is generated by GT-ITM tool (see Appendix A). The CPU capacity of each node is randomly chosen in [100, 150]. The bandwidth capacity is randomly selected in [100, 150]. The failure probability of substrate links follows a uniform distribution over the interval $[0,\ 2 \times 10^{-5}]$.

The virtual networks are also generated by GT-ITM tool. The number of virtual nodes of each VN follows a uniform distribution between 3 and 8. The virtual nodes are interconnected with probability 0.4. The CPU and bandwidth demands are uniformly chosen in [0, 20].

The VN request arrival process is Poisson with arrival rate $\lambda \in (1 \ldots 8)$ per 100 time units. The life time of each VN follows an exponential distribution with an average of 1000 time units. Each simulation lasts for 100000 time units.

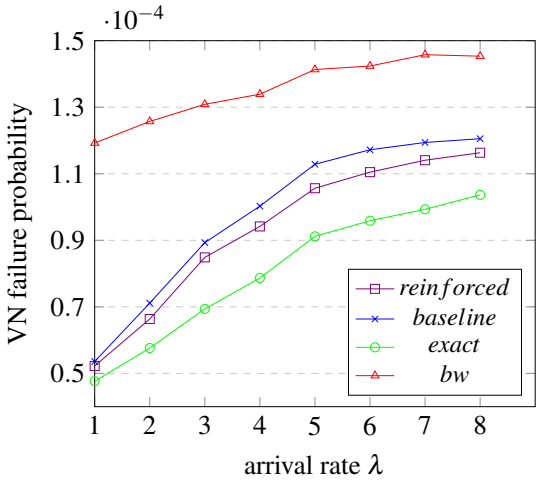


Fig. 4.10 VN failure probability

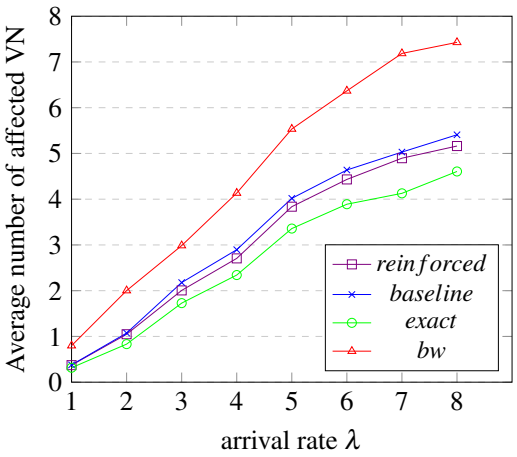


Fig. 4.11 Average number of affected VN

4.5.4.2 Numeric result

In terms of acceptance ratio, our heuristics are surely less efficient than the bandwidth-oriented method. The good news is that the difference is quite small unless in case of heavy load. Figures 4.8 and 4.9 show that, at low network loads ($\lambda < 4$), our methods achieve more than 95% of mapping task compared to *bw*. This ratio falls to 80% for heavy load ($\lambda > 4$).

Our failure-avoidance oriented approaches do work. Actually, Figure 4.10 shows that there is a significant reliability enhancement carried by our method versus the bandwidth-oriented VNE, from about 60% for low network load till 20% for heavy load.

Our heuristics also decrease the failure impact. Actually, as given by Figure 4.11, the number of impacted VNs produced by the bandwidth method in case of failure is always higher than those obtained with our method.

Now, let us take a closer look at the difference between our two methods. As shown by Figures 4.10 and 4.11, *reinforced* outperforms *baseline*. The *reinforced* method takes advantage of the dynamic metric model and it is closer to *exact* (the reference optimal value) than *baseline*. Besides, the bandwidth saved with the reinforced method allows more flexibility for the next path selection. We also want to point out that for the acceptance ratio and revenue metrics (see Figures 4.8 and 4.9), our two heuristics get the performances close to those provided by *exact*.

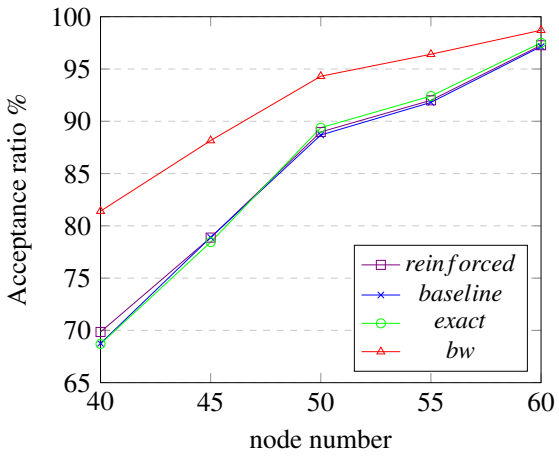


Fig. 4.12 Acceptance ratio

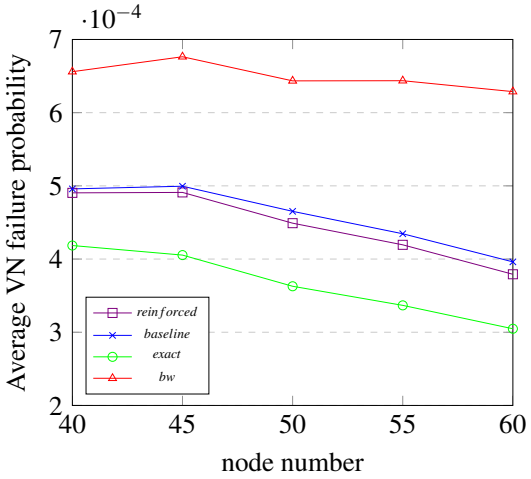


Fig. 4.13 VN failure probability

4.5.5 Scenario 2: variation of the number of substrate nodes

4.5.5.1 Configuration

In this scenario, the method performances are compared with the use of different sizes of substrate networks. The number of substrate nodes varies from 40 to 60. Recall that the difference of substrate nodes number results in different sizes of substrate networks. The CPU capacity of each node is randomly chosen in [100, 150]. The bandwidth capacity is randomly selected in [100, 150]. The failure probability of substrate links follows a uniform distribution over the interval $[0, 10^{-4}]$.

The virtual networks are also generated by GT-ITM tool. The number of virtual nodes of each VN follows a uniform distribution between 3 and 10. The virtual nodes are interconnected with probability 0.5. The CPU and bandwidth demands are uniformly chosen in [0, 20].

The VN request arrival rate λ is 4 per 100 time units. The life time of each VN follows an exponential distribution with an average of 1000 time units. Each simulation lasts for 50000 time units.

4.5.5.2 Numeric result

The numeric results are shown in Figures 4.12, 4.13, 4.14 and 4.15.

In Figure 4.12, we get a similar observation as scenario 1. Our failure avoidance oriented methods consume few additional resources, leading to the acceptance ratio lower than that obtained without failure consideration.

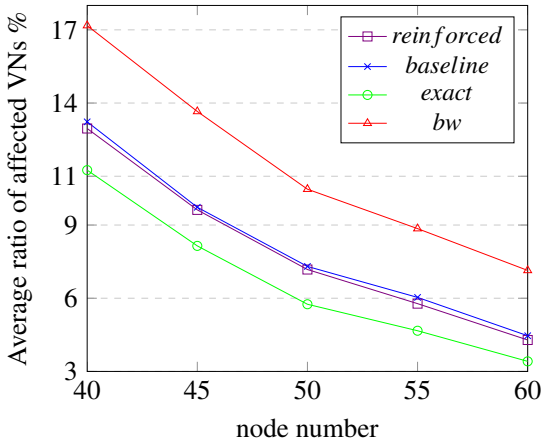


Fig. 4.14 Ratio of affected VNs

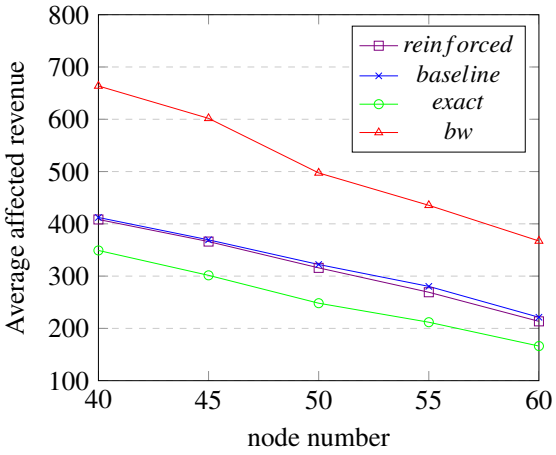


Fig. 4.15 Affected revenue

The VN failure probability in Figure 4.13 is significantly reduced with the use of our methods. This is true for different sizes of substrate network. Since the size of VNs are constant, the big substrate node number cases (55 and 60) simulate light loads. The results show that the difference between the compared methods is almost stable.

For small substrate networks, the average number of VNs crossing each link is smaller than that obtained with large substrate networks. As a result, a link failure affects much more VNs in small substrate networks compared to large substrate networks. This explains the diminution of the failure probability (Figure 4.13), the ratio of affected VNs (Figure 4.14) and the affected revenue (Figure 4.15) with the increase of the number of substrate nodes.

For the same reasons as those mentioned in scenario 1, the reinforced method is slightly better than the baseline heuristic.

4.5.6 Evaluation conclusion

The following conclusions summarize our simulation results:

- Optimization of failure probability can significantly improve the reliability. At the same time, it does not consume too much resources.
- *exact* method determines the lower bound of failure probability optimization. Considering its computation time, *exact* method is not applicable in practice.
- Inspired by Steiner tree problem, *baseline* achieves high level of optimization and can be considered as simple and efficient.

- *reinforced* method saves bandwidth resources and takes advantage of the dynamic metric model. It improves slightly the performances of *baseline* method with little extra time complexity cost.

4.6 Conclusion

Virtual networks are being more and more used as a major component for network architecture. A single component of SN may be used to support a large number of VNs and consequently, its failure may have a large impact on the reliability of these VNs. Failure recovery strategies can be applied to repair the VNs. However, the activation of these strategies induces cost and delay, leading to communication disruptions before the recovery procedure finishes.

In this chapter, we propose and present a novel approach for VNE reliability enhancement. Our approach uses a probabilistic model avoiding as much as possible the vulnerable links at the step of link mapping. In other words, our approach tries to provide *natively* a more reliable virtual network by minimizing its failure probability. Simulation results show that our method does achieve our design goal and improves the reliability compared to the traditional VNEs with bandwidth optimization as a single objective. With our approach, at the price of a slightly lower acceptance ratio, a VN provider can exhibit a better reliability for each single VN instantiation. In case of substrate link failure, the number of affected VNs is also reduced.

Chapter 5

Design of Survivable VN

5.1 Introduction

Network reliability involves a range of issues concerning the design of networks [CMDTM16]. A reliable network should be available as long as possible without service disruption, especially for real-time applications (VOIP, video conference) which are sensitive to network disruption. Reliability also means that network should survive after a link (or node) failure. We recall that networks fail for various reasons, such as software attacks, natural disasters, misconfiguration, link cut, etc. If the survivability against a failure is not guaranteed, a single network element's failure may result in severe penalties for service providers.

