UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RODRIGO RUAS OLIVEIRA

Toward Cost-efficient, DoS-resilient Virtual Networks with ORE: Opportunistic Resilience Embedding

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Prof. Dr. Marinho Pilla Barcellos
Advisor

Porto Alegre, May 2013

CIP – CATALOGING-IN-PUBLICATION

Oliveira, Rodrigo Ruas

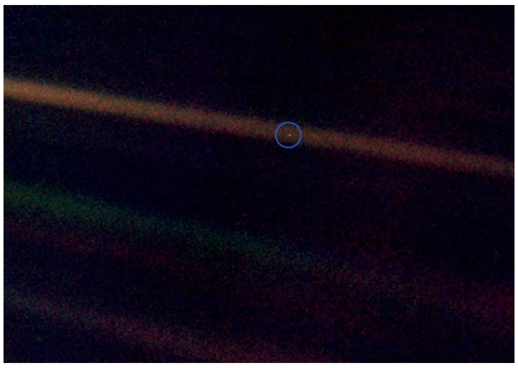
Toward Cost-efficient, DoS-resilient Virtual Networks with ORE: Opportunistic Resilience Embedding / Rodrigo Ruas Oliveira. – Porto Alegre: PPGC da UFRGS, 2013.

111 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2013. Advisor: Marinho Pilla Barcellos.

1. Network virtualization. 2. Virtual network embedding. 3. Resource allocation. 4. Optimization. 5. Algorithms. 6. Resilient. 7. Survivable. 8. Security. I. Barcellos, Marinho Pilla. II. Title.

"Pale Blue Dot" is a photograph of planet Earth taken in 1990 from 6 billion kilometers by the Voyager 1 spacecraft. In this photograph, our planet appears as a tiny blue dot, immerged in the darkness of the universe, and surrounded by a thin beam of light.



"From this distant vantage point, the Earth might not seem of any particular interest. But for us, it's different. Consider again that dot. That's here. That's home. That's us. On it everyone you love, everyone you know, everyone you ever heard of, every human being who ever was, lived out their lives. The aggregate of our joy and suffering, thousands of confident religions, ideologies, and economic doctrines, every hunter and forager, every hero and coward, every creator and destroyer of civilization, every king and peasant, every young couple in love, every mother and father, hopeful child, inventor and explorer, every teacher of morals, every corrupt politician, every 'superstar', every 'supreme leader', every saint and sinner in the history of our species lived there – on a mote of dust suspended in a sunbeam.

The Earth is a very small stage in a vast cosmic arena. Think of the rivers of blood spilled by all those generals and emperors so that in glory and triumph they could become the momentary masters of a fraction of a dot. Think of the endless cruelties visited by the inhabitants of one corner of this pixel on the scarcely distinguishable inhabitants of some other corner. How frequent their misunderstandings, how eager they are to kill one another, how fervent their hatreds. Our posturings, our imagined self-importance, the delusion that we have some privileged position in the universe, are challenged by this point of pale light. Our planet is a lonely speck in the great enveloping cosmic dark. In our obscurity – in all this vastness – there is no hint that help will come from elsewhere to save us from ourselves.

The Earth is the only world known, so far, to harbor life. There is nowhere else, at least in the near future, to which our species could migrate. Visit, yes. Settle, not yet. Like it or not, for the moment, the Earth is where we make our stand. It has been said that astronomy is a humbling and character-building experience. There is perhaps no better demonstration of the folly of human conceits than this distant image of our tiny world. To me, it underscores our responsibility to deal more kindly with one another and to preserve and cherish the pale blue dot, the only home we've ever known."

— Carl Sagan
Pale Blue Dot: A Vision of the Human Future in Space, 1994

AGRADECIMENTOS

À família. Agradeço primeiramente aos meus pais pela paciência, carinho e suporte durante todos estes anos. Sem eles eu não estaria aqui hoje, nem teria alcançado o que alcancei ao longo da vida. Também sou muito grato às minhas irmãs pelo amor e amizade; o suporte nos momentos difíceis; e as alegrias compartilhadas.

Aos meus amigos. Agradeço aos grandes amigos e colegas de apartamento Renan Maffei e Pedro Marcos, pelo bom convívio e pela ótima amizade, compartilhados ao longo de toda esta jornada. Agradeço também a todos os colegas do laboratório 208 pelo companheirismo, pela ajuda, e pelas boas risadas. Em especial, agradeço ao Leonardo, ao Daniel e ao Miguel pelas parcerias de trabalho e ao Lucas pela recente ajuda na correção deste texto. Também agradeço aos amigos de longa data Rogê Martins e Fabricio Motta.

Aos professores. Agradeço à professora Luciana Buriol e ao professor Luciano Gaspary por terem ajudado na elaboração do trabalho com extensas discussões e com críticas construtivas. Também agradeço à antiga orientadora Silvia Botelho pelas oportunidades concedidas e pelos ensinamentos repassados durante a graduação.

Ao orientador. Por ter sido extremamente capaz e profissional em sua função; por ter ajudado não só em questões acadêmicas, mas também pessoais; por ter me apoiado na tomada de decisões importantes; por ter, tão brilhantemente, repassado seus ensinamentos; e por tantas outras coisas, agradeço ao atual orientador Marinho Barcellos.

TABLE OF CONTENTS

LIST OF ALGORITHMS

LIST OF DEFINITIONS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Recently, the Internet's success has prevented the dissemination of novel networking architectures and protocols. Specifically, any modification to the core of the network requires agreement among many different parties. To address this situation, Network Virtualization has been proposed as a diversifying attribute for the Internet. This paradigm promotes the development of new architectures and protocols by enabling the creation of multiple virtual networks on top of a same physical substrate. In addition, applications running over the same physical network can be isolated from each other, thus allowing them to coexist independently.

One of the main advantages of this paradigm is the use of isolation to limit the scope of attacks. This can be achieved by creating different, isolated virtual networks for each task, so traffic from one virtual network does not interfere with the others. However, routers and links are still vulnerable to attacks and failures on the underlying network. Particularly, should a physical link be compromised, all embedded virtual links will be affected.

Previous work tackled this problem with two main strategies: using backup resources to protect against disruptions; or live migration to relocate a compromised virtual resource. Both strategies have drawbacks: backup resources tend to be expensive for the infrastructure provider, while live migration may leave virtual networks inoperable during the recovery period.

This dissertation presents ORE (Opportunistic Resilience Embedding), a novel embedding approach for protecting virtual links against substrate network disruptions. ORE's design is two-folded: while a *proactive* strategy embeds virtual links into multiple substrate paths in order to mitigate the initial impact of a disruption, a *reactive* one attempts to recover any capacity affected by an underlying disruption. Both strategies are modeled as optimization problems. Additionally, since the embedding problem is $\mathcal{NP}$-Hard, ORE uses a Simulated Annealing-based meta-heuristic to solve it efficiently. Numerical results show that ORE can provide resilience to disruptions at a lower cost.

Keywords: Network virtualization, virtual network embedding, resource allocation, optimization, algorithms, resilient, survivable, security.

Provendo Resiliência de Baixo Custo às Redes Virtuais com ORE: mapeamento com resiliência oportunística (*Opportunistic Resilience Embedding*)

RESUMO

O atual sucesso da Internet vem inibindo a disseminação de novas arquiteturas e protocolos de rede. Especificamente, qualquer modificação no núcleo da rede requer comum acordo entre diversas partes. Face a isso, a Virtualização de Redes vem sendo proposta como um atributo diversificador para a Internet. Tal paradigma promove o desenvolvimento de novas arquiteturas e protocolos por meio da criação de múltiplas redes virtuais sobrepostas em um mesmo substrato físico. Adicionalmente, aplicações executando sobre uma mesma rede física podem ser isoladas mutuamente, propiciando a independência funcional entre as mesmas.

Uma de suas mais promissoras vantagens é a capacidade de limitar o escopo de ataques, através da organização de uma infraestrutura em múltiplas redes virtuais, isolando o tráfego das mesmas e impedindo interferências. Contudo, roteadores e enlaces virtuais permanecem vulneráveis a ataques e falhas na rede física subjacente. Particularmente, caso determinado enlace do substrato seja comprometido, todos os enlaces virtuais sobrepostos (ou seja, alocados neste) serão afetados.

Para lidar com esse problema, a literatura propõe dois tipos de estratégias: as que reservam recursos adicionais do substrato como sobressalentes, protegendo contra disrupções; e as que utilizam migração em tempo real para realocar recursos virtuais comprometidos. Ambas estratégias acarretam compromissos: o uso de recursos sobressalentes tende a tornar-se custoso ao provedor de infraestrutura, enquanto a migração de recursos demanda um período de convergência e pode deixar as redes virtuais inoperantes durante o mesmo.

Esta dissertação apresenta ORE (Opportunistic Resilience Embedding – Mapeamento com Resiliência Oportunística), uma nova abordagem de mapeamento de redes para proteger enlaces virtuais contra disrupções no substrato físico. ORE é composto por duas estratégias: uma *proativa*, na qual enlaces virtuais são alocados em *múltiplos caminhos* para mitigar o impacto de uma disrupção; e uma *reativa*, a qual tenta recuperar, *parcial ou integralmente*, a capacidade perdida nos enlaces virtuais afetados. Ambas são modeladas como problemas de otimização. Ademais, como o mapeamento de redes virtuais é $\mathcal{NP}$-Difícil, ORE faz uso de uma meta-heurística baseada em Simulated Annealing para resolver o problema de forma eficiente. Resultados numéricos mostram que ORE pode prover resiliência a disrupções por um custo mais baixo.

Palavras-chave: Virtualização de redes, mapeamento de redes virtuais, alocação de recursos, otimização, algoritmos, resiliente, sobrevivente, segurança.

1 INTRODUCTION

Since its dissemination, the Internet has unleashed an unprecedented wave of innovation. This success is partially attributed to the organization of the Internet architecture (CASADO et al., 2012), a stack composed of mainly five layers (namely physical, link, network, transport and application). Towards the bottom of the stack, an array of new networking technologies – from wireless to optical – has expanded the Internet's capacity and reach, while towards the top of the stack numerous unforeseen applications – from the web to social networks – have changed human communication worldwide (KOPONEN et al., 2011). However, the popularity of the Internet has also become the Achilles' heel of network innovation.

Due to its multi-provider nature, modifying the core of the existing network architecture requires consensus among competing stakeholders (CHOWDHURY; BOUTABA, 2010). This formed a networking "narrow waist", that is, technologies at the top and bottom of the stack evolved significantly, whereas technologies at the center of the stack remained relatively stagnated – Figure 1.1 provides an illustration of the network narrow waist, as depicted by Stuckmann and Zimmermann (STUCKMANN; ZIMMERMANN, 2009). More recently, as technologies and applications matured, the demand for novel network functionality and requirements have arisen. However, alterations to the Internet architecture have become restricted to simple incremental updates and deployment of new network technologies have become increasingly difficult (CHOWDHURY; BOUTABA, 2010). This issue became commonly known as "Internet ossification" (TURNER; TAYLOR, 2005) or "Internet tussle" (CLARK et al., 2005). In response, network virtualization has been proposed as a means to overcome this impasse by providing diversification to the future inter-networking paradigm (ANDERSON et al., 2005).

1.1 Context

The growing attention around network virtualization resides on the fact that it provides an elegant solution to the current networking innovation tussle (PAUL; PAN; JAIN, 2011; CHOWDHURY; BOUTABA, 2010). The main idea behind this paradigm is that, instead of betting on a one-size-fits-all approach, it argues in favor of an heterogeneous environment, enabling the creation of specifically tailored network infrastructures and fostering the development of new architectures and protocols. Several practical benefits follow the use of this paradigm, such as, coexistence of different architectures, testing and debugging of novel applications, optimization of hardware utilization, and seamless migration of network services (KHAN et al., 2012). Essentially, such benefits are achieved by the instantiation of multiple virtual networks on top of a same physical substrate network.

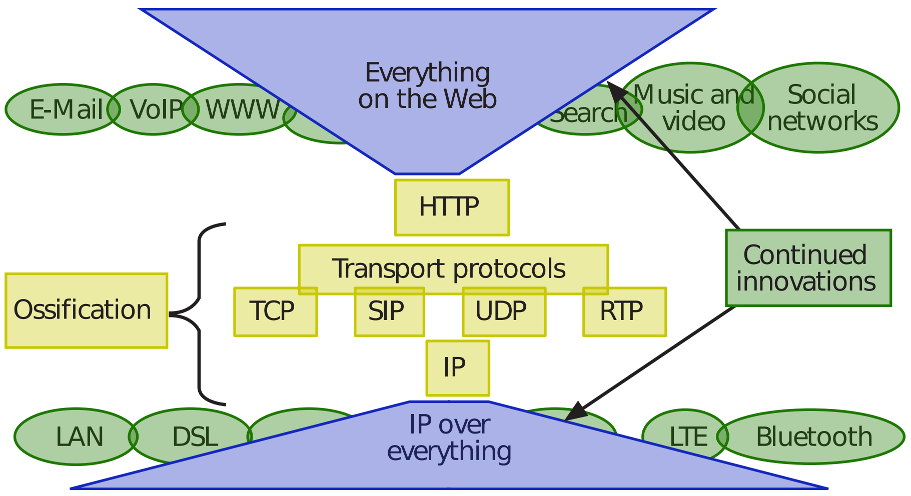


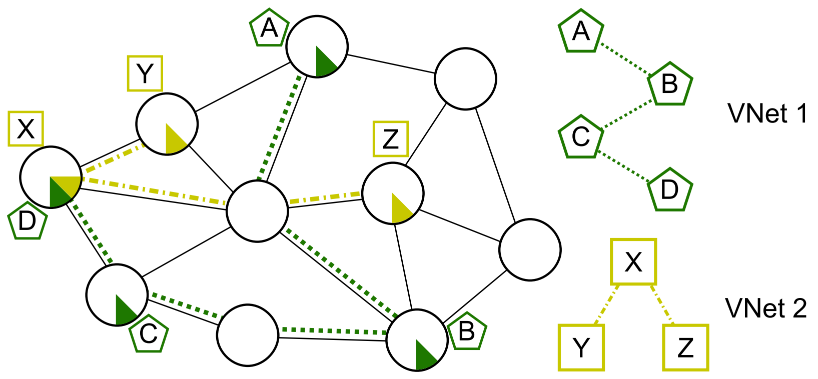
Figure 1.1: Networking “narrow waist” and the ossification problem, adapted from Stuckmann and Zimmermann (STUCKMANN; ZIMMERMANN, 2009).

In an abstract view, each virtual network defines a virtual topology to be overlaid on top of the devices of a physical network. The virtual network is composed of virtual routers and virtual links demanding a variety of requirements. The physical network is composed of physical routers, which support some virtualization technology, interconnected by physical links. Both physical routers and physical links have attributes and constrained capabilities (e.g., router/link type, location, available technologies, processing/storage/bandwidth capacity, delay, etc), that dictate which features are available to the overlaid virtual networks (HOUIDI et al., 2011). Therefore, virtual networks are free to implement potentially arbitrary services, as long as the physical network is capable of offering the necessary features.