In Chapter 4, we have presented failure avoidance oriented approaches, which aim to improve the network reliability by avoiding the vulnerable links. Although the failure avoidance enhances the network reliability without extra configurations, it cannot guarantee service continuity upon any failure. To provide a high level reliability and ensure service continuity, failure recovery is often adopted. It consists in determining backup paths that receive the affected flows when a failure occurs. Restoration and protection are the two main failure recovery techniques. Restoration is a re-active (post-failure) approach, where the backup computation is preformed after the failure occurrence. This technique does not allocate any resource before failure occurrence. On the contrary, protection is a pro-active (pre-failure) approach, which pre-computes backup paths before failures. Generally, protection reserves resources for the backup paths to guarantee enough resources upon failure and ensure service continuity.

The resilience of virtualized networks is provided in 2 stages by protection or restoration [HKA13]. generally, when a VN request arrives, service provider first determines a Virtual Network Embedding (VNE) solution that often optimizes the bandwidth and CPU [FBTB$^+$13]. Then, any restoration or protection method [Kui12] is applied.

Since the existing VNE solutions are designed to optimize primary metrics (bandwidth, latency, etc.), such solutions can lead to the difficulty to find backup paths.

Some survivable VNE solutions optimize jointly the primary and backup resources leading to non optimal routing of the flows most of the time since these latter follow primary paths.

To deal with the previous drawbacks, we propose a survivable service oriented protection-level-aware virtual network embedding. In our solution, the virtual links are mapped on primary substrate paths that are composed of links which are more likely to be protected. Protection capabilities are taken into consideration at the stage of primary mapping. This means that the primary paths are selected in a way that maximizes the protection. In some cases (heavy network loads, disconnected network topology, etc.), VNs could not be entirely protected. In such cases, we reduce the number of unprotected links that we select from the set of less vulnerable links. In this way, probability of VN disruption is minimized.

With the assumption of simple failures, we formulate the VNE problem that maximizes the reliability by selecting the protectable then less vulnerable links. As optimizing jointly primary and backup paths is time-consuming and often not possible, we separate the primary path computation from the backup path determination. In our proposition, primary flows are mapped by privileging substrate links which have feasible backup paths. Then, based on primary mapping, local backup paths are computed by applying the resource sharing between the backup paths.

The rest of this chapter is organized as follows. Section 5.2 describes the network model and the protection method that we adopted. Section 5.3 and 5.4 present respectively our formulation and heuristic propositions to solve the VNE problem which maximizes the reliability. The evaluation results are shown in section 5.5. Section 5.6 concludes this chapter.

5.2 Protection method and network model

5.2.1 Type of failure and protection method

As discussed in Chapter 2, there are various types of failures: nodes, links and SRLG. The link failure is more common than node failure and a single failure occurs ten times than multiple failures. In this chapter, we decided to focus on the most common type of failure, i.e. single link failure.

The link failure can be recovered either by path (global) protection or link (local) protection. The study and analysis of the protection methods (c.f. Chapter 2 show that the path protection methods suffer from long recovery time due to the need to notify the failure to

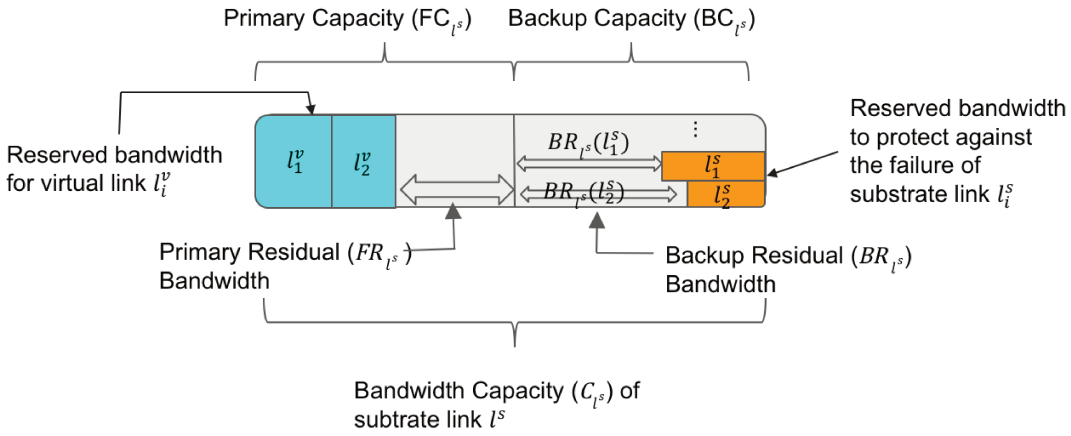


Fig. 5.1 Bandwidth separation

the source node which activates the backup path. Furthermore, dedicated protection often wastes the resources and leads to the reject of a great number of requests. Therefore, for quick failure repair and best resource utilization, the local link protection should be preferred. This type of protection pre-computes a bypass path for each link. Upon failure of a link, the corresponding bypass is activated locally and rapidly to repair the failure. To improve the resource utilization, the resources on the backup bypass paths which protect against the failures of different links[1] are shared.

5.2.2 Primary and backup resource separation

Without loss of generality and in order to simplify the resource allocation, we separate the bandwidth capacity on l^s into two pools: primary bandwidth capacity denoted by FC_{l^s} and backup bandwidth capacity denoted by BC_{l^s} (see Figure 5.1). The primary capacity is occupied by primary links ($l_1^v, l_2^v, etc.$) whereas backup capacity is dedicated for backup paths protecting against failure of substrate links ($l_1^s, l_2^s, etc.$). The primary residual bandwidth of a substrate link l^s is denoted by FR_{l^s}. Since the backup resource is shared, a substrate link l^s could protect against the failure of several substrate link risks l_i^s (l_1^s and l_2^s in Figure 5.1). For each l_i^s, the backup residual resource is defined as $BR_{l^s}(l_i^s)$.

[1] The link failure is also called *risk*.

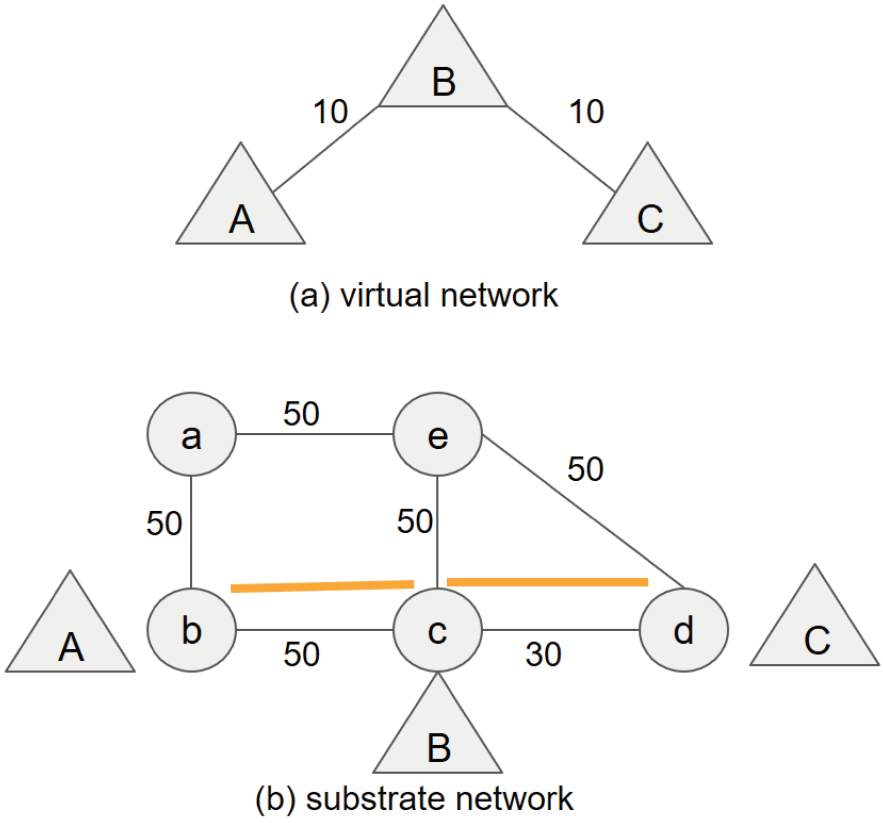


Fig. 5.2 Primary Mapping

5.2.3 VNE with protection

Since a VN is composed of several virtual links, VN protection is more difficult to achieve than the protection of a single path. VNE with link protection often includes the following steps:

1. determining the primary mapping (nodes and links),
2. computing the backup paths for all the substrate links on the primary paths with bandwidth respect,
3. reserving and updating the resources on backup links.

We briefly present the first and second steps. The third step is quite simple and thus not presented here. We also show some issues relating to VNE with link protection.

5.2.3.1 Primary mapping

VNE for primary flows can be performed in one or two stages. Recall that we are focusing on the link mapping and protection issue. We denoted $M()$ as the node mapping function, so $M(A)$ implies the substrate node on which the virtual node A is embedded. All of the embedded substrate nodes form the set N_M^S. Link mapping stage embeds the virtual links to a subset of substrate links denoted by L_M^S.

A primary mapping example is shown in Figure 5.2. The virtual network consists of 3 nodes $\{A,B,C\}$ and 2 virtual links $\{A-B,B-C\}$ (Figure 5.2(a)). Both the virtual links demand 10 units of bandwidth. The substrate network (Figure 5.2(b)) is composed of 5 nodes and 6 links. Each substrate link l^s is labelled with the corresponding primary residual bandwidth FR_{l^s} (e.g. $FR_{a-b} = 50$). We assume that the node mapping is: $M(A) = b$, $M(B) = c$ and $M(C) = d$. If the main criterion is primary bandwidth optimization, a primary mapping is given by the solid lines, i.e. $\{A-B\}$ and $\{B-C\}$ are mapped to $\{b-c\}$ and $\{c-d\}$ respectively.

5.2.3.2 Backup path computation

For each substrate link l^s used by the primary mapping, a backup path is computed to deal with its failure. The resources of backup paths which protect against different failures on the same substrate links bl^s are shared. Constrained shortest path algorithm is run on the network that includes only the link verifying the capacity constraints to determine the backup paths.

In the example of Figure 5.2, we assume that the residual backup bandwidth associated to the failure of $c-d$ is 10 on all the substrate links (e.g. $BR_{e-c}(c-d) = 10$) except for

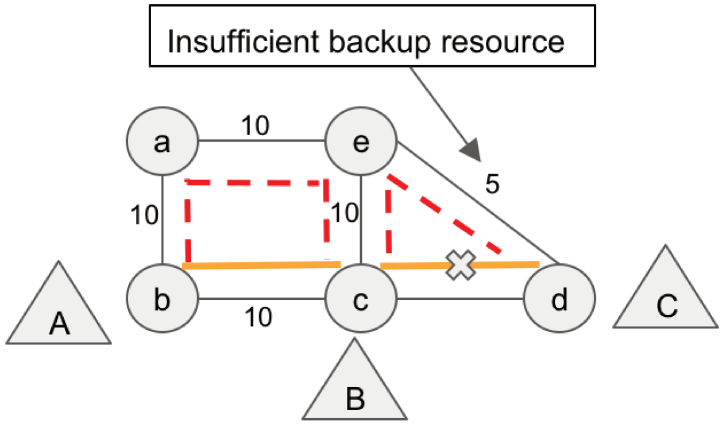


Fig. 5.3 Local share protection

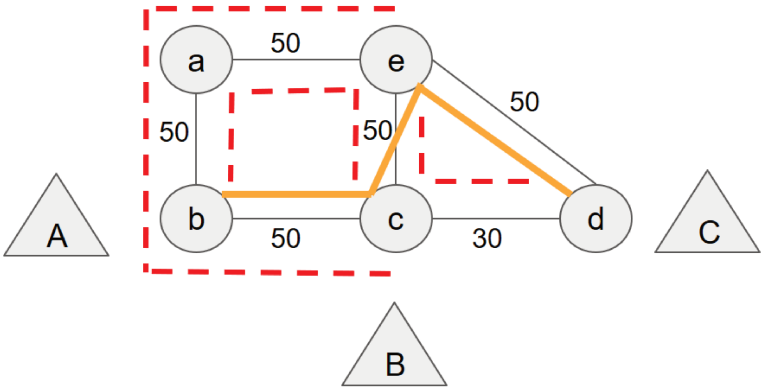


Fig. 5.4 Primary mapping with backup consideration and local share protection

$e-d$ where it is equal to 5 (i.e. $BR_{e-d}(c-d)=5$). In Figure 5.3, the primary link $b-c$ is protected by the backup path $b-a-e-c$ and the primary link $c-d$ is protected by the backup path $c-e-d$. The backup resources on $e-c$ are shared between the previous backup paths which protect against failures of different links $b-c$ and $c-d$.

Although the solution shown in Figure 5.2 optimizes the primary resources, it is insufficient to allow the VN protection. Indeed, this solution includes the primary substrate link $c-d$ that is unprotectable because of the lack of backup resource on the link $e-d$. As a result, link $c-d$ should not be used for the primary mapping. To improve the reliability, the primary path $c-e-d$ should be selected (see Figure 5.4) for the mapping of virtual link $B-C$ to ensure its full protection. In Figure 5.4, the overall primary mapping and its local share protection solution are depicted. Note that the labels on substrate links are primary residual bandwidth (as in Figure 5.2, $FR_{c-d}=30$). links $b-c$, $c-e$ and $e-d$ are protected by backup paths $b-a-e-c$, $c-b-a-e$ and $e-c-d$, respectively.

5.2.4 Position of problem

The resource sharing is useful but not sufficient to optimize the resource allocation. Moreover, the protection can fail to determine a backup path though the backup resources are reduced. As shown in Figures 5.3 and 5.4, the primary path selection has incidence on the protection capabilities: the primary path $c-d$ cannot be protected whereas the path $c-e-d$ is fully protected. Instead of rejecting a VN because of insufficient backup resources, an approach could decide to accept the VN with some unprotected links. In this case, the number of unprotected links should be minimized. In addition, the unprotected links should be as much as possible less vulnerable.

Below, we give the issues that we should solve to improve the reliability:

- the independent computation of primary and backup mappings can lead to the difficulty of backup path establishment;
- the lack of backup resource can result in the rejection of the whole VN though the number of unprotected links is low;
- the unprotected links need to be well selected to reduce the failure risks.

In the following sections, we try to solve these issues by minimizing the VN failure probability. The idea consists in combining the protection with the failure avoidance oriented approach.

5.3 Protection-level-aware VNE Formulation

In this section, we formulate the problem of VNE that maximizes the reliability by combining the protection with the failure avoidance technique. In our formulation, the primary bandwidth resources are optimized under the constraint of maximizing the reliability. Thus, in addition of primary flow and capacity constraints, we add backup path constraints. Recall that with the separation of the bandwidth capacity in primary and backup pools, it is sufficient to ensure that the residual backup bandwidth is lower than the backup capacity of each protected link.

5.3.1 Objective

Classical VNE formulation optimizes primary bandwidth resources. Given a VN request $G(N^V, L^V)$, a typical objective function corresponds to:

$$\min \quad \sum_{l^s \in L^S} \frac{1}{FR_{l^s}} \sum_{l^v \in L^V} F_{l^s}^{l^v} \tag{5.1}$$

where $F_{l^s}^{l^v}$ is a binary variable denoting that virtual link l^v is routed on a path including l^s.

We define a set of binary variables Y_{l^s} indicating if VN is protected against the failure of link l^s. Note that a VN is always protected against the failure of links which are not used for the primary mapping.

$$Y_{l^s} = \begin{cases} 1, & \text{VN is protected against the failure of } l^s \\ 0, & \text{VN is not protected against the failure of } l^s \end{cases} \tag{5.2}$$

The survivability probability $PS(Y)$ of a VN against single failure is given by:

$$PS(Y) = \prod_{l^s \in L^S} (1 - P_{l^s}(1 - Y_{l^s})) \tag{5.3}$$

Formula (5.3) computes the survivability of a VN as the survivability probability product of its primary links. This probability is equal to $(1 - P_{l^s})$ if l^s is not protected, otherwise it is equal to 1. Thus, to optimize the reliability, Formula (5.3) should be maximized. Instead of maximizing the non linear function (5.3), we look for a linear function that is optimal for the same solution as (5.3).

To determine such linear function, we apply logarithm function to (5.3):

$$\begin{aligned} log\ PS(Y) &= log \prod_{l^s \in L^S} (1 - P_{l^s}(1 - Y_{l^s})) \\ &= \sum_{l^s \in L^S} log(1 - P_{l^s}(1 - Y_{l^s})) \\ &= \sum_{l^s \in L^S} log(1 - P_{l^s})(1 - Y_{l^s}) \end{aligned} \tag{5.4}$$

(5.3) and (5.4) are maximized for the same solution. Instead of maximizing (5.4), we chose to minimize $-log\ PS(Y)$. The final objective function that maximizes the VN survivability is described below:

$$\min \sum_{l^s \in L^S} [-log(1 - P_{l^s})](1 - Y_{l^s}) \tag{5.5}$$

To optimize the primary resources under the constraints of maximizing the reliability, Formula (5.1) and (5.5) should be combined (see the next Section).

5.3.2 Formulation

By combining the survivability probability and primary bandwidth resources, we formulate the survivable VNE problem that minimizes the primary resources under condition of maximizing the reliability without path splitting as follows:

Variables:

- $F_{l^s}^{l^v}$: Primary flow (see above).
- $B_{bl^s}^{l^v}(l^s)$: A binary variable denoting whether backup link bl^s protects the substrate link failure risk l^s for the flow of l^v. In other words, $B_{bl^s}^{l^v}(l^s)$ is set to 1 if the failure of l^s results in the rerouting of the primary flow of l^v on a backup path crossing link bl^s.
- Y_{l^s} : Protection indicator (see Formula (5.2)).

Objective:

Minimize:

$$\sum_{l^s \in L^S} [-log(1 - P_{l^s})](1 - Y_{l^s}) + \varepsilon \sum_{l^s \in L^S} \frac{1}{FR_{l^s}} \sum_{l^v \in L^V} F_{l^s}^{l^v} \tag{5.6}$$

Subject to:

Primary flow constraints:

$$\sum_{n\in N^S|\exists l^s=(m,n)} F_{l^s}^{l^v} - \sum_{n\in N^S|\exists l^s=(n,m)} F_{l^s}^{l^v} = \begin{cases} 1, & m = M(src(l^v)) \\ -1, & m = M(dst(l^v)) \\ 0, & \text{otherwise} \end{cases} \quad ,\forall m \in N^s, l^v \in L^V \tag{5.7}$$

Primary bandwidth capacity constraints:

$$\sum_{l^v\in L^V} bw(l^v) F_{l^s}^{l^v} \leq FR_{l^s}, \qquad \forall l^s \in L^S \tag{5.8}$$

Backup flow constraints:

$$\sum_{n\in N^S|\exists bl^s=(m,n), bl^s\neq l^s} B_{bl^s}^{l^v}(l^s) - \sum_{n\in N^S|\exists bl^s=(n,m), bl^s\neq l^s} B_{bl^s}^{l^v}(l^s) = \begin{cases} F_{l^s}^{l^v} Y_{l^s}, & m = src(l^s) \\ -F_{l^s}^{l^v} Y_{l^s}, & m = dst(l^s) \\ 0, & \text{otherwise} \end{cases} \quad ,\forall m \in N^s, l^v \in L^V, l^s \in L^S \tag{5.9}$$

Backup bandwidth capacity constraints:

$$\sum_{l^v\in L^V} bw(l^v) B_{bl^s}^{l^v}(l^s) \leq BR_{bl^s}(l^s), \ \ \forall bl^s \in L^S, l^s \in L^S \tag{5.10}$$

Objective (5.6) combines and favors the survivability failure probability optimization (Equation (5.5)) to the primary resource allocation (Equation (5.1)). The rationale of this combined optimization is the following. By multiplying the amount of primary resources with a small constant ε, we favor the minimization of the first part in objective (5.6) which corresponds to the survivability probability. In other words, objective (5.6) privileges the selection of the best primary links which are protectable: if all the links are protected ($Y_{l^s} = 1, \forall l^s \in L^S$), bandwidth resource is optimized for primary links; When some primary links cannot be protected due to the lack of resources ($Y_{l^s} \neq 1, \forall l^s \in L^S$), the survivability probability is maximized.

Constraints (5.7) and (5.8) are primary flow and capacity constraints. $src(l^v)$ and $dst(l^v)$ retrieve respectively the source and destination nodes of l^v. Recall that $M()$ is the node mapping function.

Constraints (5.9) are backup flow constraints. These constraints can be easily linearized. Constraints (5.10) ensure that the cumulated bandwidth of the backup paths traversing bl^s and protecting against the failure risk l^s should not exceed the backup capacity of link bl^s.

Note that the ILP problem presented here does not optimize the resources of backup paths. Slight modifications to the objective function will permit such optimization. In addition, it may involve a very large number of binary variables. For instance, assume that the substrate network consists of 100 links and a VN request consists of 10 virtual links. The number of variables $B^{l^v}_{bl^s}(l^s)$ is about 10^5. Computing the primary paths while pre-determining backup paths at the same time results in explosion of the number of variables.

In the next section, we propose a heuristic to solve the precedent problem in polynomial time.

5.4 Heuristic

Here, we propose an efficient heuristic for survivable VNE problem without path splitting. In our proposition, the primary paths are selected as the shortest ones according to the survivability probability. When several paths optimizing the survivability probability exist, we choose the path that minimizes the primary resources.

Next, we describe our propositions which favor the use of protectable links for primary mapping. For this purpose, a backup feasibility algorithm is performed for each substrate link to select the protectable links.

5.4.1 Principle

Our objective (5.6) consists in determining the best mapping that maximizes the survivability probability. The best mapping is the one that minimizes the primary resource allocations among the mappings which optimize the survivability probability.

To achieve our objective, we define a new link cost allowing the maximization of the survivability probability which reducing the primary resource allocations with the use of shortest path algorithm. Our link cost is determined after separating the potential primary links in two subsets: protectable links which can be protected by backup paths, and unprotectable links which cannot be protected due to topology characteristics or to the lack of resources. Recall that a protected link is 100% reliable since we assume single failures. Below, the formal definition of the link cost is:

$$cost(l^v, l^s) = \frac{\varepsilon}{FR_{l^s}} + \begin{cases} 0, & \text{if a backup path protecting } l^s \textit{ exists} \\ -log(1 - P_{l^s}), & \text{otherwise} \end{cases} \tag{5.11}$$

For the protectable links, the primary costs depend only on resource allocations ($\frac{\varepsilon}{FR_{l^s}}$). For the unprotectable links, the primary costs depend only on failure probability ($-log(1 - P_{l^s})$). In this way, the mapping of primary virtual links on shortest paths will result in the selection of the paths which maximize the survivability probability and reduce the primary resource allocation.