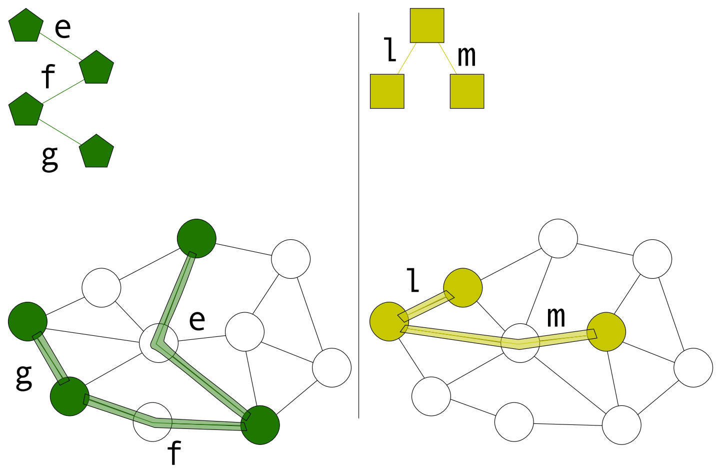
In order to match virtual requirements with physical capabilities and achieve the best utilization of resources, the physical network administrator invokes an allocation process called the *Virtual Network Embedding* (VNE)[1]. This process consists of selecting feasible substrate devices and assigning shares of their resources to the overlaid virtual networks (CHOWDHURY; RAHMAN; BOUTABA, 2012; CHENG et al., 2012; BUTT; CHOWDHURY; BOUTABA, 2010). Thus, each virtual node is mapped to a physical node that supports all requirements; likewise, each virtual link is mapped to a physical link or a continuous, loop-free physical path that supports all requirements. Additionally, the endpoints of the physical link (or path) selected to embed a virtual link should be the physical nodes embedding the endpoints of the virtual link.

Figure 1.2 exemplifies a possible mapping of two virtual networks into a substrate network. Figure 1.2(a) shows the overall mapping and resource consumption, highlighting the virtual node embedding, whereas Figure 1.2(b) shows each individual mapping and highlights the virtual link embedding. Aimed at simplicity, the figure represents resource consumption by slices on the physical nodes. Slices could represent a fraction of the node’s processing power, packet processing throughput, number of entries in the routing table, etc. Although the figure does not explicitly illustrate link requirements, they should also be taken into consideration.

[1] Also known as Virtual Network “Allocation” or “Mapping”. The terms “embedding”, “allocating” and “mapping” will be used interchangeably throughout the text.



(a) Two virtual networks mapped into a substrate network. Each virtual node is placed into a physical node, consuming a fraction of the physical node's resources.



(b) Each virtual link is mapped into a substrate link or a substrate path. Note that the selected substrate link/path should connect the endpoints of the virtual link.

Figure 1.2: Illustration of two virtual networks embedded in a substrate network.

To effectively provide the aforementioned environment, network virtualization employs some of the principles of a conventional hypervisor, essentially, hardware abstraction, resource partitioning, and isolation (KHAN et al., 2012). Hardware abstraction, together with programmability, allows the development of novel network architectures which are free from the limitations of specific protocols. In their turn, resource partitioning and isolation enable the instantiation of separated domains, which can block interference among virtual networks (BELBEKKOUCHE; HASAN; KARMOUCH, 2012; KHAN et al., 2012).

In summary, a proper network virtualization environment can offer several potential benefits. Particularly, it can increase security and fault tolerance by allowing the creation of different, isolated virtual networks for each task, thus limiting the scope of attacks and error propagation, as well as avoiding incompatibility among technologies. However, even if isolation can fully protect virtual networks among themselves, it cannot protect them from attacks and failures in the substrate network.

1.2 Motivation

Generally, when a virtual network is mapped to a substrate, its topology uses only a subset of the available devices. Furthermore, current VNE algorithms keep this subset as small as possible to avoid waisting resources for future allocations. This means that, after the VNE process, virtual networks will potentially have less diversity than their embedding physical network. In addition, the mapping process (i.e., VNE) tends to increase dependency on certain physical resources, allowing an attacker to launch Denial-of-Service (DoS) attacks on virtual networks by strategically choosing which nodes and/or links of the underlying physical substrate will be compromised. Indeed, strategically planned fiber cuts have been known to affect thousands of customers[2]. Further, recent news suggest that numerous networking equipment on the Internet operate with insecure firmware. Flaws of this nature allow attackers to perform DoS attacks from remote locations[3]. More specifically, if the VNE process fails to consider such scenarios, a failure or a successful attack on the underlying physical substrate will affect a potentially large number of embedded elements.

When oblivious to substrate network attacks and failures, the VNE process tends to make, in fact, virtual networks more susceptible to DoS. In an attempt to tackle this problem, previous research focused mainly on two strategies: i) either setting aside additional resources as backup, thus guaranteeing some level of protection (e.g., $k-1$) with redundant virtual nodes or links (RAHMAN; AIB; BOUTABA, 2010; GUO et al., 2011; YEOW; WESTPHAL; KOZAT, 2010; CHEN et al., 2010; YU et al., 2011); or ii) employing live reconfiguration and migration in order to relocate a compromised virtual node/link (WANG et al., 2008; HOUIDI et al., 2010; MARQUEZAN et al., 2010; PISA et al., 2010; MATTOS; FERNANDES; DUARTE, 2011). Both strategies have drawbacks that hinder their applicability. On one hand, setting aside backup resources may lead to expensive solutions, as the additional resources remain idle (i.e. are wasted) when there are no disruptions. On the other hand, live reconfiguration and migration schemes are designed to avoid performance degradation or sustain planned maintenances; when faced with disruptions, they may leave virtual networks inoperable during the reconfiguration period.

1.3 Contributions

This dissertation presents the design of ORE (Opportunistic Resilience Embedding), a novel approach for protecting the availability of virtual links against attacks and failures in the substrate network. In this dissertation, the term *opportunistic resilience* denotes a best effort strategy to partially or fully mitigate the impact of disruptions. ORE is a first step at studying virtual network embedding algorithms which provide joint efficient resource utilization and opportunistic resilience to the network virtualization environment. Toward this end, ORE is composed by two complementary strategies, one of them **proactive**, and the other, **reactive**. The proactive strategy aims at mitigating the initial impact of an attack by embedding each virtual link into *multiple paths*, thus avoiding that virtual links lose all their capacity. The reactive strategy attempts to partially or fully recover any capacity compromised by attacks. To achieve this goal, it uses any unaffected path (if such a path exists) to *opportunistically recover* the compromised capacity.

[2]`http://goo.gl/8KWXG`, `http://goo.gl/NeupI`

[3]`http://goo.gl/w3yNB`, `http://goo.gl/dgmtn`

The main contributions of this dissertation are summarized as follows:

- **Novel virtual network embedding algorithm.** The design of a novel embedding algorithm is provided. It aims at achieving both efficient resource utilization to the substrate network and resilience to the virtual links;
- **Mathematical modeling.** The fundamental building blocks of ORE are formalized in mathematical models. Furthermore, the reactive strategy is modeled as a complete Linear Program (LP), allowing it to be solved optimally and efficiently.
- **Meta-heuristic.** The proactive strategy in ORE requires solving the VNE problem. Unfortunately, this problem is $\mathcal{NP}$-Hard (ANDERSEN, 2002), making it unfeasible to adopt an optimal model in practice. Hence, ORE utilizes a Simulated Annealing-based algorithm. Empowered by this meta-heuristic, ORE can achieve near-optimal solutions at practical computing time.
- **Experimental findings.** ORE's potential benefits are measured with simulations. According to results, ORE can reduce both bandwidth loss and severity of disruptions by increasing the number of paths used per virtual link. Further, when compared to backup-oriented schemes, ORE achieves higher acceptance rate.

1.4 Organization

The remaining of this dissertation is organized as follows. Chapter 2 introduces the network virtualization background, compares related work in resilient virtual network embedding, and presents the main concerns with multipath routing. Chapter 3 describes the design of ORE as well as its supporting formulas. Numerical experiments, performed in a simulated environment, are shown in Chapter 4, while Chapter 5 concludes this dissertation and discusses possibilities for future work.

2 LITERATURE REVIEW

This chapter describes the fundamental concepts related to this dissertation. It begins by introducing the network virtualization paradigm. Then, it presents the state-of-the-art in resilient virtual network embedding, and compares it to conventional network protection (e.g., protection in optical network). Finally, it discusses multipath routing, which is a central aspect for the development of ORE.

2.1 Network Virtualization

This section presents an overview of the network virtualization paradigm and describes the Virtual Network Embedding problem, that is, its primary resource allocation problem.

2.1.1 Overview

As discussed previously, a networking environment supports network virtualization if it allows coexistence of multiple virtual networks on the same physical substrate. Virtual Networks (VNets) are used by an entity to provide end-to-end services with customized protocols. Physical substrate networks are used as a resource pool for virtual networks. This layer is owned and operated by an entity that aims to earn a profit from leasing network resources to its customers (BELBEKKOUCHE; HASAN; KARMOUCH, 2012). The key principle behind the network virtualization paradigm is to *decouple* major elements in computer networking. This separation can be seen in three different ways, as discussed next: the physical and logical networks, the control and data planes, and the network-associated business roles.

Physical-logical network separation

While logical refers to routing functions performed over the networks, physical refers to the underlying equipment that enables these functions. Typically, an instance of a router is both physical and logical at the same time. As a result, any changes in the physical level necessarily affects the logical level (WANG et al., 2008). In contrast, network virtualization aims to break this tight coupling by enabling *virtual* instances of the logical network to be placed on top of any physical element. This enables virtual networks to dictate arbitrary topologies and use resources in an on-demand basis. Moreover, virtual routers and links can be moved freely from one physical equipment to another. Advantages are manifold: networks are able to growth, shrink, and shift according to traffic variations (GAO et al., 2012); planned maintenance operations are facilitated since routers and paths can be migrated promptly in order to avoid unnecessary disruptions

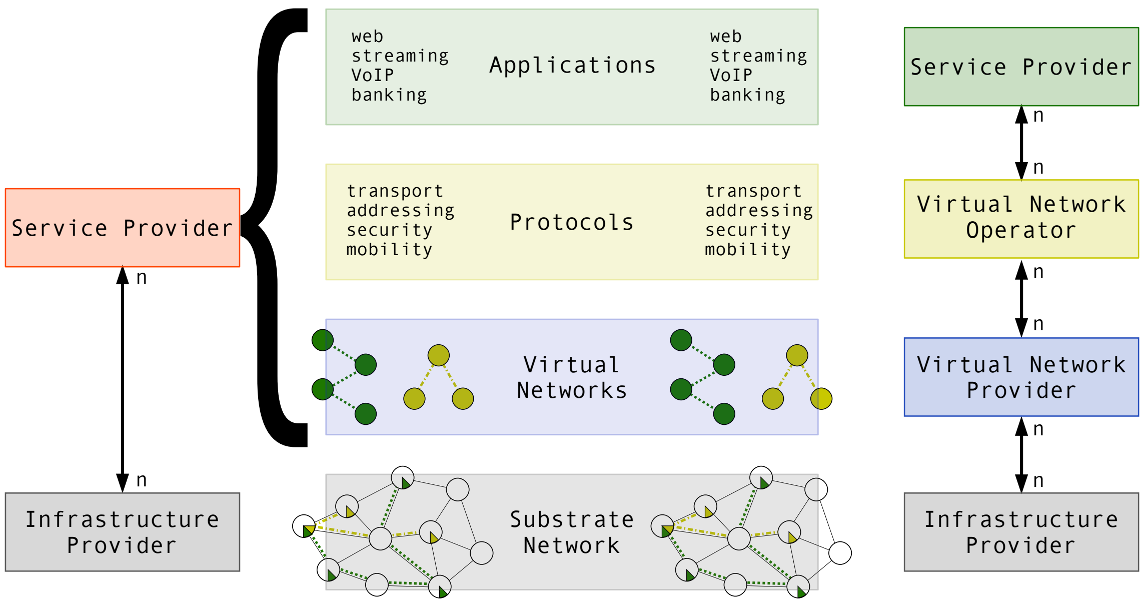


Figure 2.1: Conceptual models for the relationships between business players in network virtualization.

(MARQUEZAN et al., 2010). Moreover, this independence from the underlying network allows coexisting virtual networks which differ in terms of topology, routing and forwarding functionalities, and control protocols (BELBEKKOUCHE; HASAN; KARMOUCH, 2012).

Control-data plane separation

The network infrastructure has two main functionalities: i) performing forwarding operations to every incoming packet; and ii) defining the general behavior of the network in terms of routing, management, load balancing, etc. These functionalities are commonly known as data plane and control plane, respectively (CASADO et al., 2012). Currently, both are performed by inside the router – regardless if it is a physical or virtual one (REITBLATT et al., 2012).

Decoupling these two planes enables each one to evolve separately: the hardware can focus on data-plane operations, increasing performance, while the software can focus on building better control-plane algorithms, without requiring complex operations to deploy updates (GUDE et al., 2008). Moreover, this separation allows the control plane to be placed anywhere in the network without causing interruptions or inconsistencies to the data plane (KOPONEN et al., 2010; GUDE et al., 2008). Hence, control-data plane separation is essential to virtual router migration (as will be discussed in Section 2.2.2).

Business roles separation

The clear separation of roles between the virtual and physical networks create opportunities for novel business models. Specifically, two conceptual models have been proposed to represent the new business players in the network virtualization environment (BELBEKKOUCHE; HASAN; KARMOUCH, 2012), both are depicted in Figure 2.1. In the first model (left), a Service Provider (SP) is responsible for creating different VNets with personalized network protocols, while an Infrastructure Provider (InP) is responsible

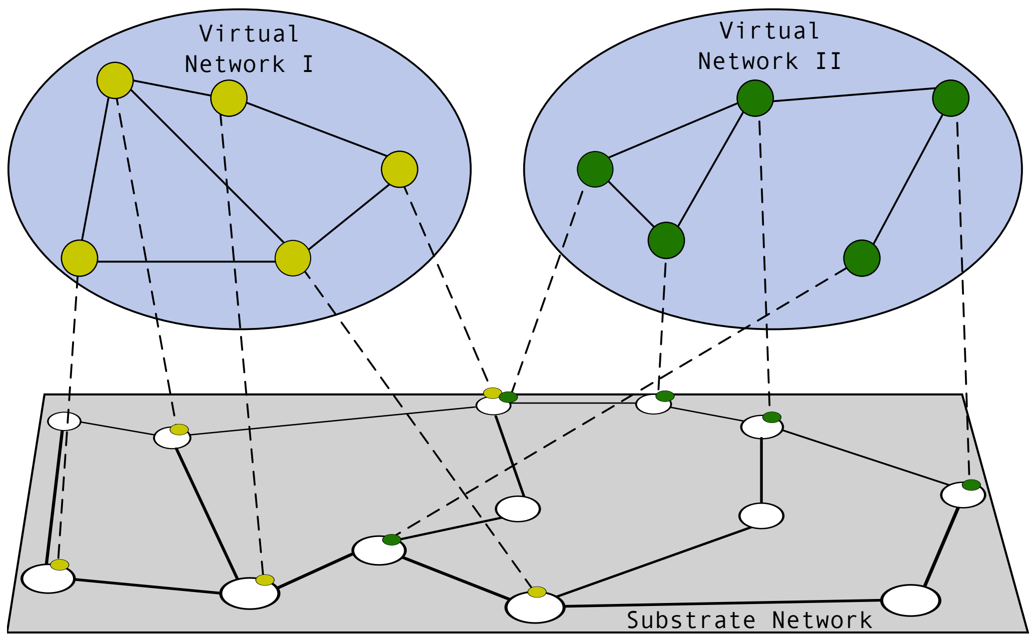


Figure 2.2: VNet elements mapped into substrate network equipments, adapted from Belbekkouche, Hasan, and Karmouch (BELBEKKOUCHE; HASAN; KARMOUCH, 2012).

for physical network management, maintenance and upgrade. This simple scenario allows SPs to deploy and manage customized end-to-end services on virtual networks by utilizing underlying network resources leased from InPs (FEAMSTER; GAO; REXFORD, 2007). Moreover, the relationship between SPs and InPs should be completely decoupled, allowing one business player to hire/serve multiple others.