5.4.2 Simple on-line backup verification

In our first proposition, we compute on-line the link costs described in the previous section. For each potential substrate link of a virtual link, we check for the existence of a backup path that protects it. Accordingly, we determine the primary costs (see Equation 5.11) and deduce the shortest primary paths. Then, backup paths protecting the links of the determined primary paths are established and configured.

The details of our proposition are shown in Algorithm 5. The link costs are computed in lines 4-9. The cost part related to the failure probability is added when backup feasibility fails to determine a backup path (unprotectable links). Primary link mapping is then determined with the use of the shortest path algorithm that optimizes the link cost (line 10). For each primary link, a local backup path is determined in line 18. We note that the shortest path algorithm can be used for backup path feasibility verification and determination.

The resources are allocated for backup links in line 20. The backup residual bandwidth ($BR_{bl^s}(l^s)$) is updated for link l^s so that the backup feasibility verification will take it into account in next mappings.

In Algorithm 5, we deliberately omit to specify the backup path search procedure (line 5 and 18). By assuming a complexity of $O(T)$ for the backup path search procedure, we deduce the complexity of Algorithm 5 which correspond$O(|L^V|\ |L^S|\ T)$. This means that the minimal complexity corresponds to $O(|L^V|\ |L^S|^2)$ since the quickest backup search procedure has complexity $O(|L^S| + |N^S| = O(|L^S|)$. When the backup paths correspond to the shortest ones, the complexity is $O(|L^V|\ |L^S|^2\ log|N^S|)$.

Algorithm 5: On-line backup verification based link mapping

Input : virtual network request $G^V(N^V, L^V)$
Output : link mapping and backup solution

begin
  foreach $l^v \in L^V$ **do**
    foreach $l^s \in L^S$ **do**
      Set $cost(l^v, l^s) = \frac{\varepsilon}{FR_{l^s}}$;
      Determine backup path π' for l^s, such that: $\forall bl^s \in \pi' : bw(l^v) \leq BR_{bl^s}(l^s)$;
      if $\pi' = NULL$ **then**
        $cost(l^v, l^s) = cost(l^v, l^s) + [-log(1 - P_{l^s})]$
      end
    end
    Determine shortest $cost(l^v, l^s)$ based primary path π;
    if $\pi = null$ **then**
      Free pre-allocated resource;
      return no solution;
    end
    else
      Pre-allocate primary resource;
      foreach $l^s \in \pi$ **do**
        Determine backup path π' for l^s, such that:
        $\forall bl^s \in \pi' : bw(l^v) \leq BR_{bl^s}(l^s)$;
        if π' *exists* **then**
          Pre-allocate backup resource on π';
        end
      end
    end
  end
  return primary link mapping and backup solution;
end

5.4.3 Backup path pre-verification

The shortcoming of the on-line backup feasibility verification is that it requires for each VN to compute the backup paths protecting all the potential primary links. This on-line computation is time consuming.

Below, we propose a second approach that pre-determines the backup paths, in advance at network initialization (i.e. before VN arrivals). It is based on maximum flow algorithm.

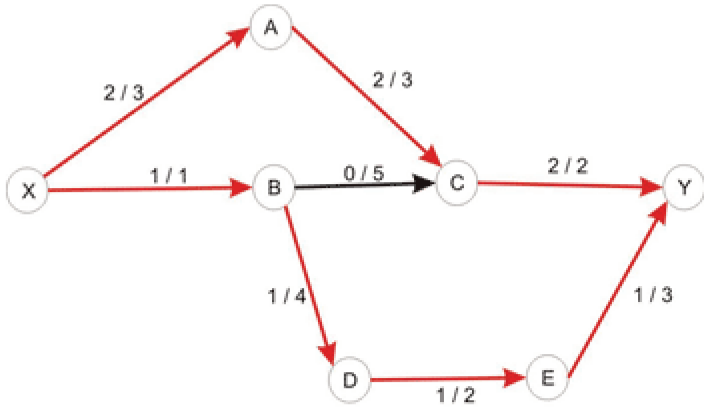


Fig. 5.5 Maxflow example

5.4.3.1 Backup path computation

With the assumption of simple link failures, the backup paths protecting against different link failures can share their resources. Thus, to verify the backup bandwidth constraints (i.e., backup admission control), it is possible and sufficient to preform separately the backup admission control for each different link risk. Such backup admission control is done by ensuring that the cumulated bandwidth of the backup paths which protect against any link risk is always lower or equal to the backup capacity.

From above, we conclude that, for any given link risk, the maximum value of the cumulated bandwidth of backup paths protecting the link risk is completely independent from the backup paths that are setup to protect against other link risks. In this way, we deduce the upper bound to the cumulated bandwidth of the backup paths which protect against the links risk $l^s = (u, v)$ as follows:

$$upper\ bound(l^s) = MaxFlow(u, v, G = (N^s, L^s / l^s))$$

where $MaxFlow$ returns the maximum flow between u and v in the network topology after pruning l^s (see below).

Max-flow Maximum flow problems involve finding a feasible flow through a single-source, single-sink flow network that is maximum. Given a network, in which every link l^s has a capacity BC_{l^s} associated with it, a starting node (the source), and an ending node (the sink). We are asked to associate a flow value f satisfying $f \leq BC_{l^s}$ for each l^s such that for every node other than the source and the sink, the sum of the values associated to the edges that enter it must equal the sum of the values associated to the edges that leave it. Maximizing

Algorithm 6: max-flow backup path pre-computation

```
   Input  : G^S(N^S, L^S)
1  begin
2      foreach l^s ∈ L^S do
3          Compute max-flow graph MFGr(l^s) on G^S(N^S, L^S/l^s);
4          Determine set of path Π(l^s) in MFGr(l^s) such that the cumulated bandwidth
             of path in Π(l^s) corresponds to the max-flow value;
5      end
6  end
```

the sum of the values associated to the links leaving the source corresponds to the maxflow in the network.

In Figure 5.5, the maximum flow solution is shown from source X to sink Y. The amount of flow/capacity correspond to the labels in the figure. The maxflow for this example is 3. From this result, we can deduce the paths to achieve the maxflow: $\{X-A-C-Y\}$ and $\{X-B-D-E-Y\}$.

The maxflow problem is solvable with polynomial time algorithms like the Ford-Fulkerson's method [FJF09]. The idea behind the algorithm is as follows: as long as there is a path from the source to the sink, with available capacity on all edges in the path, we send flow along one of the paths. Then we find another path, and so on. The procedure is repeated until no path available capacity can be determined.

The max-flow value for l^s corresponds to the maximum amount of flow that can be protected (or that are protectable) on the primary link l^s. In Algorithm 6, we pre-compute for each substrate link l^s of capacity BC_{l^s} the set of backup paths $\Pi(l^s)$ allowing to maximize the flows between the extremity nodes of l^s, by applying Ford-Fulkerson's algorithm [FJF09] after pruning l^s. Each path π in $\Pi(l^s)$ is stored and associated with a capacity BC_π and a backup residual bandwidth BR_π.

5.4.3.2 Backup path verification

When a VN request arrives, Algorithm 7 is run to determine the mapping. The pre-computed set of backup paths are used to determine the costs of substrate links (line 4-7) according to Equation 5.11. A substrate link l^s is protectable on a virtual link l^v if there is a path $\pi \in \Pi(l^s)$ such that $BR_\pi \geq bw(l^v)$.

After cost determination, a shortest primary path is determined (line 9). This path is then protected with the selection and configuration of pre-computed backup paths (lines 15-20).

Algorithm 7: Backup pre-verification based link mapping

Input : virtual network request $G^V(N^V, L^V)$
Output : link mapping and backup solution

begin
  foreach $l^v \in L^V$ **do**
    foreach $l^s \in L^S$ **do**
      Set $cost(l^v, l^s) = \frac{\varepsilon}{FR_{l^s}}$;
      if $\forall \pi \in \Pi(l^s) : BR_\pi \leq bw(l^v)$ **then**
        $cost(l^v, l^s) = cost(l^v, l^s) + [-log(1 - P_{l^s})]$
      end
    end
    Determine shortest $cost(l^v, l^s)$ based primary path π;
    if $\pi = null$ **then**
      Free pre-allocated resource;
      return no solution;
    end
    else
      Pre-allocate primary resource;
      foreach $l^s \in \pi$ **do**
        Select one backup path such that $\pi' \in \Pi : bw(l^v) \leq BR_{\pi'}$
        if π' *exists* **then**
          Pre-allocate backup resource on links of π';
          $BR_{\pi'} \leftarrow BR_{\pi'} - bw(l^v)$;
        end
      end
    end
  end
  return link mapping and backup solution;
end

As the maximum number of paths in any set of backup path $\Pi(l^s)$ is lower than $|L^S|$, the the worst-case complexity of Algorithm 7 corresponds to $O(|L^V|\ |L^S|^2)$.

5.5 Performance evaluation

5.5.1 Environment

To include the protection, we extend the simulation described in the previous chapter. The simulation architecture is depicted in Figure 5.6. VN requests are generated by GT-ITM (see Appendix A), and are added to the queue with their arrival and depart events. Link failures

follow the Weibull distribution with the same parameters as in Chapter 4. To protect against the link failures, backup mapping is implemented and added to the embedding module. By preforming the embedding algorithm, primary and backup resources are allocated on substrate network. When a VN request leaves, the primary and backup resources are freed. When a link failure occurs, the protected flows are rerouted to the backup paths, whereas the unprotected flows are affected by the failure.

For the performances study, we use different instances of networks, including a small well connected network, a medium network (less connected network compared to the first one) and a random network. We also set different substrate link failure probability models for the scenarios. The details of the differences between the scenarios are described later in the next sub-sections. Here we show the common configurations.

On the substrate network, the CPU capacity of each node is randomly chosen within [1000, 1500]. The bandwidth capacity is randomly selected within [1000, 1500]. The primary and backup capacities are determined by a primary capacity ratio τ ($0 \leq \tau \leq 1$) that is configurable for each substrate network.

The number of virtual nodes of each VN follows a uniform distribution between 3 and 8. The virtual nodes are interconnected with probability 0.4. The CPU and bandwidth demands are uniformly chosen in [0, 20].

The VN request arrival process is Poisson with arrival rate λ per 100 time units. In our experiments, we compare the methods with different arrival rates λ. This latter varies from 8 (low load) to 12 (high load). The life time of each VN follows an exponential distribution with an average of 2000 time units. Each simulation lasts for 1×10^5 time units.

5.5.2 Compared methods

For the comparison study, we use two categories of protected VNEs: (1) full protection (FP) based VNE and (2) best effort protection (BEP) based VNE. With the first category, a VN request is satisfied only when all the corresponding primary links are protected whereas we accept and accommodate the VN requests of the second category even when some primary links are not protected.

We select two methods in the first category, referred as *Basic_FP* and *SOL_FP*. *Basic_FP* uses classical shortest path algorithm to compute primary mapping without backup feasibility verification, while *SOL_FP* adopts simple on-line backup verification method (Section 5.4.2).