In the second model (right), the SP is only responsible for creating new services and applications. The role of deploying and configuring new protocols is assigned to a Virtual Network Operator (VNO), while the creation and management of the VNet itself is performed by a Virtual Network Provider (VNP) (PAPADIMITRIOU et al., 2009). In both models, the same company can assume multiple roles without affecting the semantics behind them. In other words, a substrate network owner could design customized services (SP) to use them in its own network (InP), and hire one or more third parties to develop network-level protocols (VNO) and assemble virtual networks (VNP).

2.1.2 Virtual Network Embedding

Virtual Network Embedding (VNE) is a resource allocation problem which consists of selecting substrate (physical) network resources to embed their virtual counterparts. In this problem, a VNP issues a Virtual Network Request (VNR)[1] to one or more InPs. VNRs are issued on an *on-demand* basis describing a VNet topology and node/link requirements. The physical network and the virtual network are both composed of nodes connected by links, forming a network topology. Each virtual node should be hosted on a physical node; and each virtual link should be established over a physical link or path (i.e., a set of physical links), essentially becoming a subset of the underlying physical network (Figure 2.2).

[1]A VNR is an analogy to todays' Service Level Agreements (SLAs).

The *VNR requirements* describe which *substrate network features*, among those available, are being demanded by the VNet. Generally, requirements and features refer to a characteristic of a specific resource or group of resources, for example, the type of node or link, slices of processing power/bandwidth, and propagation delay (BAYS et al., 2012; CHOWDHURY; RAHMAN; BOUTABA, 2009, 2012; CHENG et al., 2012; YU et al., 2008; ALKMIM; BATISTA; FONSECA, 2011). Therefore, the term *attribute* is used, when possible, to denote requirements or features in general.

Attributes can be either functional or non-functional (HOUIDI et al., 2011). Functional attributes are inherent characteristics of the devices which are associated with a *functionality*, such as a specific state (location) or technology (cryptography). Such requirements are mostly static, that is, any change requires external action. Non-functional attributes are common to every equipment of a certain type and are usually *consumables* (e.g., bandwidth capacity). Table 2.1 presents a non-exhaustive list of attributes commonly seen in VNE algorithms.

Table 2.1: Attributes considered by VNE algorithms. The functional/non-functional classification was based on guidelines described in (HOUIDI et al., 2011).

Attribute	Element	Functional/ Non-Functional	Examples
Type [2]	Node/Link	F	Router, switch, host, copper, optical fiber, satellite, etc
Computing power [1-10]	Node	NF	Capacity in packets per second, number of cores, or fraction of CPU
Storage [1,3,4]	Node	NF	Amount of space in RAM/disk, or entries in routing table
Bandwidth [1-10]	Link	NF	Throughput in (M)bps
Delay [1,5]	Node/Link	F	Time required to process a packet header in a physical router or to send bits through a physical link
Location [1-3,5]	Node	F	Geographic coordinates or logical identification
Availability [2,7-9]	Node/Link	F	Level of redundancy, uptime
Confidentiality [3]	Node/Link	F	Cryptography and tunneling technology; requirement for disjoint mapping between VNet pairs
Utilization cost [2,8]	Node/Link	NF[a]	Monetary value attributed to per unit utilization of a particular resource (e.g., bandwidth)

List of references used in this table	
[1]	(ALKMIM; BATISTA; FONSECA, 2011, 2012)
[2]	(BARLA; SCHUPKE; CARLE, 2012)
[3]	(BAYS et al., 2012)
[4]	(BUTT; CHOWDHURY; BOUTABA, 2010)
[5]	(CHENG et al., 2011, 2012)
[6]	(CHOWDHURY; RAHMAN; BOUTABA, 2009, 2012)
[7]	(GUO et al., 2011)
[8]	(RAHMAN; AIB; BOUTABA, 2010)
[9]	(YEOW; WESTPHAL; KOZAT, 2010)
[10]	(YU et al., 2008)

[a] Costs are usually considered as an evaluation metric, rather than a requirement or feature.

In order to succeed, a VNE algorithm needs to find a subset of the physical substrate network which can map each element of the VNR. This requires matching VNet requirements to substrate features, which, in turn, may generate nontrivial *matching* operations depending on the type of attribute. A trivial matching operation would be, for example, to validate a location constraint (BAYS et al., 2012). In this case, each node – physical or virtual – would have a location ID. Thus, locations can be validated by comparing the ID of a virtual router to the ID of a physical router, requiring a simple equality operation ($=$). In contrast, for a nontrivial example, suppose the following case. For confidentiality reasons, two virtual networks cannot share physical equipment (BAYS et al., 2012). This requirement can be modeled as a pair of attributes, where the first represents a unique VNet ID, while the second defines which VNets are conflicting and need to be mapped disjointly. Although the model is simple, the operation involves checking the entire mapping of each pair of conflicting VNets to ensure that both are disjoint.

As currently stated, the problem tries to capture the relationship between VNP and InP and is unconcerned with the relationship between the other parties. Those relationships may originate complementary problems, such as defining new protocols or composing virtual network topologies and requirements based on high level descriptions (FEAMSTER; GAO; REXFORD, 2007; PAPADIMITRIOU et al., 2009). However, those are out of the scope of this dissertation.

2.2 Resilient Virtual Network Embedding

The Resilient Virtual Network Embedding problem is a resource allocation problem similar to the VNE problem (discussed in Section 2.1.2). The difference between these problems is that former also has to consider the interruption of the communication between virtual nodes, caused by disrupting parts of the underlying substrate network (i.e., the physical routers and physical links). This section presents the two sets of approaches used for protecting virtual networks against substrate network disruptions, namely *backup reservation*, and *live reconfiguration and migration*. It then discusses and compares these approaches to conventional network protection.

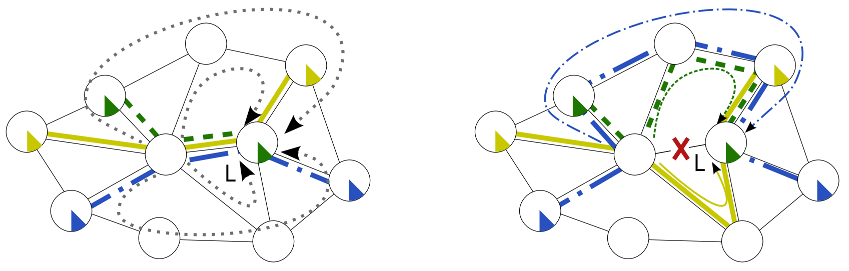
2.2.1 Backup reservation

The network virtualization community has made a recent effort to protect virtual networks against physical disruptions using backup resources. The idea is to prevent disruptions from affecting virtual networks in the first place. This is performed by creating additional copies of the original resources (denoted primary) of a virtual network and allocating these copies in disjoint substrate resources during the VNE phase. Hence, if the primary resource fails, the backup assumes its place.

This strategy is effective as the additional resources are mapped disjointly in order to avoid co-related or cascading disruptions. Its main disadvantage is that the use of backup resources may became too expensive. The extra overhead in resource consumption may hinder the ability of the substrate network to accept future requests. Further, they are wasted if no disruption occurs.

Virtual link protection

Literature proposes two ways of protecting virtual links with backup resources: *pre-allocation* of detours (RAHMAN; AIB; BOUTABA, 2010; GUO et al., 2011); and *on-*



(a) Three virtual links are embedded in the same physical link L; dotted, gray-colored lines highlight possible detours.

(b) When L fails, the virtual links change their mapping to use one of the available detours.

Figure 2.3: Illustration of the "preallocated detours" protection mechanism, based on the SVNE approach (RAHMAN; AIB; BOUTABA, 2010).

demand embedding of restoration paths (CHEN et al., 2010; GUO et al., 2011). Preallocated detours try to protect the connectivity of the endpoints of each *physical* link, while restoration paths are used to protect an entire *virtual* link.

The concept of backup detours is introduced in Survivable Virtual Network Embedding (SVNE) (RAHMAN; AIB; BOUTABA, 2010). Detours are micro-backup paths installed in neighboring links which connect the endpoints of a link. They are created prior to any virtual network request (i.e., in an empty substrate network) and they consume a pre-configured backup capacity in the physical links. Hence, if a physical link is disrupted, the virtual links traversing the interrupted link are immediately rerouted through the detours. Figure 2.3 shows an example of this procedure. Consider three simple VNets with two virtual nodes and one virtual link each. Due to a circumstance of the embedding algorithm, all of them share the same physical link L [Figure 2.3(a)]. If the link L is disrupted, all embedded virtual links will became unavailable. To avoid this problem, the mechanism redirects the traffic of each virtual link in one of the detours belonging to L [Figure 2.3(b)]. Notice that detours are oblivious to the mapping of virtual links, as they protect the connectivity of physical links. Consequently, even if a shorter path to reach the endpoint of the virtual link is available (which is the case with the blue, slash-dotted virtual link in the example), the detour still redirects the traffic to the endpoint of the disrupted physical link.

In contrast, a restoration path aims to protect each virtual link individually. They are an additional, disjoint mapping of a particular virtual link, assigned on-demand during the embedding phase. Moreover, restoration paths remain idle while the primary mapping is functional; they are only used if a disruption on a physical link occurs. In such scenarios, restoration paths are immediately activated and the traffic belonging to virtual links is rerouted (CHEN et al., 2010; GUO et al., 2011).

To improve the efficiency in resource utilization, restoration paths can be shared by different virtual links. However, to ensure full recoverability, resource sharing is only employed when primary mappings are disjoint. Figure 2.4 shows how restoration paths implement resource sharing. In the example, a VNet with three virtual nodes and two virtual links is protected from substrate link disruptions. First, if possible, primary virtual links are embedded in disjoint substrate paths (full, green-colored lines) with enough bandwidth capacity (numbers, representing Mbps). Then, the embedding algorithm chooses substrate links to embed restoration paths (dot-slashed, yellow-colored lines) such that:

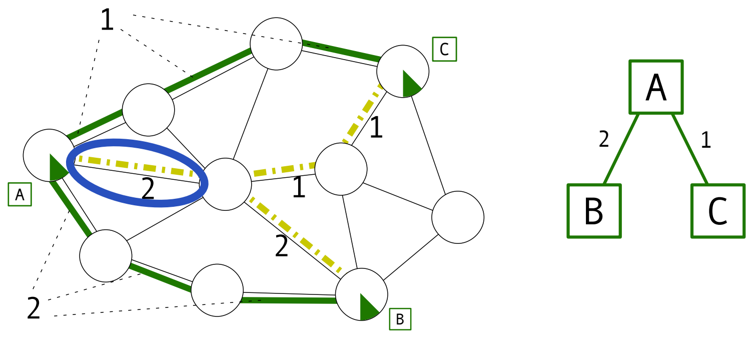


Figure 2.4: "Restoration paths" using shared backup resources to provide cheaper virtual link protection. Illustration based on the Pardalis approach (CHEN et al., 2010).

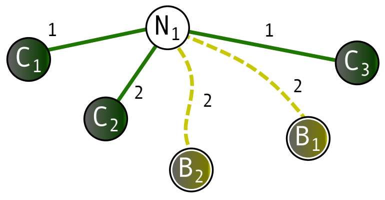
(a) each restoration path is disjoint to its primary virtual link; and (b) the amount of bandwidth reserved on a shared substrate link is the maximum value among the restoration paths using that link (this can be observed in the circled area of Figure 2.4).

Virtual node protection

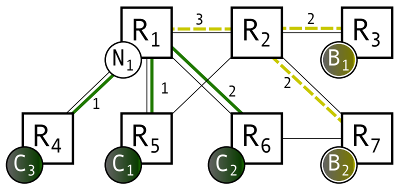
This type of mechanism tries to protect "critical" virtual nodes of a virtual network. Such nodes are required to remain always available for the VNet to retain minimal functionality. The simplest method requires the embedding algorithm to reserve one additional backup node per critical virtual node, thus providing single-failure protection. In addition, backup paths should be created to preserve the connectivity and the throughput capacity of protected nodes. In this scheme, backup paths are solely used to maintaining the topology, they do not offer direct protection for virtual links against substrate link disruptions. A failing virtual link can be recovered indirectly by reassigning one or both of its virtual nodes endpoints.

This method may became too expensive and inefficient, since it requires doubling the number of allocated critical virtual nodes, and each node is only protected against a single disruption event. A better approach consists of requesting *K* shared, redundant virtual nodes (YU et al., 2011). This allows any critical node of a protected VNet to sustain up to *K* consecutive node disruptions. Moreover, backup paths now have complex patterns since they have to maintain the connectivity of any primary critical node. Since this could easily overload the network with backup paths, authors propose two mitigation strategies: i) minimizing the number of extra paths in an optimization problem, and ii) sharing resource consumption whenever possible.

Figure 2.5 illustrates how backup virtual nodes are shared in virtual node protection schemes. Consider a single VNet with 4 virtual nodes – an unprotected node (N_1) and three critical nodes (C_{1-3}) – which has been extended with two backup nodes (B_{1-2}) [Figure 2.5(a)]. As discussed, backup nodes reproduce the connectivity and capacity of the protected nodes. Figure 2.5(b) shows a possible mapping of this VNet in a substrate network with seven physical routers (R_{1-7}). Virtual link $B_1 - N1$ is mapped in physical path $R_3 - R_2 - R_1$, while virtual link $B_2 - N1$ is mapped in $R_7 - R_2 - R_1$. In order to keep resource utilization at a minimum, backup paths are able to share bandwidth. Segments $R_3 - R_2$ and $R_7 - R_2$ need to guarantee enough bandwidth for any critical node, thus requiring 2 Mbps (same as node C_2). In contrast, the shared segment $R_2 - R_1$ needs to guarantee enough bandwidth for any *combination* of critical nodes; by sharing bandwidth



(a) Backup nodes need to reproduce the connectivity and the capacity of any protected node.



(b) To keep resource utilization at a minimum, backup paths are able to share bandwidth.

Figure 2.5: Illustration of the redundant node sharing scheme, adapted from Yeow, Westphal, and Kozat (YEOW; WESTPHAL; KOZAT, 2010).

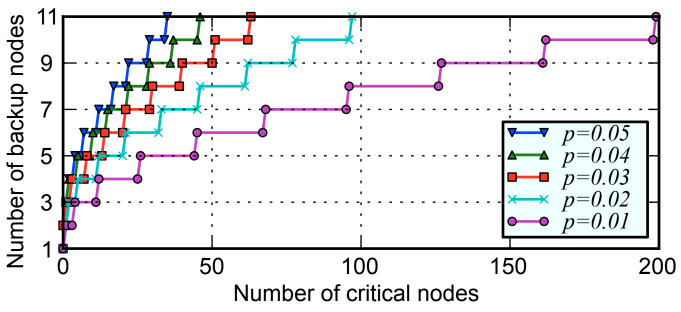
this segment requires only 3 Mbps, instead of the aggregated amount of 4 Mbps.

Looking from another angle, its possible to further improve the sharing factor of backup nodes. Opportunistic Redundancy Pooling (ORP) (YEOW; WESTPHAL; KOZAT, 2010) is a protection mechanism designed to share backup nodes among critical nodes of different virtual networks. The scheme is highly focused on random failures, but provides a very flexible and inexpensive way to use backup nodes. ORP assumes that hardware failures have known probabilities to offer a scheme based on reliability, that is, the guarantee that all virtual nodes will remain available x% of the time. The scheme leverages the fact that the amount of backup nodes required to provide some level of reliability is usually a real number, meaning that virtual networks end up being overprovisioned since the number needs to be rounded up. Therefore, the strategy proposed in ORP is to share backup nodes among different VNets, as long as the reliability of each VNet is not reduced bellow a desirable amount.

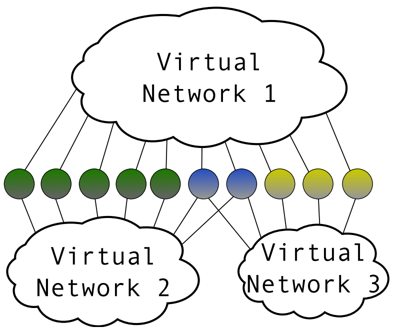
A motivational example for ORP can be seen in Figure 2.6. Consider Figure 2.6(a), which shows the amount of backup nodes required for providing a 99.999% reliability guarantee over various failure probabilities ($p = 0.01, \dots, 0.05$). As expected, when the number of critical nodes increases, the number of required backup nodes also increases. However, the curves sub-linear property can be exploited to reduce redundancy. For example, for the case of $p = 0.01$, consider three virtual networks: VNet1 has 100 critical nodes, VNet2 has 50 critical nodes, and VNet3 has 25 critical nodes [Figure 2.6(b)]. In this case, the total amount of backup nodes would be 18 (8 for VNet1, 6 for VNet2, and 4 for VNet3). This number can be reduced to 10 when VNets (with combined $n = 175$) pool their backup nodes together, saving 44.44% in additional resource consumption.

Finally, virtual node protection has also been explored in the context of regional failures. A regional failure occurs when a single disruption event causes multiple devices in a same region to stop working[2]. A possible approach for surviving such an event is to create a replica of each virtual node and link in different regions, thus recreating the entire virtual network in another location (YU et al., 2010).

[2]Such events are mostly attributed to natural disasters (e.g., earthquake, flooding, etc), hence a region usually represents a geographical location. However, regional failures also have other root causes which may result a region to have other forms. For example, electrical wiring usually create nontrivial patterns; in consequence, electrical failures may not necessarily affect a same geographical location, rather, it usually affects a logical one (BODÍK et al., 2012).



(a) Minimum number of backups needed for a reliability guarantee of five-9's (99.999%), extracted from Yeow, Westphal, Cedric, and Kozat (YEOW; WESTPHAL; KOZAT, 2010).



(b) Example of backup node sharing among three VNets, based on the ORP mechanism (YEOW; WESTPHAL; KOZAT, 2010).

Figure 2.6: Motivational example for inter-VNet sharing of backup nodes.

Joint protection

Virtual nodes and virtual links protection was recently proposed in the context of inter-cloud virtual networks (BARLA; SCHUPKE; CARLE, 2012). In this scenario, virtual links can connect two different types of nodes: the virtual cloud Data Center (DC), which hosts services for a particular Service Provider; and the virtual router, which is directly connected to (thus representing) a group of clients. The objective is to protect a service installed in the cloud from becoming unavailable to the clients. Since virtual routers are assumed to represent unique geographical locations, the approach is aimed at protecting virtual DCs and virtual links.

The proposed approach is divided into two different protection strategies: one is offered by the Virtual Network Provider, while the other is offered by the Infrastructure Provider. In the first strategy, the VNP spreads the services' VMs among two or more DCs from different InPs. Then, it creates disjoint primary and backup paths connecting DCs to virtual routers. To ensure that the protection works properly, InPs are made aware of which virtual links must be disjoint. In the second strategy, protection is provided solely by the InP. The VNP delivers a Virtual Network Request describing which nodes must remain connected. Then, the InP is responsible for allocating additional, geographically different, DCs and provide redundant paths for each virtual link.

2.2.2 Live reconfiguration and migration

In contrast to the previously mentioned proposals, some papers attempt to recalculate and reallocate virtual nodes/links whenever a disruption occurs, or migrate resources prior to disruption events. This kind of approach does not rely on backup resources, making it a cheaper solution in terms of expended resources. However, in order to avoid interruption, the whole procedure has to be planned in advance and the migrating resources (i.e., virtual routers and/or virtual links) have to remain operational during the time. This means live reconfiguration and migration schemes are better suited to prevent performance degradation (e.g., due to oversubscription of resources) or avoid disruptions caused by planned maintenance operations. In face of attacks or random failures – if no other protection mechanism is available – the affected virtual networks will became inoperable during the time it takes to perform the required operations. Nonetheless, such schemes are still useful to protect against these scenarios, as they help maintaining the internal state of a virtual router.

Related papers can be separated in two categories: the first set focus on the reconfiguration process (HOUIDI et al., 2010; MARQUEZAN et al., 2010), which involves selecting virtual resources and deciding where to migrate them; and the second set concentrates on the migration process itself (WANG et al., 2008; MATTOS; FERNANDES; DUARTE, 2011; PISA et al., 2010; KELLER et al., 2012). Both processes are described next.

Reconfiguration

Finding a new configuration for the current allocation state of virtual networks is a difficult problem. Three possible approaches are (i) using a centralized algorithm; (ii) using decentralized algorithms performed by the substrate network; (iii) using decentralized algorithms performed by the virtual networks. The centralized approach consists of running another round of the VNE algorithm with all currently allocated virtual networks. A drawback in this approach is that the algorithm has to reconsider all of the network state, instead of a potential region of interest. This can be addressed by gathering information about the network before starting the algorithm, but the process adds latency and may leave virtual networks inoperable for long periods (MARQUEZAN et al., 2009). In contrast, another approach for this problem is using a *self-organizing networks* model, which are decentralized in nature.

The second approach relies on the physical routers to perform this task, instead of relying on a central entity (HOUIDI et al., 2010; MARQUEZAN et al., 2010). Physical routers are empowered with heuristics and intelligent algorithms that make decisions regarding their embedded virtual routers. The process can be summarized in three steps. It begins with each neighboring physical router exchanging information – mostly keep-alive messages and IO utilization statistics (e.g., buffers, memory, hard disk, etc). Next, each router analyzes the exchanged information to decide which virtual resources should be moved and to which location. Finally, resources are reserved on the destination devices and the migration process is started (this process is described in the following).

The third and final approach would be for each virtual network to detect its own state and make a decision regarding when to migrate and to which location. This adds flexibility to the reconfiguration processes – that is, virtual networks would be free to instantiate their own recoverability mechanisms – but it also creates two major problems. First, it requires the physical network to disclose its own topology, breaking transparency and possibly creating security issues. Second, it breaks isolation as virtual networks need information about resource utilization, which can implicitly reveal information about other virtual networks.

Migration

On the surface, virtual router migration might not seem very different from existing Virtual Machine (VM) migration techniques. In this analogy, the physical machines would be physical routers and the VM image would be composed of routing-protocol binaries, configuration files and data-plane state. However, while a virtual machine can be frozen and restored without interruption, virtual routers must remain operational dur-

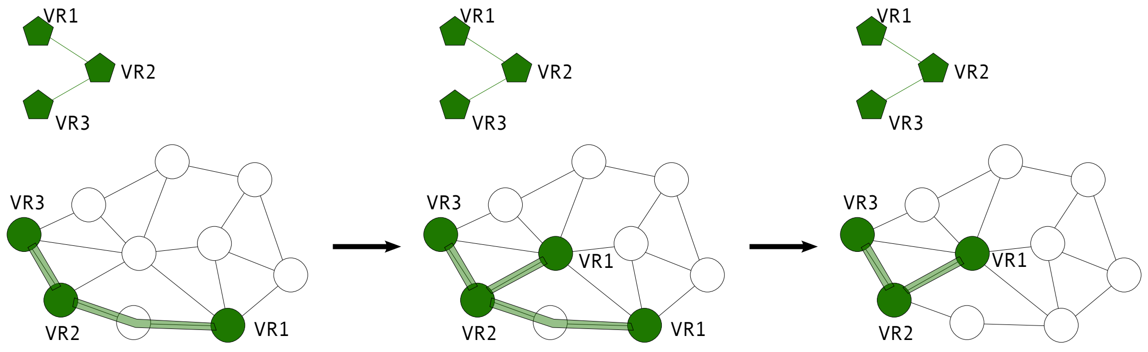


Figure 2.7: Example of a virtual router and link migration process. Virtual router 1 (VR1) is relocated from one physical router to another, while the connecting virtual link is migrated accordingly to preserve the original topology. During the process, VR1 is duplicated in both the source and target physical routers.

ing the migration process, as not to interrupt the application(s) running on top of them (WANG et al., 2008).

Figure 2.7 illustrates the migration process. Such processes rely heavily on control-data plane separation (Section 2.1.1) and require the migrating virtual router to coexist in both source and target location. First, the control software of a migrating virtual router is moved to the target location, while the data plane continues to operate. Then, a new virtual link is installed and flows begin to be transfered gradually; concurrently, routing table entries (i.e., the data plane) are inserted on-demand. Finally, the original virtual router is disabled (MATTOS; FERNANDES; DUARTE, 2011; WANG et al., 2008; PISA et al., 2010; KELLER et al., 2012).

2.2.3 Comparison to conventional network protection

Resilient virtual network embedding differs from conventional network protection in several ways. Differences mostly concern the *online/offline* natures of the problems and the type of *attributes* in consideration.

Regarding resource allocation, resilient virtual network embedding problems are online in nature; this is because virtual network requests (VNRs) are received on an on-demand basis. In contrast, conventional network survivability problems can be offline or online, depending on the focus of the optimization (MEDHI, 2006). The problem is formulated offline when the focus is to enhance the network capacity or protect permanent (or semi-permanent) scheduled requests. These problems are commonly known as network protection design (or survivable network design) problems (KATIB; MEDHI, 2012).

Offline models cannot be used without modifications since they assume that a traffic matrix is provided in advance. This assumption changes the VNE problem significantly because all VNRs would need to be known in advance, breaking the purpose of using substrate resources on-demand (BELBEKKOUCHE; HASAN; KARMOUCH, 2012). Despite this fact, some of its concepts have been adapted to the virtual network context. For example, the restoration paths method discussed in Section 2.2.1 is similar to the protection mechanism used in IP over MPLS networks. However, the model had to be redesigned to account for the online nature of virtual network requests.

Conventional network survivability can also be modeled as an online problem when requests are switched or routed on a per-unit basis and are of short duration (MEDHI, 2006). For example, an SONET/SDH network can offer on-demand high-rate optical services (HUANG; MARTEL; MUKHERJEE, 2011). This is similar to on-demand VRNs in VNE problems. However, the main difference is that connections represent pairs of pre-defined source/sink nodes, in contrast to customized virtual network topologies. Further, virtual networks can request several attributes, not only on the network links, but also on the network nodes. More specifically, each virtual router will require additional compute resources on its embedding physical router, or specialized technologies to implement its customized routing algorithms (as discussed in Section 2.1). Meanwhile, conventional networks are focused only on link-related attributes, such as throughput, delay, and other QoS parameters (e.g., jitter) (HEEGAARD; TRIVEDI, 2009; ZHANG et al., 2010). Therefore, even when the problem is formulated online, conventional network protection proposals are unsuitable for network virtualization as node embedding (i.e., placement and requirements) are not considered.

Finally, in IP networks, restoration tasks are performed by the routing algorithm, which updates the routing tables to account an unresponsive link or router. This is orthogonal to the VNE phase since the virtual networks have their own routing algorithms, possibly containing customized restoration mechanisms. Such algorithms are not widely deployed, but, in the future, they may originate multilayer virtual network protection problems by integrating aspects of these algorithms into the embedding process – similarly to multilayer protection in conventional networks (e.g., IP/MPLS over OTN) (SCHUPKE, 2012). This is out of the scope of this dissertation, as well as current resilient virtual network embedding algorithms.

2.3 Multipath Routing

One key aspect of ORE is to split the capacity of a single virtual link over multiple substrate paths, hence, relying on multipath routing. This technique consists of multiplexing incoming packets among a set of (usually few) paths that lead to the same destination. Although already applied in practice, the use of multipath routing creates two primary concerns (HUANG; MARTEL; MUKHERJEE, 2011; WANG et al., 2008): the *differential delay* and the *routing overhead*. This section explains both of these concerns and their mitigation strategies.

2.3.1 Differential delay

When traffic is sent through multiple paths, it may happen that the selected paths do not present the same end-to-end propagation delay. Particularly, the use of multiple paths with relatively different propagation delays aggravates out-of-order packet delivery, hurting the performance of protocols that guarantee reliable and ordered delivery, such as TCP. This problem is commonly known as the *differential delay* problem (SRIVASTAVA et al., 2005). The negative effect that the differential delay causes on protocols is passed to upper layers, ultimately reaching the users. An illustration of the problem can be seen in Figure 2.8. Two packets (P1 and P2) with the same destination are sent by the same origin through different paths. Packet P1 is suppose to reach the destination first, but it is sent over a path with higher end-to-end delay. When P1 finally reaches the destination, P2 has already been processed.