In the second category of protected VNEs, we select 3 methods: *Basic_BEP*, *SOL_BEP* and *MFP_BEP*. With these last methods, the acceptation of a VN request is determined only by the available primary resources. *Basic_BEP* and *SOL_BEP* have the same mapping

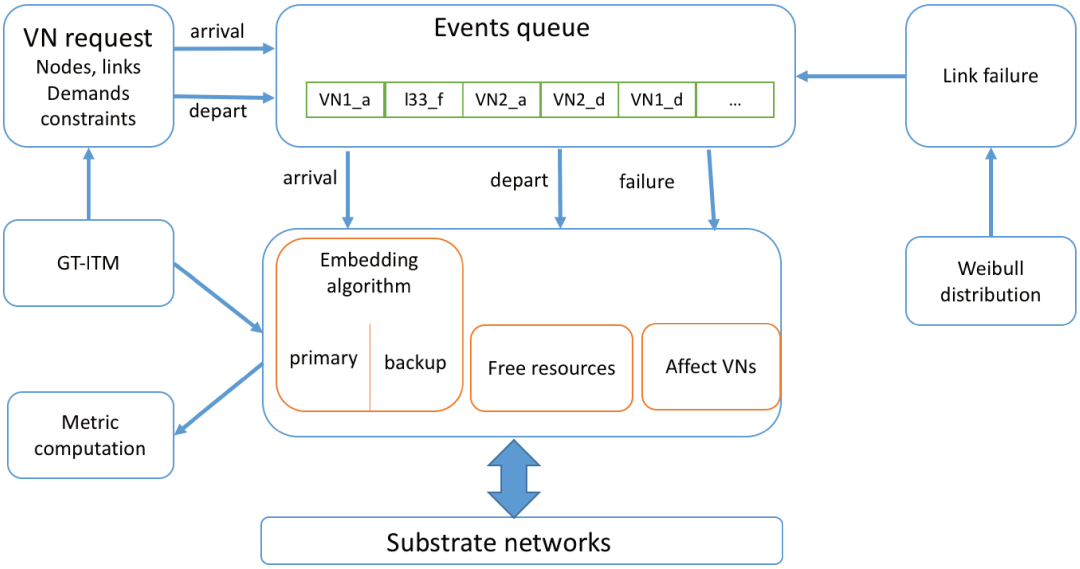


Fig. 5.6 Simulation procedure with protection

methods as *Basic_FP* and *SOL_FP*, respectively. *MFP_BEP* applies the max-flow based backup path pre-verification method presented in Section 5.4.3.

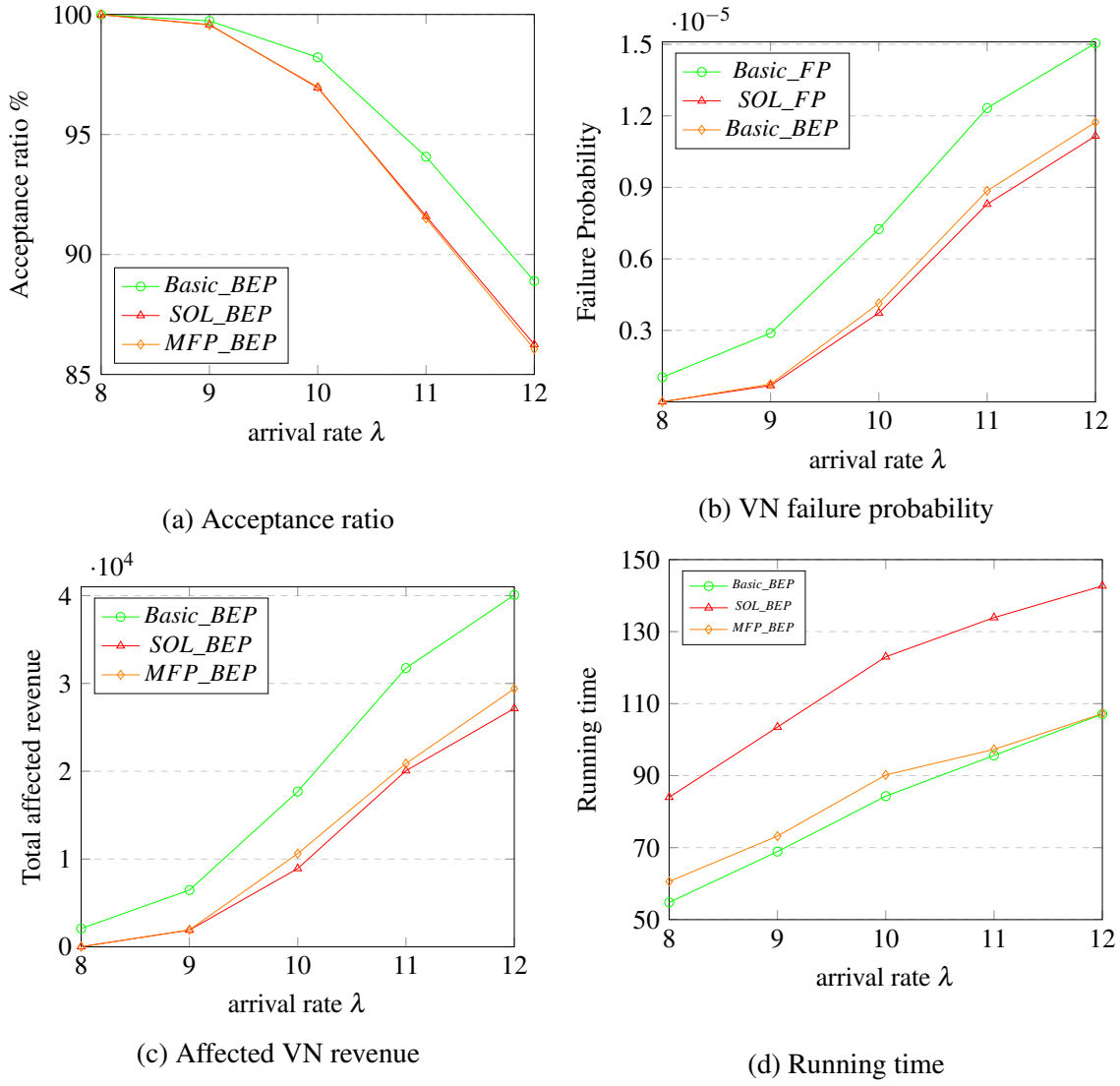
5.5.3 Metrics

In order to measure the performances of the compared methods, we use the following metrics:

- VN acceptance ratio: this metric is defined in Section 3.6.4 and it corresponds to the ratio of accepted VNs over the VN requests.
- VN failure probability: this metric (defined in 4.5.3) shows the failure probability of a VN.
- affected revenue: this metric is also defined in 4.5.3 and it shows the negative effect of the link failure.
- computational time: this metric measures the running time of the compared methods.
- full protection ratio (FPR): it computes ratio of full protected VNs over the accepted VNs,

$$FPR = \frac{full\_num}{acc\_num}$$

where acc_num is the number of accepted VNs and $full\_num$ is the number of full protected VNs.



(a) Acceptance ratio

(b) VN failure probability

(c) Affected VN revenue

(d) Running time

Fig. 5.7 Scenario 1: small size network

5.5.4 Scenario 1: small network

We first compare the methods (*Basic_BEP*, *SOL_BEP* and *MFP_BEP*) on the European optical network (*cost239*) which contains 11 vertices and 26 edges. Cost239 is a well connected network (minimum degree is 3). In this scenario, we assume an identical link failure model. Each substrate link has the same link failure probability (1×10^{-4}). This model is useful for networks with similar link characteristics. The ratio of the primary bandwidth corresponds to 0.75 of the total capacity.

The comparison results are shown in Figures 5.7a, 5.7b, 5.7c and 5.7d. For different arrival rates λ, Figures 5.7a, 5.7b, 5.7c and 5.7d depict respectively the evolution of the

acceptance ratios of VNs, the VN failure probability, the total affected revenue and the computation time of algorithms.

In Figure 5.7a, we see that *Basic_BEP* has VN acceptance ratio slightly better than our proposals (*SOL_BEP* and *MFP_BEP*). *Basic_BEP* focuses only on the primary mapping so it is normal that this method can embed more VNs than our methods. We deduce that the backup feasibility verification has a slight side effect on the primary mappings.

Figures 5.7b and 5.7c are obtained by a random generation of link failure (according to the probability law). Figure 5.7c shows clearly that *SOL_BEP* and *MFP_BEP* have lower affected revenue than *Basic_BEP*. More specifically, VN failure probability in Figure 5.7b has been cut down by our proposals *SOL_BEP* and *MFP_BEP*. Note that the difference between our 2 proposals is tiny.

In Figure 5.7d, we see that the running times of *Basic_BEP* and *MFP_BEP* are close and much smaller than those of *SOL_BEP*. *MFP_BEP* uses pre-computed backup paths to select the protectable links, *SOL_BEP* spends much time on the on-line computation of all the backup paths.

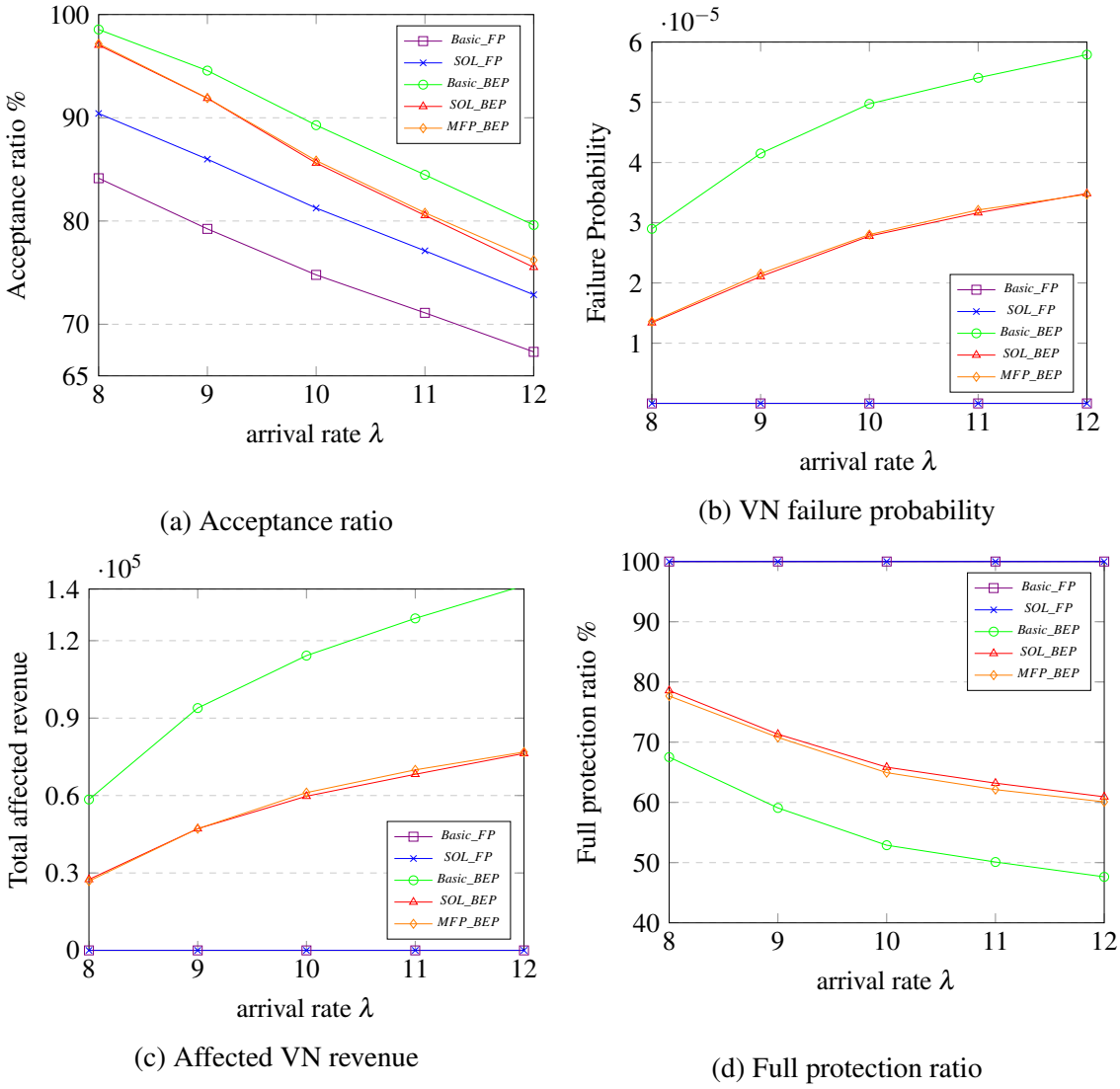
5.5.5 Scenario 2: medium size network

We retrieved a real medium size network (*germany50*) from SNDlib (see Appendix A). This network interconnects German cities and is composed of 50 vertices and 88 edges. The minimum degree of this network topology is 2. It is less connected than cost239. There is nearly no direct link between extremity nodes. Therefore, the path between extremity nodes are quite long. Different from scenario 1, the link failure probability is not the same. The link failure probability follows a uniform distribution on $[0,\ 1 \times 10^{-4}]$. This random probability model is suitable for a network with various physical link characteristics.

Figures 5.8a, 5.8b, 5.8c and 5.8d show respectively the evolution of the VN acceptance ratios, the VN failure probability, the total affected revenue and the full protection ratio.

In Figure 5.8a, *Basic_BEP*, *SOL_BEP* and *MFP_BEP* accept more VNs than *Basic_FP* and *SOL_FP*. In case of insufficient backup resource, accepting VNs with some unprotected links can largely increase the VN acceptance ratio. *Basic_BEP* improves and has the highest acceptance ratio at the cost of a very high failure probability (Figure 5.8b) and a low full protection ratio (Figure 5.8d). The methods (*SOL_BEP* and *MFP_BEP*) adopting our proposals remedy this problem by the pre-verification of backup resources at the step of primary mapping. Besides, our proposals decrease the affected revenue (Figure 5.8c) and thus increase the SP profit.

By comparing *Basic_FP* and *SOL_FP*, we deduce that with the same level of reliability (full protection), our proposed method *SOL_FP* accept more VNs in Figure 5.8a and has a



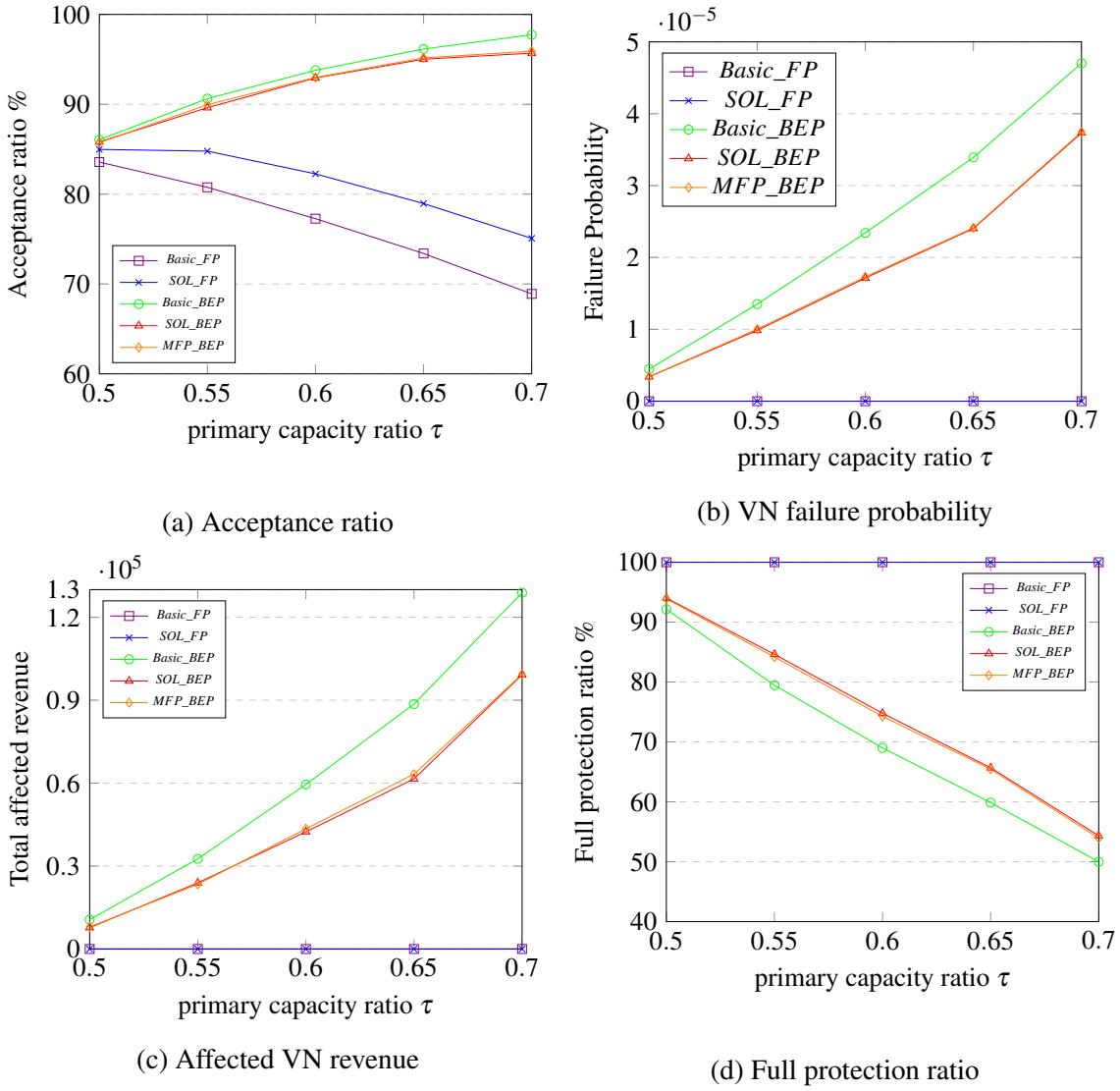
(a) Acceptance ratio

(b) VN failure probability

(c) Affected VN revenue

(d) Full protection ratio

Fig. 5.8 Scenario 2: medium network



(a) Acceptance ratio

(b) VN failure probability

(c) Affected VN revenue

(d) Full protection ratio

Fig. 5.9 Scenario 3: primary capacity ratio τ

better utilization of resource. Since the primary and backup mappings are independent with *Basic_FP*, more VNs are rejected due to the lack of backup resources.

5.5.6 Scenario 3: primary capacity ratio τ

In this scenario, we study the impact of primary capacity ratio τ on the performances of the compared methods. The substrate network, which is generated by GT-ITM, is composed of 30 nodes and 55 links. τ varies from 0.5 to 0.7. The VN arrival rate is 10 for each τ. The link failure probability follows a uniform distribution in $[0,\ 1 \times 10^{-4}]$.

In Figure 5.9a, with the increase of τ, the difficulty to full protect a VN increases (recall that the increase of τ decreases the backup resources). In Figures 5.9b and 5.9c, the difference between our methods (*SOL_BEP* and *MFP_BEP*) and *Basic_FP* increases as τ augments from 0.5 to 0.65. However, the difference stabilizes and does not increase for $\tau \geq 0.7$. In Figure 5.9d, *SOL_BEP* and *MFP_BEP* show higher full protection ratios than *Basic_BEP*.

To summarize, this last scenario shows that less the backup resources are, better our proposals are. When the backup resources are large and sufficient, there is almost no difference between our proposals and the basic ones.

5.5.7 Conclusion of simulation

The simulations allows us to conclude that:

- with a slight decrease of VN acceptance, the backup feasibility verification improves significantly the reliability for different networks;
- pre-computing backup paths according to max-flow algorithm makes the backup feasibility verification method time-saving;
- with the same level of reliability, our backup feasibility verification methods are efficient in terms of resource utilization;
- in case of insufficient backup resources, it is better to accept the non protected VNs and to use the failure avoidance method to minimize the VN failure probability.

5.6 Conclusion

In a virtualized network environment, virtual networks share the physical resources. A single link failure can lead to fatal dysfunction of several virtual networks. As a result, network reliability is a desirable and, in our opinion, indispensable for virtual networks.

To enhance the survivability of VNs, survivable network design methods are used. Restoration computes a backup path after failure occurrence whereas protection pre-computes backup paths before failures so that the disruption time is reduced.

In order to protect against single link failures, enough resources should be preserved for backup paths. Classical protection methods consider separately primary mapping and backup paths, leading to non optimal resilience against failures.

In this chapter, we proposed a novel virtual network protection framework, which maps the primary links in such a way that backup path feasibility is taken into consideration. In

our proposals, we first select the protectable links which reduce the resource allocations for primary mappings. When full protection of VNs is not guaranteed, we proposed to include the less vulnerable links. After describing an exact solution that maximizes reliability while reducing resource allocations, we proposed 2 scalable and efficient heuristics.

Our proposals are validated by simulations. The numeric results show that our propositions reduce the VN failure probability with slight decrease of VN acceptance.

Chapter 6

Conclusion and perspective

6.1 Conclusion

Network Virtualization (NV) is one of the key technologies for the future network. It allows the creation of various specific-purpose independent logical networks (virtual networks) on top of a shared physical network (substrate network). The VNE (Virtual Network Embedding) problem, which consists in finding an efficient mapping between substrate network and virtual network in terms of resource allocation, is a main challenge for network virtualization.

A user-specific virtual network may spread over substrate networks of several operators. We thus have a VNE problem over multi-domain, each domain being the substrate network of a single operator. In order to achieve the multi-domain VNE, domains need to cooperate whereas it is probable that each domain does not expose its complete information to others. With limited information disclosed by protocols like BGP, existing solutions mainly focus on the partitioning of a multi-domain VN to each domain. The inter-domain connections are determined independently by shortest path algorithms. This leads to the selection of the same peering links for the mapping and results in an inefficient resource utilization of intra domain mapping.

To address this issue, we propose a mapping approach which improves VNE by including both the intra domain links with the peering links in the mapping computation. Our proposal simplifies the mapping that is done in two stages: VN partitioning and link mapping. A series of iterative downsizing link mapping is performed by leveraging the VNP layer partial information. The downsizing algorithm maps the intra domain and peering links in the same multi-commodity flow problem so that the peering links are co-optimized with the intra domain links. Furthermore, we explore the problem of determining the domain sequence which improves the mapping and proposes a recursive approach that determines efficient mapping solutions.

The survivability of VN is also an important issue of network virtualization. It aims to ensure service continuity by avoiding as much as possible the failures and/or to repair the affected flows upon failures. The recovery of the network should be transparent for end-users.