The differential delay poses as a major hurdle for multipath routing. In an attempt

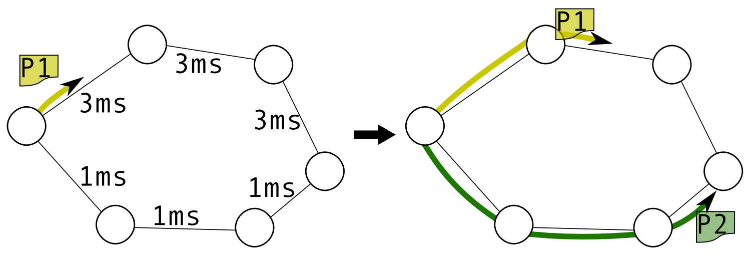


Figure 2.8: Illustration of the differential delay problem: two packets are sent to the same destination over different paths; despite being sent first, P1 arrives later because of the differences in delay between the paths.

to prevent this problem from impeding the adoption of this technique, two types of mitigation strategies have been proposed: *i*) multiplexing traffic at flow-level granularity; and *ii*) packet-level multiplexing with the aid of reordering buffers. Both strategies have aggregated costs, but the benefits (such as resilience, load balancing, and service differentiation) can overcome the trade-offs. Additionally, these strategies are considered to perform well in practice (WANG et al., 2008).

Flow-based multiplexing is not affected by differential delay. Its main requirement is identifying to which flow each packet belongs. Therefore, this technique is only applicable to devices that have this capability (e.g., network layer routers). In IP networks, flows can be identified by five fields in the packet header which are immutable during the lifetime of a flow (namely the src/dst IP addresses, the transport protocol, and the src/dst ports). Once in possession of this information, flows can be multiplexing either by *hashing* or by *caching* this information (HE; REXFORD, 2008). Hashing involves first dividing the hash space into weighted partitions corresponding to the outbound paths. Then, packets are hashed based on their header information and forwarded on the corresponding path. In contrast, a *flow-cache* is a forwarding table that keeps track of which path each active flow traverses. The advantage of flow caching over hashing is that when new flows arrive, they can be placed on any path, which may lead to better control of dynamic splitting percentages. The downside with flow-caches is that on high-speed links, with tens of thousands of concurrent flows, they can consume a significant amount of additional memory in the router.

At the optical packet-switching level, the information about a flow is considered data. Therefore, to use multiple paths, this layer requires *multiplexing traffic at packet-level granularity* (HUANG; MARTEL; MUKHERJEE, 2011). Intuitively, this type of traffic multiplexing is better since it provides a finner-grained control over splitting ratios. However, addressing the differential delay requires installing buffers for reordering packets, which, in turn, may have a higher cost than the previous approach. This technique works as follows. After reaching a destination, but before being processed by higher level devices – i.e., network layer routers – packets are stored in a special buffer in their relative order. When the buffer has enough ordered packets, or when a predefined timer expires, they are released to higher levels.

The use of reordering buffers suffers from two main drawbacks: first, propagation delay is skewed to the highest value because packets traversing faster paths are kept waiting in the buffer for packets traversing slower ones; second, it restricts the number of possible paths to select when multiplexing a traffic since, in order to maintain performance and keep the size of buffers low, the differential delay has to be bounded to a maximum predefined value.

2.3.2 Routing overhead

Multipath routing has other pragmatic effects on the performance of the network. Generally, each path consumes extra entries in the routing tables – for example, the flow caching technique presented in the previous subsection. Further, several techniques even require multiple tables to coexist. All of these methods can lead to high memory consumption on routers depending on the load (CHO; ELHOURANI; RAMASUBRAMANIAN, 2012; KINI et al., 2010).

In addition to extra memory consumption, each path requires periodic software maintenance to check its availability, spare bandwidth, and other quality parameters. The amount of resources consumed by these management tasks grows proportionally with the number of paths (HARTMAN et al., 2012). Hence, management-related overhead tends to be high if the number of paths grows uncontrollably.

Therefore, a desirable feature is to have control over the number of paths utilized by multipath routing techniques. Reducing the number of paths saves resources and makes network management more efficient, both for automatic software and network operators (HARTMAN et al., 2012). More specifically to virtual network embedding algorithms, any solution for assigning virtual links to multiple paths should be upper bounded, thus allowing better control over the outcome of the embedding phase.

3 OPPORTUNISTIC RESILIENCE EMBEDDING

This chapter presents the design of ORE (Opportunistic Resilience Embedding). The goal of ORE is to offer an embedding algorithm with joint efficient resource utilization and opportunistic protection to virtual links against substrate network disruptions. The key idea in ORE is to use the substrate diversity and spare capacity to proactively and reactively protect against these disruptions. Toward this end, ORE defines two complementary strategies, one **proactive** (Section 3.3, Resilient Virtual Network Embedding) and the other **reactive** (Section 3.4, Opportunistic Capacity Recovery).

The organization of this chapter is given as follows. First, Section 3.1 presents a general description of ORE, defining its composing elements. Then, Section 3.2 defines fundamental aspects used in ORE. Finally, the proactive and reactive strategies are designed in Sections 3.3 and 3.4, respectively.

3.1 ORE Overview

As discussed in Chapter 2 (Section 2.2), current proposals are aimed at either protecting or reacting to substrate disruptions. Their strategies are to protect nodes or links using backup resources; or to react by reallocating the affected virtual resources. In contrast, ORE intelligently utilizes the substrate network diversity at the embedding phase, and explores its spare capacity at disruption events, as follows.

3.1.1 Joint efficient embedding and opportunistic protection

Efficient embedding

As mentioned earlier, in a network virtualization environment, virtual network requests are received on-demand. This implies that, on the long-run, the allocation and deallocation of resources may lead to fragmentation. Such fragmentation, in turn, can cause a set of physical links to become saturated. When the VNE algorithm does not consider such scenarios, virtual network requests can be rejected even if the substrate has enough aggregated available capacity. In an attempt to provide better utilization of substrate network resources, a common optimization objective in VNE algorithms is to perform load balance. This technique tends to avoid the creation of bottlenecks, hence increasing the VNR acceptance rate (CHOWDHURY; RAHMAN; BOUTABA, 2012).

Figure 3.1 illustrates the formation of a bottleneck due to saturation-unaware resource allocation algorithms. The figure shows a substrate network embedding several virtual networks. In this example, most of the capacity of virtual links is being allocated to a pair of substrate links. When substrate links become saturated, the substrate nodes utilizing

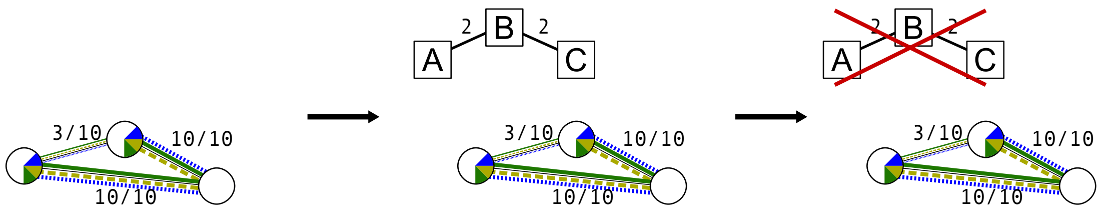


Figure 3.1: Rejection of a VNR due to the formation of a bottleneck.

these links become unreachable. As a result, new virtual network requests are rejected despite the fact that the substrate network still has enough aggregated capacity.

Mitigating disruptions

Opportunistic protection is provided by splitting the capacity of a virtual link into multiple paths. During the VNE phase, each virtual link is embedded into multiple, distinct substrate paths, and its bandwidth demand is distributed among the capacity of these paths. Hence, if a disruption occurs, only a subset of the paths of each virtual link should be affected. If the strategy is successful virtual links will remain operational – at a lower capacity – after disruptions.

The effectiveness of this strategy depends on the paths chosen in the embedding phase. More specifically, unless the selected paths are sufficiently different, a virtual link will still be vulnerable when a single substrate link embeds most – or all – of the virtual link capacity. Figure 3.2 shows an example of a virtual link with its capacity split into two paths. This mapping is valid, but inefficient in terms of resilience. When a disruption occurs on a shared substrate link, the virtual link becomes unresponsive. Therefore, the embedding phase attempts to avoid path similarity, whenever possible, when selecting substrate paths to embed virtual links.

3.1.2 Opportunistically reacting to disruptions

The objective of the second strategy is to rely on the spare capacity of the substrate network to try recovering the compromised capacity of a virtual link. After a disruption event occurs, a subset of the virtual links will be either partially of fully compromised. A virtual link is partially compromised when some of its embedding substrate paths remain operational; these paths are called *active paths*. Hence, if a virtual link has active paths, the strategy tries to increase the reserved bandwidth of these paths to compensate the lost capacity.

An example of the reactive strategy is shown in Figure 3.3. Two virtual networks, each with two virtual nodes and one virtual link, are embedded on the same substrate

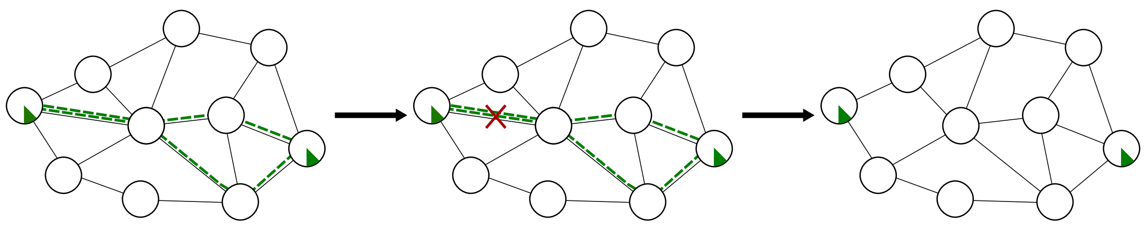


Figure 3.2: Inefficient resilient multipath embedding.

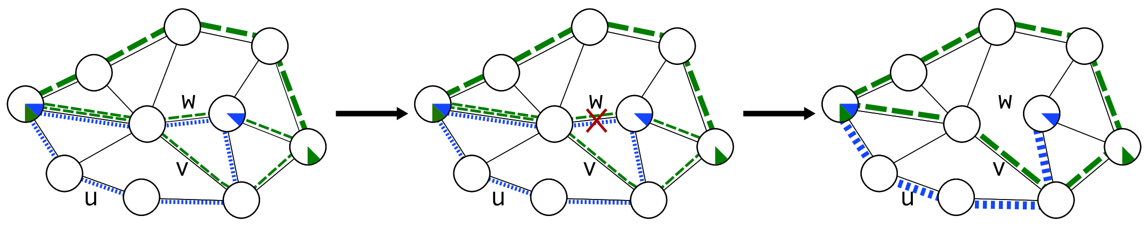


Figure 3.3: ORE approach for surviving substrate network disruptions.

network. The virtual link of the first network (green, slashed lines) is embedded on three substrate paths; similarity, the virtual link of the second network (blue, dotted lines) is embedded on two substrate paths. In the example, when an embedding substrate link "w" becomes unresponsive, the capacity of any available active path on both virtual networks is increased (e.g., substrate paths using links "u" and "v").

3.2 Preliminaries

This section describes the fundamental aspects supporting ORE. First, it defines a formal statement of the Virtual Network Embedding problem. Next, it presents the requirements and features considered by the approach. Then, it discusses how the multipath issues are addressed. Finally, it introduces the load balancing mechanism used in the embedding phase. The symbols defined in this section are also utilized in the next section; for ease of reference, they are listed in Table 3.1 (page 48).

3.2.1 Formal VNE problem statement

From an abstract standpoint, a virtual network request or a substrate network can be represented as a weighted graph with a list of attributes. Nodes, edges, and the connections between them correspond to the network topology, while the list of attributes represents virtual network requirements or substrate network features. Further, the previous discussion about VNE (Section 2.1.2) stated that the problem consists of mapping elements of a VNet graph into elements of a substrate graph, such that attributes are met and criteria are satisfied. Virtual nodes and links should be mapped to the physical substrate, essentially becoming a subset of the underlying topology. In addition, VNet requirements and substrate features should be matched; such operations may be trivial or nontrivial.

Although correct, this description lacks the precision of a formal definition. Hence, to support the ORE design, this subsection provides a formal statement of the Virtual Network Embedding problem. Three definitions are devised to compose this statement. First, the [virtual/substrate] network model is presented. Then, since matching attributes may yield nontrivial operations, a generic operator that *represents* these matchings is provided. Finally, supported by these definitions, the problem statement and criteria are formalized.

Definition 3.1 (Network Model)**.** Tuple $G\ <N,E,P,A>$ denotes a weighted, directed or undirected graph with a set N of nodes, a set E of edges (links), a set P of all possible continuous, loop-free paths, and a set A of node, edge, or path attributes. Furthermore, $\texttt{att}(x,A) \mid x \in N \cup E \cup P$ denotes the list of all attributes belonging to a particular node, edge or path.

□

Definition 3.2 (Matching Operator)**.** Let $\lhd$ denote a matching operator between attributes of a virtual network request and attributes of a substrate network. The left side of the operator indicates *requirements*, while the right side of the operator indicates *features*. Therefore, $R \lhd F$ yields **true** *iff* the features F can satisfy the requirements R, otherwise, it yields **false**. This operator has the following properties:

P1) Overloading: operations may be of different natures, some may be trivial – e.g., equality ($=$) of subset containment ($\subseteq$) – while others may be more complex – e.g., evaluating a disjoint mapping requirement; hence, $\lhd$ may operate and perform differently depending on which attributes are being matched;

P2) Recursion: matching a list of attributes "A" to another list of attributes "B" generates a list of matching operations, one for each requirement (on the left side of the operator) that must be satisfied;

P3) Aggregation: a single requirement may demand more than one feature – e.g., satisfying the bandwidth consumption of a virtual link requires verifying the bandwidth available on each individual physical link that composes its embedding; accordingly, features may be aggregated before matching.

□

Definition 3.3 (Virtual Network Embedding Problem)**.** Given a substrate network topology, henceforth denoted G^S; also, given a virtual network request, describing a virtual network topology, henceforth denoted G^V. Find a subgraph of G^S which maps each element of G^V and satisfies the following criteria:

C1) Let $\mathcal{M_N} : N^V \mapsto N^S$ denote the mapping of virtual nodes of set N^V into substrate nodes of set N^S; then, each node of $N^V \in G^V$ should only be mapped to a node in $N^S \in G^S$ which can provide its required attributes:

$$\begin{gathered} \mathcal{M_N} : n^V \to n^S \Rightarrow \texttt{att}(n^V, A^V) \lhd \texttt{att}(n^S, A^S), \\ \forall n^V \in N^V, \forall n^S \in N^S. \end{gathered} \tag{3.1}$$

C2) Let $\mathcal{M_E} : E^V \mapsto P^S$ denote the mapping of virtual links (edges) of set E^V into substrate paths of set P^S; then, each link of $L^V \in G^V$ should only be mapped to a path in $P^S \in G^S$ which can provide its required attributes and connect its virtual node endpoints:

$$\begin{gathered} \mathcal{M_E} : e^V \to p^S \Rightarrow \\ \texttt{att}(e^V, A^V) \lhd \texttt{att}(p^S, A^S) \ \wedge \ \exists \mathcal{M_N} : u^V \to x^S, \ \exists \mathcal{M_N} : w^V \to y^S \\ \mid (u^V, w^V) = e^V, \ \{(x^S, n^S), \ \ldots, (m^S, y^S)\} = p^S, \\ \forall e^V \in E^V, \forall p^S \in P^S. \end{gathered} \tag{3.2}$$

□

3.2.2 VNet requirements and substrate features

Ideally, VNE problems should consider multiple requirements and features to provide a more realistic scenario. However, the problem tends to become harder if a large number of attributes is being considered. Therefore, related work has restricted the number of attributes to a subset of interest. ORE uses the same method to allow the modeling and analysis of the problem.

In particular, the VNR modeled by ORE considers a CPU capacity on nodes and a bandwidth capacity on links. Moreover, VNR also state two constraints: i) distinct node mapping, that is, virtual nodes of the same virtual network should be embedded on distinct physical nodes; and ii) maximum number of paths per virtual link. The model for these requirements, as well as their related operations, are formalized as follows.

Definition 3.4 (Requirements and Features)**.** Given a virtual network request G^V and its set of attributes A^V. Then, function $\texttt{att}(., A^V)$ applied to each $n^V \in N^V$ retrieves two elements: a CPU demand and a distinct node mapping constraint. Further, function $\texttt{att}(., A^V)$ applied to each $e^V \in E^V$ retrieves two elements: a bandwidth demand and a maximum number of paths.

Similarly, given a substrate network G^S and its set of attributes A^S. Then, function $\texttt{att}(., A^S)$, when applied to each $n^S \in N^S$ retrieves the currently available CPU capacity of the node; and when applied to each $e^S \in E^S$ retrieves the currently available bandwidth capacity of the link.

□

Definition 3.5 (ORE Matching Operations)**.** Let $c(.)$ denote a node-only function that yields a *CPU demand*, when applied to a virtual node n^V; or yields a *currently available CPU capacity*, when applied to a substrate node n^S. Also, let $b(.)$ denote a link-only function that retrieves a *bandwidth demand*, or *currently available bandwidth capacity*, when respectively applied to virtual links (e^V) or substrate links (e^S). Moreover, let $|P|$ denote the maximum number of paths that should be used to embed a virtual link. Then, when the matching operator ($\lhd$) is applied to the attributes of the virtual and substrate networks: $\texttt{att}(x^V, A^V) \lhd \texttt{att}(x^S, A^S) \mid x^V \in N^V \cup E^V, x^S \in N^S \cup E^S$, the following set of matching operations must satisfy.

O1) Virtual nodes can only be mapped to physical nodes with enough CPU capacity available:

$$\text{Node capacity check} = \begin{cases} \textbf{true}, & \begin{array}{c} \mathcal{M}_{\mathcal{N}} : n^V \to n^S \\ \Rightarrow \\ c(n^V) \leq c(n^S); \end{array} \\ \textbf{false}, & \text{otherwise.} \end{cases} \tag{3.3}$$

O2) Virtual nodes belonging to the same virtual network should be mapped on different physical nodes; that is, virtual nodes (e.g., u^V and v^V) can only be mapped to the same physical node (e.g., n^S), if they belong to different virtual networks (i.e., $u^V \in N_i^V \in G_i^V, v^V \in N_j^V \in G_j^V \mid G_i^V \neq G_j^V$):

$$\text{Node distinct mapping} = \begin{cases} \textbf{true}, & \begin{array}{c} \mathcal{M}_{\mathcal{N}} : u^V \to n^S, \mathcal{M}_{\mathcal{N}} : v^V \to n^S \\ \Rightarrow \\ u^V \in N_i^V \in G_i^V, v^V \in N_j^V \in G_j^V \\ \mid G_i^V \neq G_j^V; \end{array} \\ \textbf{false}, & \text{otherwise.} \end{cases} \tag{3.4}$$

O3) On single path embedding, virtual links can only be mapped on a substrate path that can embed all of their demanded bandwidth; more specifically, all substrate links composing the selected path (i.e., $\forall e^S \in p^S \mid \mathcal{M}_{\mathcal{E}} : e^V \rightarrow p^S$) need to have enough available bandwidth to supply the demanded capacity (i.e., $b(e^V) \leq \min\limits_{\forall e^S \in p^S} \big(b(e^S)\big)$):

$$\text{Link capacity check (single path)} = \begin{cases} \textbf{true}, & \mathcal{M}_{\mathcal{E}} : e^V \rightarrow p^S \\ & \Rightarrow \\ & b(e^V) \leq \min\limits_{\forall e^S \in p^S} \big(b(e^S)\big)\,; \\ \textbf{false}, & \text{otherwise.} \end{cases} \tag{3.5}$$

O4) Let $\mathcal{M}_{\mathcal{E},i} : E^V \mapsto P^S \mid i \in |P|$ denote the mapping of virtual links (edges) of set E^V into substrate paths of set P^S, such that each virtual link has $|P|$ instances of mapping $\mathcal{M}_{\mathcal{E},i}$; and also, let p_i^S denote the i^{th} path to embed a particular virtual link; then, on multipath embedding, a virtual link can only be mapped on a set of substrate paths, if the aggregated capacity of these paths is enough to embed all the bandwidth demanded by the link:

$$\text{Link capacity check (multipath)} = \begin{cases} \textbf{true}, & \mathcal{M}_{\mathcal{E},i} : e^V \rightarrow p_i^S,\ \forall i \in |P| \\ & \Rightarrow \\ & b(e^V) \leq \sum\limits_{i \in |P|} \min\limits_{\forall e^S \in p_i^S} \big(b(e^S)\big)\,; \\ \textbf{false}, & \text{otherwise.} \end{cases} \tag{3.6}$$

□

3.2.3 Addressing multipath embedding concerns

As discussed in Chapter 2 (Section 2.3), the VNE algorithm has to deal with two primary concerns when splitting the capacity of a virtual link over multiple paths: out-of-order delivery, caused by the differential delay; and routing overhead, caused by replicated state in routing tables and maintenance tasks.

ORE tackles the *out-of-order delivery* issue by multiplexing network traffic at flow-level granularity. In non-virtual, IP networks, this is achieved by assigning each flow to a specific path based on its “flow-id” (namely the src/dst IP addresses, the transport protocol, and the src/dst ports) (HE; REXFORD, 2008). Similarly, to utilize such technique, a virtual network environment would need to be able to identify from which flow belongs each packet. This should be straightforward, as non-IP virtual networks would only be required to inform which bits in the packet header should be used to identify a flow (YU et al., 2008).

Limiting routing overhead requires that the amount of paths is kept as low as possible, preferably below an upper bound (HARTMAN et al., 2012). Thus, ORE keeps track on the number of paths used to embed each virtual link. This allows the substrate network to control how much resources are being used on each virtual link, and enables virtual networks to specify how many paths their virtual links can be split.

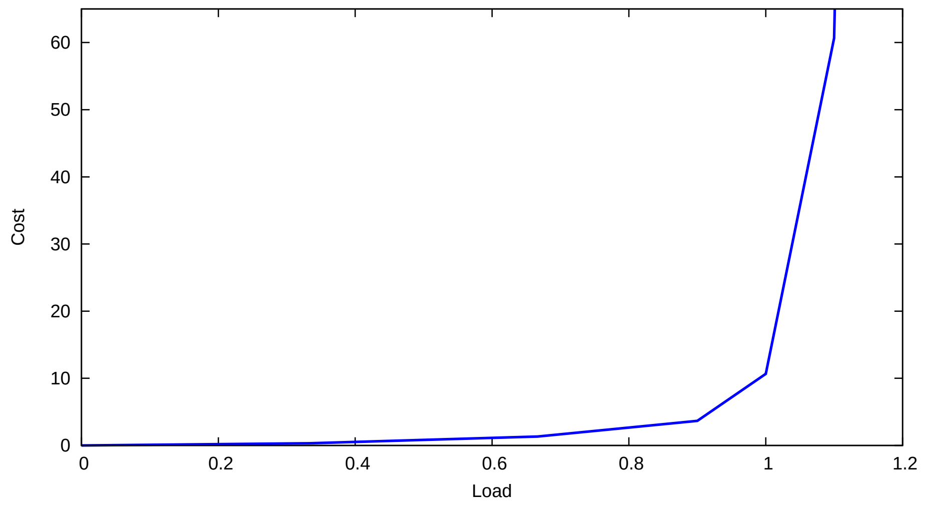


Figure 3.4: Link cost as a function of the load for a link capacity, extracted from (FORTZ; THORUP, 2000). The exponential cost is used to measure the relative importance of a link during saturation.

3.2.4 Load balancing

To provide load balancing, ORE utilizes a classic Traffic Engineering (TE) function available in literature (WANG et al., 2008; FORTZ; THORUP, 2000, 2004). This function defines an exponentially growing cost which is proportional to the substrate link capacity and utilization. This function is used with the same values defined in literature, as illustrated in Figure 3.4 (FORTZ; THORUP, 2000, 2004).

ORE uses this function as metric when performing the VNE process. Particularly, ORE gives preference to lower cost substrate links when composing the paths that embed virtual links. This way, substrate links with high utilization are less likely of being selected. The following statement defines the notation for the cost function.

Definition 3.6 (Link Utilization Cost)**.** Let $\Phi(.)$ denote an exponentially growing cost function that yields the relative importance of a link based on its utilization, as first defined in (FORTZ; THORUP, 2000). Accordingly, $\Phi(\texttt{util}(e^S))$, henceforth denoted ϕ_{e^S}, yields the current cost of a given substrate link $e^S \in E^S$.

$\square$

Table 3.1: Symbols used in Sections 3.2 and 3.3.

Symbol	Description
Sets, Variables, and Constants	
$G < N, E, P, A >$	Weighted, directed or undirected graph representing a physical or a virtual network.
G^S, G^V	Physical (Substrate) and Virtual networks, respectively.
N^S, N^V	Set of physical or virtual nodes.
E^S, E^V	Set of physical or virtual edges (links).
P^S	Set of physical continuous, loop-free paths.
$\|P\|$	Maximum number of paths used to embed a virtual link.
A^S, A^V	Set of physical or virtual attributes (virtual network requirements or substrate network features).
κ	Aggregated cost of all substrate links when saturated (100% utilization).
π_{n^S}	Probability of choosing a physical node.
π_{e^S}	Probability of choosing a physical link.
$x_{e^V,p^S}\%$	Fraction of bandwidth demand of a virtual link e^V, allocated in a given path p^S.
Operators and Functions	
$\lhd$	Matching operator; the operation $R \lhd F$ means that features F must satisfy requirements R.
$\mathcal{M}_{\mathcal{N}} : N^V \mapsto N^S$	Function mapping nodes of set N^V into nodes of set N^S.
$\mathcal{M}_{\mathcal{E}} : E^V \mapsto P^S$	Function mapping edges of set E^V into paths of set P^S (applied to single path mapping).
$\mathcal{M}_{\mathcal{E},i} : E^V \mapsto P^S$	Function mapping edges of set E^V into multiple paths of set P^S; paths are indexed by $i \in \|P\|$
$\texttt{att}(x, A)$	List of all attributes of an element x, where $x \in N \cup E \cup P$.
$c(.)$	Required CPU demand of a virtual node $\big(c(n^V) \mid n^V \in N^V\big)$, or currently available capacity of a physical node $\big(c(n^S) \mid n^S \in N^S\big)$
$b(.)$	Required bandwidth demand of a virtual link $\big(b(e^V) \mid e^V \in E^V\big)$, or currently available capacity of a physical link $\big(b(e^S) \mid e^S \in E^S\big)$
$\texttt{util}(.)$	Fraction (within [0,1]) of utilization of a given substrate resource
$\Phi(.)$	Link saturation measurement function: $\Phi(\texttt{util}(e^S))$ returns a cost which is exponentially proportional to link utilization.
$\texttt{cost}(.)$	Cost function of Simulated Annealing: $\texttt{cost}(S)$ evaluates the total link saturation of the substrate network of a potential solution "S" (passed as parameter).
$\texttt{initial\_solution}()$	Solution generation function of Simulated Annealing; creates an initial solution "S" using a greedy algorithm.
$\texttt{neighbor}(.)$	Neighbor function of Simulated Annealing: $\texttt{neighbor}(S)$ creates a solution "S'" which is similar to a previously generated solution "S" (passed as parameter)
$\texttt{alloc}(.)$	Amount of resources allocated in a substrate resource
$\texttt{resid}(.)$	Amount of available resources of a substrate resource
Mathematical Symbols	
$\mapsto, \rightarrow$	Maps to: represents mapping between sets and between elements, respectively.
$\Rightarrow$	Implication: $A \Rightarrow B$ means, if A is true, then B has to be true.
iff, $\iff$	If, and only if; double implication: $A \iff B$ means that A is only true if B is true, and vice-versa.

3.3 Resilient Virtual Network Embedding

This strategy attempts to embed virtual network requests on a given substrate network with respect to the constraints stated in Section 3.2 (Def. 3.3 and 3.5). The goal is to provide resilience for virtual networks and efficient allocation of substrate resources. Since the VNE phase is a $\mathcal{NP}$-Hard problem (ANDERSEN, 2002), an optimal model cannot scale to large inputs. To solve this problem efficiently, ORE developments a Simulated Annealing-based (SA) meta-heuristic. The solutions attained by this meta-heuristic algorithm are sub-optimal, however, they are computed at practical time. A description of the SA algorithm and its related sub-processes is provided next. Table 3.1 serves as reference to the symbols used in this section.

Simulated annealing

The SA pseudo-code is described in Algorithm 3.1. It starts by creating an initial mapping which may, or may not be a valid solution. Then, instead of exploring the entire solution space for the optimal mapping, this algorithm iteratively generates possible mappings until a maximum number of steps (i.e., K) is reached. In each step (k), SA tries to improve the current solution by comparing it to several (R) similar solutions – also called neighbors (S'). In addition, SA uses the concept of temperature (T) and cooling factor ($0 < c < 1$) to escape local minima. This works by selecting slightly worse solutions with the probability being directly proportional to the temperature ($e^{-[cost(S')-cost(S)]/T}$). Since the temperature is decreased at each step (k), this action is less likely to happen at the last few steps of SA, when the solution has probably converged to a global minimum. Finally, when SA terminates, the best overall attained solution (S^*) is returned.

The parameters of SA, namely the number of steps (K), the number of neighbors per step (R), the temperature (T), and the cooling factor (c), were tunned during experimentation. Therefore, their discussion is deferred to Chapter 4. Next, the key components of Simulated Annealing are presented, that is: (i) the *cost* function that measures the quality of a solution; (ii) the *initial solution*, which is iteratively modified in an attempt to converge to the global optimum; and (iii) the *neighbor* function, used to modify a solution.