For efficient and reliable VNs, the survivability should be taken into consideration at the step of primary mapping. In this way, the survivable VNE selects a reliable primary mapping and often a backup mapping which can deal with the network failures.

Network failure may occur both on substrate layer or virtual layer. Both the links and nodes can fails. In addition, failures may be multiple. To provide the survivability for VNE, pre-failure and post failure methods can be adopted. Post failure methods react after the failure occurrence by activating the backup restoration procedures. Pre-failure methods pre-compute survivable mappings before the failures and often reserve resources for the backup mechanism (to ensure enough resources after the recovery). Failure avoidance and failure protection are two main categories in pre-failure methods.

Failure-avoidance provides a primary routing which is determined in a way that failures affect as little as possible the VNs. Such technique generally determines routes which bypass the network components which are statistically the most vulnerable to failures. Without extra configurations, this technique enhances the network reliability.

After studying and proving that the VNE problem which minimizes the failure probability is NP-hard, we proposed new efficient heuristics implementing the failure avoidance. Two main cases of networks are treated: (1) network with infinite resources on links and (2) network with limited resources on links. The first one corresponds to the case where the resources are much higher than the demands, whereas the second one correspond to the case where the substrate link resources are comparable to the demands.

For the case of infinite resources on links, we prove that the VNE problem optimizing the failure probability is equivalent to the Steiner tree problem.

For the case of limited resources on links, we first propose an exact ILP formulation that includes the resource constraints. Then, we give two efficient heuristics to solve the problem: baseline heuristic and reinforced heuristic. The baseline heuristic optimizes only the failure probability by running the shortest path algorithm which maps the virtual links on the most reliable substrate links. The reinforced heuristic improves the precedent heuristic by including the resource (i.e. the bandwidth) factor to the cost function so that it also reduces the bandwidth utilization.

Failure avoidance technique improves the reliability but cannot guarantee the service continuity upon any failure. As a result, failure protection, which provides a high reliability level, should be implemented and eventually combined with the failure avoidance technique.

Protection pre-computes backup paths and reserves the backup resources. Three types of protection can be used: node protection, path protection and shared link protection. Virtual node protection replaces the affected nodes by migration. Path protection, which searches for a disjoint backup path for the primary path, suffers from the high recovery latency problem. Shared link protection determines one backup path for each link on the primary path.

For rapid recovery, the shared link protection is preferred to the path protection. Besides, to save the bandwidth, the backup paths protecting against different risks should share their resources. Due to the difficulty of conjoint optimization of the primary and backup resources, the backup path computations are often performed independently from the primary mapping. This can lead to the rejection of some backup paths because of inefficient choice of the primary paths.

To address the problem, we propose a protection-level-aware survivable VNE. In our proposal, the existence of the backup paths is checked and used for primary mapping. The virtual links are mapped to the primary paths which offer the best possibility of protection and failure avoidance. In other words, we combine the failure avoidance with the protection such that the survivability is optimized.

For the checking of backup path existence, we propose two methods: on-line method and max-flow based method. The first one computes on-line for each virtual link and substrate link a backup path, whereas the backup path computations are done off-line (at network initialization) and quickly with the max-flow based method.

6.2 Perspective

Network virtualization is a key technique for the future network and it has been implemented in laboratory and industry environments. In this thesis, we study and focus on the resource allocation problem in network virtualization. Our objective is to provide new efficient and survivable virtual network embeddings under various substrate and virtual requirements. We gives some perspectives in this area:

- The protection method itself is very costly. It is not necessary for service providers to have a full protection of every service. Some services, which are sensitive to disruption, need full protection, whereas others not. According to the Service-level agreement (SLA), the survivable VNE should be able to adapt different reliability levels with minimal resource consumption. This leads to some new parameters and objective functions related to the different reliability levels. Sometimes, multiple criterion (bandwidth, probability, delay, etc.) need to be taken into account. Multi-objective optimization could be a working direction of this kind of problem.

- We only studied the link failure case. Node protection is also an important issue that should be solved. The failure of intermediate substrate node can be seen as the failure of all its adjacent substrate links. In this way, the failure of a substrate node can be treated as a failure of multiple links. Multi-failure protection methods can be adopted.
- We have studied the multi-domain VNE problem. The problem of scalability still exists. If the virtual network request concerns many domains all over the world, the organization of the network need to be evaluated. The current information exchange model is not suitable for the case of large number of domains. New information exchange model need to be developed.
- The network virtualization is not an independent technique. To implement and commercialize the network virtualization, various techniques (SDN, NFV) need to be integrated. SDN provides a good centralized architecture and facilitate the information exchange and management. NFV provides virtualization of function and the separation of function with physical equipment. By combining NV, SDN and NFV, we are looking forward to a full virtualization of network. Some new issues can arise during this process.

References

[APST05] Thomas Anderson, Larry Peterson, Scott Shenker, and Jonathan Turner. Overcoming the internet impasse through virtualization. *Computer*, (4):34–41, 2005.

[ARN16] Max Alaluna, Fernando MV Ramos, and Nuno Neves. (literally) above the clouds: Virtualizing the network over multiple clouds. In *NetSoft Conference and Workshops (NetSoft), 2016 IEEE*, pages 112–115. IEEE, 2016.

[BHFDM12] Juan Felipe Botero, Xavier Hesselbach, Andreas Fischer, and Hermann De Meer. Optimal mapping of virtual networks with hidden hops. *Telecommunication Systems*, 51(4):273–282, 2012.

[BHH+13] I. B. Barla, K. Hoffmann, M. Hoffmann, D. A. Schupke, and G. Carle. Shared protection in virtual networks. In *Proc. IEEE Int. Conf. Communications Workshops (ICC)*, pages 240–245, June 2013.

[BHK12] Abdeltouab Belbekkouche, Md Hasan, and Ahmed Karmouch. Resource discovery and allocation in network virtualization. *Communications Surveys & Tutorials, IEEE*, 14(4):1114–1128, 2012.

[CB09] NM Mosharaf Kabir Chowdhury and Raouf Boutaba. Network virtualization: state of the art and research challenges. *IEEE Communications magazine*, 47(7), 2009.

[CB10] NM Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862–876, 2010.

[CMDTM16] C. Colman-Meixner, C. Develder, M. Tornatore, and B. Mukherjee. A survey on resiliency techniques in cloud computing infrastructures and applications. *Communications Surveys & Tutorials, IEEE*, PP(99):1, 2016.

[CRB09] NM Mosharaf Kabir Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Virtual network embedding with coordinated node and link mapping. In *INFOCOM*, pages 783–791. IEEE, 2009.

[CSB10] Mosharaf Chowdhury, Fady Samuel, and Raouf Boutaba. Polyvine: policy-based virtual network embedding across multiple domains. In *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, pages 49–56. ACM, 2010.

[CSZ+11] Xiang Cheng, Sen Su, Zhongbao Zhang, Hanchi Wang, Fangchun Yang, Yan Luo, and Jie Wang. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Computer Communication Review*, 41(2):38–47, 2011.

[DRP13a] David Dietrich, Amr Rizk, and Panagiotis Papadimitriou. Autoembed: automated multi-provider virtual network embedding. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 465–466. ACM, 2013.

[DRP13b] David Dietrich, Amr Rizk, and Panagiotis Papadimitriou. Multi-domain virtual network embedding with limited information disclosure. In *IFIP Networking Conference, 2013*, pages 1–9. IEEE, 2013.

[FBTB+13] Andreas Fischer, Juan Felipe Botero, M Till Beck, Hermann De Meer, and Xavier Hesselbach. Virtual network embedding: A survey. *Communications Surveys & Tutorials, IEEE*, 15(4):1888–1906, 2013.

[FJF09] LR Ford Jr and DR Fulkerson. Maximal flow through a network. In *Classic papers in combinatorics*, pages 243–248. Springer, 2009.

[FSF13] C. Fuerst, S. Schmid, and A. Feldmann. Virtual network embedding with collocation: Benefits and limitations of pre-clustering. In *Proc. IEEE 2nd Int Cloud Networking (CloudNet) Conf*, pages 91–98, November 2013.

[GBM+16] Rafael L Gomes, Luiz F Bittencourt, Edmundo RM Madeira, Eduardo Cerqueira, and Mario Gerla. Bandwidth-aware allocation of resilient virtual software defined networks. *Computer Networks*, 100:179–194, 2016.

[GDTV14] R. Guerzoni, Z. Despotovic, R. Trivisonno, and I. Vaishnavi. Modeling reliability requirements in coordinated node and link mapping. In *Proc. IEEE 33rd Int. Symp. Reliable Distributed Systems*, pages 321–330, October 2014.

[GQW+14] Bingli Guo, Chunming Qiao, Jianping Wang, Hongfang Yu, Yongxia Zuo, Juhao Li, Zhangyuan Chen, and Yongqi He. Survivable virtual network design and embedding to survive a facility node failure. *Lightwave Technology, Journal of*, 32(3):483–493, 2014.

[GWMT11] T. Guo, N. Wang, K. Moessner, and R. Tafazolli. Shared backup network provision for virtual network embedding. In *Proc. IEEE Int. Conf. Communications (ICC)*, pages 1–5, June 2011.

[GWQ+15] Kailing Guo, Ying Wang, Xuesong Qiu, Wenjing Li, and Ailing Xiao. Particle swarm optimization based multi-domain virtual network embedding. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 798–801. IEEE, 2015.

[HKA13] Sandra Herker, Ashiq Khan, and Xueli An. Survey on survivable virtual network embedding problem and solutions. In *ICNS 2013, The Ninth International Conference on Networking and Services*, pages 99–104, 2013.

[HLAZ11] Ines Houidi, Wajdi Louati, Walid Ben Ameur, and Djamal Zeghlache. Virtual network provisioning across multiple substrate networks. *Computer Networks*, 55(4):1011–1023, 2011.

[HMT16] A. Hmaity, F. Musumeci, and M. Tornatore. Survivable virtual network mapping to provide content connectivity against double-link failures. In *Proc. 12th Int. Conf. the Design of Reliable Communication Networks (DRCN)*, pages 160–166, March 2016.

[HZ14] Benfei Hu and Dong Zhang. Survivable virtual network solution based on multi-embedding algorithm. In *Proc. th Int Wireless Communications, Networking and Mobile Computing (WiCOM 2014) Conf*, pages 425–430, September 2014.

[JK12] Abdallah Jarray and Ahmed Karmouch. Column generation approach for one-shot virtual network embedding. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 863–868. IEEE, 2012.

[JK15] A. Jarray and A. Karmouch. Cost-efficient mapping for fault-tolerant virtual networks. *IEEE Transactions on Computers*, 64(3):668–681, March 2015.

[KMB81] L Kou, George Markowsky, and Leonard Berman. A fast algorithm for steiner trees. *Acta informatica*, 15(2):141–145, 1981.

[KRV+15] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[Kui12] Fernando A Kuipers. An overview of algorithms for network survivability. *ISRN Communications and Networking*, 2012, 2012.

[LCP+05] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93, 2005.

[LDZ+14] G. Luo, H. Ding, J. Zhou, J. Zhang, Y. Zhao, B. Chen, and C. Ma. Survivable virtual optical network embedding with probabilistic network-element failures in elastic optical networks. In *Proc. 13th Int Optical Communications and Networks (ICOCN) Conf*, pages 1–4, November 2014.