Cost function

The quality of attained solutions is compared by measuring link saturation over the network. The objective is to achieve better resource utilization by avoiding bottlenecks. This is performed by employing the traffic engineering function Φ, previously described in Section 3.2.4 (Definition 3.6). However, this function cannot be used "as-is" in this meta-heuristic, since the algorithm should differentiate valid solutions from invalid ones.

During the execution of a meta-heuristic, several solutions are obtained at random. Even though valid solutions with worse embeddings have higher costs, invalid solutions may not. For example, consider a case where an empty substrate has to embed one virtual network with a single virtual link; further, suppose two solutions are found, one which the virtual link is properly embedded on a number of paths, and another where no path has been selected to embed this virtual link. Clearly, the first solution is valid while the second is invalid. However, by measuring only link saturation, the second solution will have a lower cost – in fact, the cost will be zero (i.e., the minimum value) because no substrate link is saturated.

To solve this problem, the cost function adds a constant value κ to all invalid solutions;

Algorithm 3.1: Pseudo-code of Simulated Annealing

```
Input: $K$                      // maximum number of outer steps
Input: $R$                      // maximum number of inner steps
Input: $T$                      // temperature
Input: $c$                      // cooling factor
Output: $S^*$                   // best global solution
$S \leftarrow$ initial_solution();
$S^* \leftarrow S$;
for $k \leftarrow 1$ upto $K$ do
    for $r \leftarrow 1$ upto $R$ do
        $S' \leftarrow$ neighbor($S$);
        if cost($S'$) $<$ cost($S^*$) then
            $S^* \leftarrow S'$;
        end
        if cost($S'$) $<$ cost($S$) then
            $S \leftarrow S'$;
        else
            with probability $e^{-[\texttt{cost}(S')-\texttt{cost}(S)]/T}$, do $S \leftarrow S'$;
        end
    end
    $T \leftarrow T \cdot c$;
end
```

κ should be equal to the worst possible solution. This value is calculated by multiplying the number of substrate links (i.e., $|E^S|$) by the value of a fully saturated link [i.e., $\Phi(1)$][1]. The cost function is given by the following expression:

$$\texttt{cost} = \begin{cases} \sum\limits_{e^S \in E^S} \phi_{e^S}, & \text{if solution is } \textit{valid}; \\ \sum\limits_{e^S \in E^S} \phi_{e^S} + \kappa \quad | \quad \kappa = |E^S| \cdot \Phi(1), & \text{if solution is } \textit{invalid}. \end{cases} \tag{3.7}$$

Initial solution

A solution is obtained by a greedy algorithm composed of three steps: the node embedding phase, the link embedding phase, and the capacity assignment phase. The entire procedure is described by Algorithm 3.2. In the node embedding phase, each virtual node is randomly placed on substrate nodes with enough capacity to hold them. Since virtual nodes from the same request cannot share the same physical node, the probability of choosing a physical node (π_n) is inversely proportional to the number virtual nodes already allocated:

$$\pi_{n^S} = \frac{1}{1 + \sum\limits_{\forall n^V \in N^V} \mathcal{M}_{\mathcal{N}} : n^V \to n^S}, \tag{3.8}$$

[1]It is noteworthy that, in practice, this constant should always be at most half of the value of the maximum representation of the data type used in implementation (i.e., integer, float, double etc). If the constant exceeds this value, there exists the possibility of overflow. However, even the integer data type, which in a 32bit processor can represent numbers up to $2^{32} - 1$, is enough to hold any value of a fully saturated link [$\Phi(1) \approx 10.67$ (FORTZ; THORUP, 2004)] for networks of at least 201 million edges.

where π_{n^S} is normalized so that the sum of all probabilities adds to 1 (i.e., $\sum_{\forall n^S \in V} \pi_{n^S} = 1$).

The link embedding phase uses a depth-first search to allocate the paths of each virtual link. Yet, instead of selecting edges deterministically, it uses a probability π_e based on two metrics: (a) link utilization; and (b) path similarity. The higher the value of one of these metrics (or both), the lower the probability to select the link:

$$\pi_{e^S} = \frac{1}{1 + \texttt{alloc}(e^S)} + \frac{1}{1 + \sum\limits_{\forall i \in |P|} \mathcal{M}_{\mathcal{E},i} : e^V \rightarrow p^S \mid e^S \in p^S}. \tag{3.9}$$

In the first term of Equation (3.9), function $\texttt{alloc}(.)$ denotes how much bandwidth is already allocated, thus guiding the solution to less saturated links. The second term of the equation indicates how many paths are using the given link, hence trying to avoid selecting the same substrate link in more then one path. Similarly to π_{n^S}, this probability is normalized so that the sum of the π_{e^S}'s of all neighboring links add to 1, (i.e., $\sum_{\forall u^S \in N^S} \pi_{e^S}, \forall v^S \in N^S \mid (u^S, v^S) = e^S \in E^S$).

Algorithm 3.2: Pseudo-code of the Initial Solution procedure

```
Output: $S$                                                // solution

1  $S \leftarrow \emptyset$;
   /* Node embedding phase:                                         */
2  for $n^V \in N^V$ do
3      $V \subseteq N^S \mid n^S \in V \iff c(n^S) \geq c(n^V)$;
4      choose $n^S \in V$ with probability $\pi_{n^S}$;
5      add $n^V$ mapping to $S$;
6  end
   /* Link embedding phase:                                         */
7  for $e^V \in E^V$ do
8      for $p \in |P|$ do
9          depth-first search with probability $\pi_{e^S}$ of selecting each $e^S \in E^S$;
10     end
11     add $e^V$ mapping to $S$;
12 end
   /* Capacity assignment phase:                                    */
13 for $e^V \in E^V$ do
14     for $p \in |P|$ do
15         Proportionally assign $x_p\%$ of $b(e^V)$ to path $p$;
16     end
17     add $e^V$ capacity allocation to $S$;
18 end
```

Finally, the capacity assignment phase consists of distributing the bandwidth demand of a virtual link among its embedding paths. In this phase, each path will receive a fraction proportional to its available bandwidth, as follows:

$$x_{e^V,p^S}\%\% = \frac{\texttt{resid}(p^S)}{\sum\limits_{\forall q^S \in P^S \mid \mathcal{M}_{\mathcal{E}} : e^V \rightarrow p^S} \texttt{resid}(q^S)}, \quad \forall p^S \in P^S \mid \mathcal{M}_{\mathcal{E}} : e^V \rightarrow p^S. \tag{3.10}$$

In Equation (3.10), function `resid`(.) indicates how much spare capacity is available in a given path; this is calculated by obtaining the minimum spare capacity among the links composing the path, as stated in Definition 3.5. Notice that before actually assigning capacity, it is necessary to check if the path has enough bandwidth. Thus, the amount of bandwidth allocated to a path p is upper bounded by the following expression: $\min\big(\texttt{resid}(p),\ x_p\% \cdot b(e^V)\big)$.

Neighborhood and local improvement

SA uses neighbors in an attempt to improve the current solution or escape a local minimum. A neighbor is generated by applying simple modifications to the current solution, resulting in a similar (potentially better) solution. This procedure is composed of two phases: the first selects a random solution from the available neighborhood, while the second repeatedly improves this solution. Both phases execute the same key steps, with a minor difference at the probability of selecting a new neighbor. Algorithm 3.3 provides a pseudo-code of the entire procedure.

Algorithm 3.3: Pseudo-code of the Generate Neighbor procedure

```
   Input: $S$                                                  // current solution
   Output: $S'$                                                // neighbor

   /* Random Neighbor                                                        */
1  $S' \leftarrow S$;
2  Select a virtual node $n^V \in N^V$ with probability $1/|N^V|$;
3  Erase node mapping of selected $n^V$ in $S'$;
4  Perform Node embedding phase for selected $n^V$ in $S'$;
5  Perform Link embedding phase in $S'$ as necessary;
6  Perform Capacity assignment phase in $S'$ as necessary;
   /* Local Improvement                                                      */
7  $S'' \leftarrow S'$;
8  repeat
9      $S' \leftarrow S''$;
10     Select a virtual node $n^V \in N^V$ with probability $\pi_{n^S}$;
11     perform steps 3 to 6 of this algorithm on $S''$;
12 until cost($S'$) $\geq$ cost($S''$) ;
```

The algorithm consists of moving a randomly selected virtual node to another physical node and recalculating paths affected by this action. At the first phase of the algorithm, a virtual node n^V is selected uniformly from the set of all virtual nodes N^V. This allows a solution to escape a local minimum by generating a similar solution (this initial solution is potentially worse). Then, at the second phase, the new solution is repeatedly improved by trying to reallocate nodes with probability π_{n^S} [Equation (3.8)], thus attempting to converge to another – potentially better – local minimum.

The node/link embedding and capacity assignment phases of this procedure are performed similarly to the ones described in the previous procedure (Algorithm 3.2). The sole difference being that the node embedding phase is performed only on the selected node; and accordingly, the link embedding and capacity assignment phases are performed only on the virtual links that where broken by the reallocation of the virtual node.

3.4 Opportunistic Capacity Recovery

Whenever a physical link becomes inaccessible, one or more paths selected to embed a virtual link may disappear. If the virtual link has no more paths left, it means it was fully compromised. Otherwise, one or more paths remained operational – from this point onwards, these paths will be referred as *active paths*.

A key contribution of this dissertation is the *opportunistic recovery* mechanism. Instead of allocating backup resources, the capacity compromised by a disruption is *recovered* over the available bandwidth along the set of active paths. The full formulation of this strategy as a Linear Program (LP) model is given next. For ease of exposition, the model is repeated in Algorithm 3.4.

Linear program

In this LP model, a single variable keeps tracks of how much bandwidth are being requested, while input sets define which substrate links will be used and how much bandwidth was lost. The objective of the model is to allocate all spare capacity of the active paths. Constraints are used to avoid waisting resources.

Variable: $\mathcal{Q}_{p,e^V} \in \mathbb{R}$ within $[0,1]$ is used to represent the fraction of the virtual link $e^V \in \dot{E}^V$ capacity which will be recovered in active path p. For example, $\mathcal{Q}_{p,e^V} = 0.1$ means 10 percent of the capacity of e^V is being reassigned to active path p. Supposing e^V has a nominal capacity of 10 Mbps, that means, an additional 1 Mbps will be reserved on each substrate link along active path p.

Input:

- set $\hat{E}^S \subset E^S$ defines the substrate links that remain available after a disruption event; since compromised links are unaccessible, they cannot be used to reallocate any bandwidth;
- set $\dot{E}^V \subseteq E^V$ enumerates the virtual links affected by the disruption; all other virtual links are omitted from the model (i.e., only the virtual links with compromised capacity need to be considered);
- set $F_{e^V} \in \mathbb{R}$ within $[0,1]$ expresses the fraction of capacity of virtual link e^V which was compromised by a disruption event; and
- sets P_{e^V} and $P_{p,e^S,e^V} \in \mathbb{B}$ preserve the previous mapping of the active paths of virtual link e^V; P_{e^V} is used to enumerate the paths, while P_{p,e^S,e^V} indicates if the active path p of virtual link e^V contains physical link e^S.

Objective: *maximize the bandwidth capacity recovered after a disruption event.* Therefore, function

$$\max T = \sum_{\forall e^V \in E^V} \sum_{\forall p \in P_{e^V}} \mathcal{Q}_{p,e^V} \tag{3.11}$$

attempts to allocate all spare bandwidth available over the set of active paths. In order to avoid waisting resources, this model is subject to two constraints. The first constraint [Equation (3.12)] guarantees that the amount of bandwidth reallocated cannot be greater than the amount compromised by an attack. The second constraint [Equation (3.13)] ensures that the allocated bandwidth remains within the available capacity of physical links.

Constraints:

- the sum of all instances of variable Q_{p,e^V} of a given virtual link e^V needs to be upper bounded by F_{e^V}; consequently, the total amount of additional bandwidth requested by this virtual link $\left(\text{i.e., } \sum_{\forall p \in P_{e^V}} Q_{p,e^V}\right)$ will not surpass the amount it actually needs (F_{e^V});

$$\sum_{\forall p \in P_{e^V}} Q_{p,e^V} \leq F_{e^V}, \qquad \forall e^V \in \dot{E}^V; \tag{3.12}$$

- if a substrate link e^S is used to embed any active path of any virtual link (i.e., when $P_{p,e^S,e^V} = 1\big)$, then the sum of all multiplications of variable Q_{p,e^V} by value $b(e^V)$ is upper bounded by $b(e^S)$; accordingly, the total amount of additional bandwidth requested by all compromised virtual links $\left[\sum_{\forall e^V \in E^V} \sum_{\forall p \in P_{e^V}} P_{p,e^S,e^V} \cdot Q_{p,e^V} \cdot b(e^V)\right]$ from a single substrate link e^S will not exceed the physical capacity of this link $\big[b(e^S)\big]$.

$$\sum_{\forall e^V \in E^V} \sum_{\forall p \in P_{e^V}} P_{p,e^S,e^V} \cdot Q_{p,e^V} \cdot b(e^V) \leq b(e^S), \qquad \forall e^S \in \hat{E}^S. \tag{3.13}$$

The full linear program is described in Algorithm 3.4. Line 1 corresponds to the objective function in Equation (3.11), while lines 2 and 3 correspond respectively to the constraints in Equations (3.12) and (3.13).

Algorithm 3.4: Linear program for the Opportunistic Capacity Recovery strategy

```
Input: $\hat{E}^S \subset E^S$                  // substrate links available after disruption
Input: $\dot{E}^V \subset E^V$                 // virtual links affected by disruption
Input: $F_{e^V} \in \mathbb{R}^{[0,1]}$        // compromised capacity of a virtual link
Input: $P_{e^V}$                               // active paths of each virtual link $e^V \in \hat{E}^S$
Input: $P_{p,e^S,e^V} \in \mathbb{B}$          // physical links used to embed active paths
Output: $Q_{p,e^V}$ // fractions of capacity recovered in each active path

/* Optimize                                                              */
1 max $T = \sum_{\forall e^V \in E^V} \sum_{\forall p \in P_{e^V}} Q_{p,e^V}$;
/* Subject to                                                            */
2 s.t.: $\sum_{\forall p \in P_{e^V}} Q_{p,e^V} \leq F_{e^V}$,    $\forall e^V \in \dot{E}^V$;
3 $\sum_{\forall e^V \in E^V} \sum_{\forall p \in P_{e^V}} P_{p,e^S,e^V} \cdot Q_{p,e^V} \cdot b(e^V) \leq b(e^S)$,    $\forall e^S \in \hat{E}^S$;
```

4 EVALUATION

This chapter presents the experiments used to measure the performance of the proposed approach. Experiments where performed in a simulated environment and executed on an Intel i7 with 8 cores of 2.93 GHz and 8 GB of RAM. Their goal is to evaluate ORE's resilience to disruptions, lower cost in terms of resource allocation, and compute time.