[LK09] Jens Lischka and Holger Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 81–88. ACM, 2009.

[LML10] H. W. Lee, E. Modiano, and K. Lee. Diverse routing in networks with probabilistic failures. *IEEE/ACM Transactions on Networking*, 18(6):1895–1907, December 2010.

[LSC15] Shuopeng Li, Mohand Yazid Saidi, and Ken Chen. Survivable virtual network embedding with resource sharing and optimization. In *Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS), 2015 International Conference on*, pages 1–6. IEEE, 2015.

[LSC16a] Shuopeng Li, Mohand Yazid Saidi, and Ken Chen. A cloud-oriented algorithm for virtual network embedding over multi-domain. In *Local Computer Networks Workshops (LCN Workshops), 2016 IEEE 41st Conference on*, pages 50–57. IEEE, 2016.

[LSC16b] Shuopeng Li, Mohand Yazid Saidi, and Ken Chen. Multi-domain virtual network embedding with coordinated link mapping. *Advances in Science, Technology and Engineering Systems Journal*, 2(3):545–552, 2016.

[LSC17a] Shuopeng Li, Mohand Yazid Saidi, and Ken Chen. A failure avoidance oriented approach for virtual network reliability enhancement. In *Proc. IEEE Int. Conf. Communications (ICC)*, 2017.

[LSC17b] Shuopeng Li, Mohand Yazid Saidi, and Ken Chen. Survivable services oriented protection-level-aware virtual network embedding (under submission process). *International Journal of Network Management*, 2017.

[MAB+08] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[MSG+16] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262, 2016.

[OPTW07] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. In *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, April 2007. http://sndlib.zib.de, extended version accepted in Networks, 2009.

[PLG+16] Mahsa Pourvali, Kaile Liang, Feng Gu, Hao Bai, Khaled Shaban, Samee Khan, and Nasir Ghani. Progressive recovery for network virtualization after large-scale disasters. In *Computing, Networking and Communications (ICNC), 2016 International Conference on*, pages 1–5. IEEE, 2016.

[QGH+11] Chunming Qiao, Bingli Guo, Shanguo Huang, Jianping Wang, Ting Wang, and Wanyi Gu. A novel two-step approach to surviving facility failures. In *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference*, pages 1–3. IEEE, 2011.

[RB13] Muntasir Raihan Rahman and Raouf Boutaba. Svne: Survivable virtual network embedding algorithms for network virtualization. *IEEE Transactions on Network and Service Management*, 10(2):105–118, 2013.

[SFAM13] Oussama Soualah, Ilhem Fajjari, Nadjib Aitsaadi, and Abdelhamid Mellouk. Pr-vne: Preventive reliable virtual network embedding algorithm in cloud's network. In *Global Communications Conference (GLOBECOM), 2013 IEEE*, pages 1303–1309. IEEE, 2013.

[SWP+09] Gregor Schaffrath, Christoph Werle, Panagiotis Papadimitriou, Anja Feldmann, Roland Bless, Adam Greenhalgh, Andreas Wundsam, Mario Kind, Olaf Maennel, and Laurent Mathy. Network virtualization architecture: proposal and initial prototype. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 63–72. ACM, 2009.

[SXYC14] Meng Shen, Ke Xu, Kun Yang, and Hsiao-Hwa Chen. Towards efficient virtual network embedding across multiple network domains. In *Quality of Service (IWQoS), 2014 IEEE 22nd International Symposium of*, pages 61–70. IEEE, 2014.

[SYAL13] Gang Sun, Hongfang Yu, Vishal Anand, and Lemin Li. A cost efficient framework and algorithm for embedding dynamic virtual network requests. *Future Generation Computer Systems*, 29(5):1265–1277, 2013.

[Wax88] Bernard M Waxman. Routing of multipoint connections. *IEEE journal on selected areas in communications*, 6(9):1617–1622, 1988.

[WLQL16] Ying Wang, Xiao Liu, Xuesong Qiu, and Wenjing Li. Prediction-based survivable virtual network mapping against disaster failures. *International Journal of Network Management*, 26(5):336–354, 2016.

[WSW12] Cong Wang, Shashank Shanbhag, and Tilman Wolf. Virtual network mapping with traffic matrices. In *Communications (ICC), 2012 IEEE International Conference on*, pages 2717–2722. IEEE, 2012.

[WWC14] Z. Wang, J. Wu, and D. Cheng. Coding-aware virtual network mapping for surviving single link failure. In *Proc. IEEE Int. Conf. Communications (ICC)*, pages 3025–3030, June 2014.

[WWW+14] Z. Wang, J. Wu, Y. Wang, N. Qi, and J. Lan. Survivable virtual network mapping using optimal backup topology in virtualized sdn. *China Communications*, 11(2):26–37, February 2014.

[XWM+14] A. Xiao, Y. Wang, L. Meng, X. Qiu, and W. Li. Topology-aware virtual network embedding to survive multiple node failures. In *Proc. IEEE Global Communications Conf*, pages 1823–1828, December 2014.

[YAQ12] Hongfang Yu, Vishal Anand, and Chunming Qiao. Virtual infrastructure design for surviving physical link failures. *The Computer Journal*, 2012.

[YAQS11] Hongfang Yu, Vishal Anand, Chunming Qiao, and Gang Sun. Cost efficient design of survivable virtual infrastructure to recover from facility node failures. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

[YCL+15] Hui Yang, Lei Cheng, Guangjun Luo, Jie Zhang, Yongli Zhao, Huixia Ding, Jing Zhou, and Yang Wang. Survivable virtual optical network embedding with probabilistic network-element failures in elastic optical networks. *Optical Fiber Technology*, 23:90–94, 2015.

[YQA+10] Hongfang Yu, Chunming Qiao, Vishal Anand, Xin Liu, Hao Di, and Gang Sun. Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–6. IEEE, 2010.

[YSzXY11] Yang Yu, Chen Shan-zhi, Li Xin, and Wang Yan. Rmap: An algorithm of virtual network resilience mapping. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*, pages 1–4. IEEE, 2011.

[YWK11] Wai-Leong Yeow, Cédric Westphal, and Ulas C Kozat. Designing and embedding reliable virtual infrastructures. *ACM SIGCOMM Computer Communication Review*, 41(2):57–64, 2011.

[YYRC08] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review*, 38(2):17–29, 2008.

[ZA06] Yong Zhu and Mostafa H Ammar. Algorithms for assigning substrate network resources to virtual network components. In *INFOCOM*, volume 1200, pages 1–12, 2006.

[ZCP12] Xian Zhang, Xiuzhong Chen, and Chris Phillips. Achieving effective resilience for qos-aware application mapping. *Computer Networks*, 56(14):3179–3191, 2012.

[ZPC11] Xian Zhang, Chris Phillips, and Xiuzhong Chen. An overlay mapping model for achieving enhanced qos and resilience performance. In *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2011 3rd International Congress on*, pages 1–7. IEEE, 2011.

Appendix A

Substrate network generation

A.1 SNDlib

SNDlib [1] is a library of test instances for Survivable fixed telecommunication Network Design. Its purpose is:

- to make realistic network design test instances available to the research community,
- to serve as a standardized benchmark for testing, evaluating, and comparing network design models and algorithms,
- to be a source of information and resources related to fixed network design, and to provide a contact platform for researchers and practitioners working in this field.

To this end, the library includes 26 network design instances. The file of an instance is composed of nodes and link description. For example, the instance *germany50* has 50 nodes with location and 88 edges. The file is described as follows,

```
# network germany50

# NODE SECTION
#
# <node_id> [(<longitude>, <latitude>)]
NODES (
  Aachen ( 6.04 50.76 )
  Augsburg ( 10.90 48.33 )
  Bayreuth ( 11.59 49.93 )
```

[1] Web site: http://sndlib.zib.de

```
  .
  .
  .
  Wesel ( 6.37 51.39 )
  Wuerzburg ( 9.97 49.78 )
)

# LINK SECTION
#
# <link_id> ( <source> <target> ) <pre_installed_capacity> <pre_installed_capacity_co

LINKS (
  L1 ( Duesseldorf Essen ) 0.00 0.00 0.00 0.00 ( 40.00 3290.00 )
  L2 ( Dortmund Essen ) 0.00 0.00 0.00 0.00 ( 40.00 3290.00 )
  .
  .
  .
  L87 ( Wuerzburg Nuernberg ) 0.00 0.00 0.00 0.00 ( 40.00 3720.00 )
  L88 ( Regensburg Nuernberg ) 0.00 0.00 0.00 0.00 ( 40.00 3720.00 )
)
```

A.2 GT-ITM

The Georgia Tech Internetwork Topology Models (GT-ITM) [2] are built on top of the Stanford GraphBase (SGB), a platform of data structures and routines for representing and manipulating graphs.

Here is an example of usage of ITM. The following is a valid specification file called r10:

```
# <method keyword> <number of graphs> [<initial seed>]
# <n> <scale> <edgemethod>  <alpha> [<beta>] [<gamma>]
geo 3
30 10 3 .2
```

If we run 'itm r10' then three pure random graphs of 30 nodes each will be created. The nodes of each graph will be generated in a 10 by 10 (logical) grid. The probability of an edge

[2] Web site: www.cc.gatech.edu/projects/gtitm/

is 0.2, as given by the $< alpha >$ parameter; $< beta >$ and $< gamma >$ are not needed for this method. The output files will be named: r10-0.gb, r10-1.gb, r10-2.gb.

Appendix B

Cplex optimization file

The optimization problem is solved by Cplex[1]. For a optimization problem,

$$Min \;\; x_1 + 2\,x_2 + 3\,x_3 + x_4$$

$$-x_1 + x_2 + x_3 + 10\,x_4 \leq 20$$

$$x_1 - 3\,x_2 + x_3 \leq 30$$

$$x_2 - 3.5x_4 = 0$$

$$0 \leq x_1 \leq 40$$

$$2 \leq x_4 \leq 3$$

The input to Cplex is a *.lp* file shown as follows,

```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
General
 x4
End
```

[1]Web site: https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

The output of Cplex for this problem is,

```
Found incumbent of value 46.000000 after 0.01 sec. (0.00 ticks)
Tried aggregator 2 times.
Aggregator did 1 substitutions.
Reduced MIP has 2 rows, 3 columns, and 6 nonzeros.
Reduced MIP has 0 binaries, 1 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.00 sec. (0.01 ticks)
Tried aggregator 1 time.
Reduced MIP has 2 rows, 3 columns, and 6 nonzeros.
Reduced MIP has 0 binaries, 1 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.00 sec. (0.00 ticks)
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 4 threads.
Root relaxation solution time = 0.00 sec. (0.00 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                          46.0000      163.0000           254.35%
      0     0      125.2083     1       46.0000      125.2083        3  172.19%
*     0+    0                         122.5000      125.2083            2.21%
      0     0        cutoff           122.5000                      3    0.00%
Elapsed time = 0.02 sec. (0.04 ticks, tree = 0.00 MB, solutions = 2)

Root node processing (before b&c):
  Real time             =    0.02 sec. (0.04 ticks)
Parallel b&c, 4 threads:
  Real time             =    0.00 sec. (0.00 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    0.02 sec. (0.04 ticks)
The solutions begin here :
x1 40.0
x2 10.5
```

```
x3 19.5
x4 3.0
```