The remaining of this chapter is organized in three sections, as follows. First, Section 4.1 describes the simulator used in the evaluation. Then, Section 4.2 provides the configuration of the workload, and the parameters used in the Simulated Annealing algorithm. Finally, Section 4.3 discusses the experiments and their results.

4.1 Simulator

The simulator is composed of a main loop performing calls to the algorithms presented in the previous chapter. Most of the code was created using the C/C++ language. The only exception is the LP model, which was created using A Mathematical Programming Language (AMPL)[1]. The main loop of the simulator is described in Algorithm 4.1.

Algorithm 4.1: Pseudo-code of the simulator

```
$rounds \leftarrow 0$;
repeat
    $Events \leftarrow$ generate_events();
    for $event \in Events$ do
        if event is VNR then
            simulated_annealing($K, R, T, c$) ;        // resilient VNE
        end
        if event is disruption then
            call cplex with Alg. 3.4 ;   // opportunistic capacity recovery
        end
    end
    $rounds = rounds + 1$;
until $rounds \leq$ MAX_ROUNDS ;
```

Two kinds of events are generated during the course of the simulation: i) virtual network requests; and ii) disruptions. The former generates a call to the *resilient virtual network embedding* algorithm, which is implemented internally as a Simulated Annealing-

[1]http://www.ampl.com

based allocation algorithm. The later generates a call to the *opportunistic capacity recovery* algorithm, which is implemented as a Linear Program. To solve the LP, the simulator makes external calls to the CPLEX 12.3[2] solver.

The time of the experiments is organized in rounds, where each round can receive one or more virtual network requests or disruptions. The round only finishes after concluding all steps of the allocation and/or recovery algorithms. Therefore, each step of simulation takes a different amount of time. More specifically, a round may terminate instantly if there are no incoming events; it can take a few milliseconds during the execution of the LP model; or several seconds during the execution of the SA algorithm (the execution time was evaluated in Section 4.3).

All simulations generate a scenario where several VNRs are either accepted or rejected. When plotting the acceptance rate as a function of the simulation rounds, the curve exhibits a very characteristic pattern[3]. Figure 4.1 shows an illustration of a simulation. Although pictorial, this example was generated with the same workload used in the rest of this chapter (workload configurations are described in the next section).

Graphs such as Figure 4.1 should be read as follows. At first $(0 \sim 500)$, the substrate has sufficient resources to embed all of the arriving virtual network requests, thus none of them are rejected. As time passes and more requests arrive, resources become scarce $(500 \sim 2500)$ which results in some requests being rejected. Finally, the allocation process stabilizes $(2500 \sim \infty)$ when the amount of resources being freed is approximately the amount demanded by new requests, and acceptance rate stays around a small interval $(50\%$ in the example).

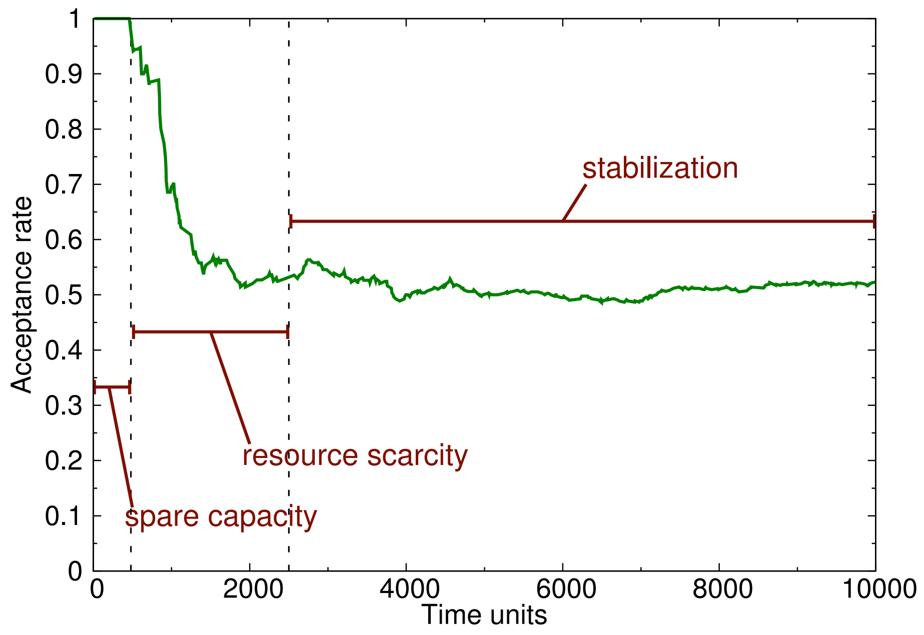


Figure 4.1: Illustration of a single simulation run.

[2]`http://goo.gl/WhvWq`

[3]This pattern has been seen numerous times in literature (ALKMIM; BATISTA; FONSECA, 2011, 2012; BAYS et al., 2012; BUTT; CHOWDHURY; BOUTABA, 2010; CHENG et al., 2012; CHOWDHURY; RAHMAN; BOUTABA, 2009, 2012)

4.2 Basic Settings

This section discusses the workload used in the experiments and the choice of parameters set to Simulated Annealing.

4.2.1 Workload

Each simulation was executed during 10000 rounds. The workload is composed of one substrate network, several virtual networks, and arrival and departure rates for VNR and disruption events. Each network is composed of nodes and links with cpu and bandwidth requirements, respectively. Event arrivals and departures follow known distributions. As network virtualization is still an open field, the choice of setting for generating the workload was based on related work (CHOWDHURY; RAHMAN; BOUTABA, 2012; CHENG et al., 2012).

The *network topologies* were created using BRITE[4] with the Barabási-Albert (BA-2) network model (ALBERT; BARABÁSI, 2000). The *substrate* network is composed of 50 nodes and 194 links. The CPU and bandwidth capacity of nodes and links are uniformly distributed within [50,100]. *Virtual networks* vary in size; the number of nodes in a virtual network topology is uniformly distributed between 2 and 10, while the number of links follows a $n - 1$ pattern defined by the BA-2 model. The capacity of virtual nodes and virtual links is uniformly distributed within [5,15] and [15,30], respectively. Similarly to previous work, units are kept abstract; they could represent Mbps on links or time slices on nodes.

There are two types of *events* in this environment: virtual network requests, and disruptions. The arrival rate of each VNR is given by a Poisson process with an average of 70 requests per 100 rounds. The duration of each request is geometrically distributed with an average of 100 rounds. Similarly, the disruptions are modeled by a Poisson process with mean of 1 disruption per 100 rounds and duration of 5 rounds. Each disruption follows the worst case scenario and removes the physical node that has the greatest bandwidth allocation on its links; this is a common assumption made to model strategic attacks (MAGNIEN; LATAPY; GUILLAUME, 2011). Since simulations executed during 10000 rounds, each experiment generated approximately 7000 requests and 100 disruptions.

4.2.2 Tunning SA parameters

As presented in the previous chapter (Section 3.3), the parameters of SA are: the maximum number of outer steps; the maximum number of inner steps (also meaning the number of neighbors per outer step); the temperature; and the cooling factor. These parameters are denoted by K, R, T and c, respectively. The main idea behind them is that by increasing the number of outer (K) and inner (R) steps, and adjusting the temperature (T) and cooling factor (c), the SA algorithm can achieve better solutions. However, better solutions come at the cost of more computation. Specifically, notice that K and R have multiplicative cost, that is, increasing K by 1 results in R additional substeps. Therefore, SA parameters should be set to an appropriate value. In this dissertation, they were first defined empirically, and then tunned until there was no sensitive improvement in the quality of the solutions.

Figure 4.2 shows the performance of the SA algorithm, measured in acceptance rate, with different samples of configurations for parameters (curves); in this plot, the higher the value in the y axis, the better the performance. Starting from the bottom-up, the lowest

[4]`http://www.cs.bu.edu/brite`

performance in SA occurs when using the number of outer steps and inner steps equal to 1 ($K = R = 1$). This is equivalent to generating a single *initial solution* (Algorithm 3.2) and applying a *local improvement* (Algorithm 3.3). In this experiment, SA performs poorly and the acceptance rate stays below 20%.

As the number of steps increases, the performance of the algorithm becomes proportionally better. After stabilizing, the second curve (bottom-up) reaches an acceptance rate around 40 and 45%, more than doubling the performance with only 100 steps ($K = 10$ and $R = 10$). Acceptance rate improves again when increasing the number of steps to 1000 ($K = 10$ and $R = 100$), reaching almost 55%. As observed in the next two curves, this relative gain in performance decreases rapidly when acceptance rate gets close to 70%. The final combination of parameters required 160,000 (160 thousand) steps to reach an acceptance rate of approximately 72%.

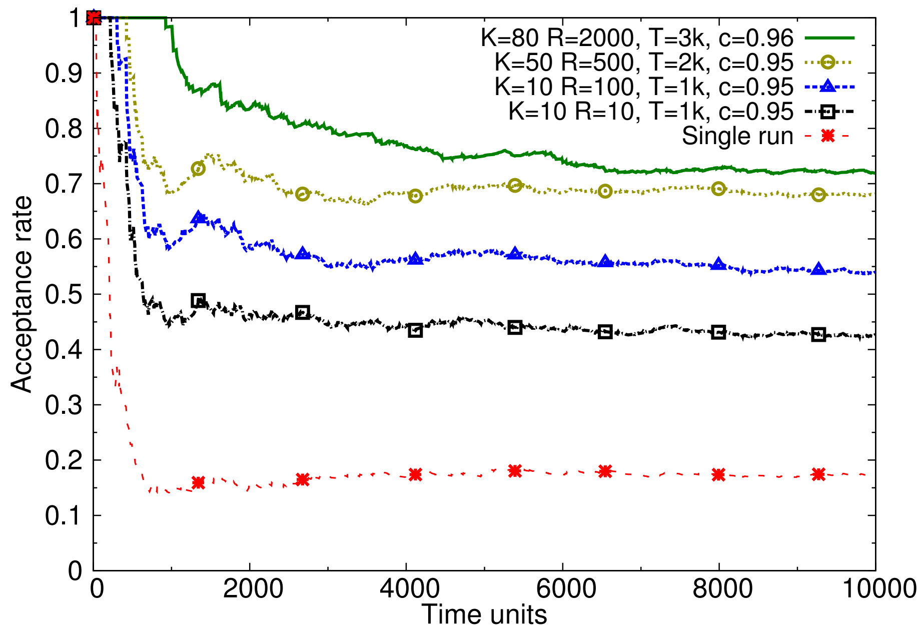


Figure 4.2: Behavior of the Simulated Annealing algorithm for different values of parameters.

Based on these findings, the set of parameters used in the rest of the experiments were the following: the number of outer steps (K) was set to 80, the number of neighbors per step (R) to 2000, the temperature (T) to 3000, and the cooling factor (c) to 0.96.

4.3 Evaluation Results

As described in the previous chapter, ORE has two main roles: i) to *prevent* against the impact of disruptions by allocating, efficiently, virtual links into multiple paths; and ii) to *react* against these disruptions, whenever possible, by reallocating compromised capacity over a set of active paths. This section evaluates the proposed approach by quantifying the impact of attacks over the affected virtual links as well as the amount of accepted requests. Further, the time needed to allocate virtual network requests is also measured. All samples were collected after the allocation process stabilized (see the explanation of Figure 4.1). Error bars represent 95% confidence interval for 20 executions.

ORE helps to mitigate disruptions by preventing bandwidth loss

Figure 4.3 shows how ORE protects virtual links against bandwidth loss using multiple paths. The plot depicts the average bandwidth loss (y axis) as a function of the number of paths to be used when embedding a virtual link (x axis); hence, the lower the value, the better it is. The baseline has only one path per virtual link. In this case, when a disruption occurs, ORE cannot protect any bandwidth, which causes the virtual links to become inoperable. Intuitively, increasing the number of paths should make the network more resilient.

As expected, the effectiveness of ORE's protection scheme is proportional to the number of paths per virtual link. Observe that, by using 2 paths per virtual link, ORE decreases bandwidth loss to approximately 61%, that is, it protects about 39%. Furthermore, by increasing the number of paths to 4 and 6, ORE can protect 65% and 79% of the bandwidth, respectively. The potential for bandwidth protection stabilizes around 85%, when using 8 and 10 paths per virtual link.

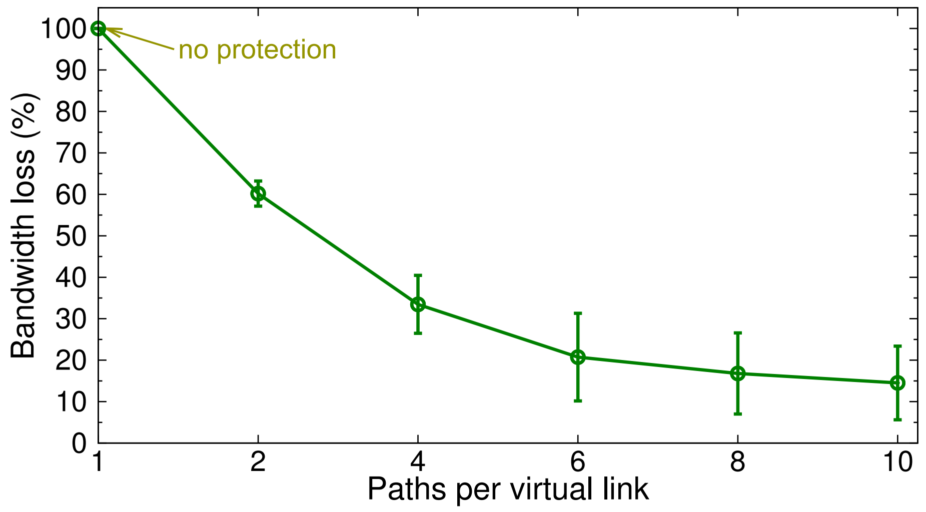


Figure 4.3: Bandwidth loss on scenarios with 1 to 10 paths per virtual link.

Figure 4.4 shows the relative gain in bandwidth protection achieved by ORE. The y axis represents protected bandwidth, while the x axis represents the number of paths per virtual link. In other words, the plot depicts how much extra bandwidth was saved by each increment in the number of paths. Although the efficiency of ORE's strategies increases with the number of paths, this behavior is nonlinear. When using 2 and 4 paths, ORE protects, respectively, 39% and 65% of the bandwidth capacity. This means a gain of approximately 26% in bandwidth protection. In comparison, this gain is reduced to about 14% when the number of paths increases to 6. Moreover, the relative gain of increasing from 6 to 8 paths is only 4%, and from 8 to 10 is negligible (around 1%). This happens because the substrate network has a limited amount of disjoint paths available to embed each virtual link. The next experiment provides further evidence of this behavior.

Multipath allocation reduces the severity of disruptions in critical links

A "critical link" is defined as a substrate link that concentrates the full capacity of a virtual link by a circumstance of the physical network and/or the allocation procedure. According to this definition, if an attack on a critical link succeeds, one or more virtual

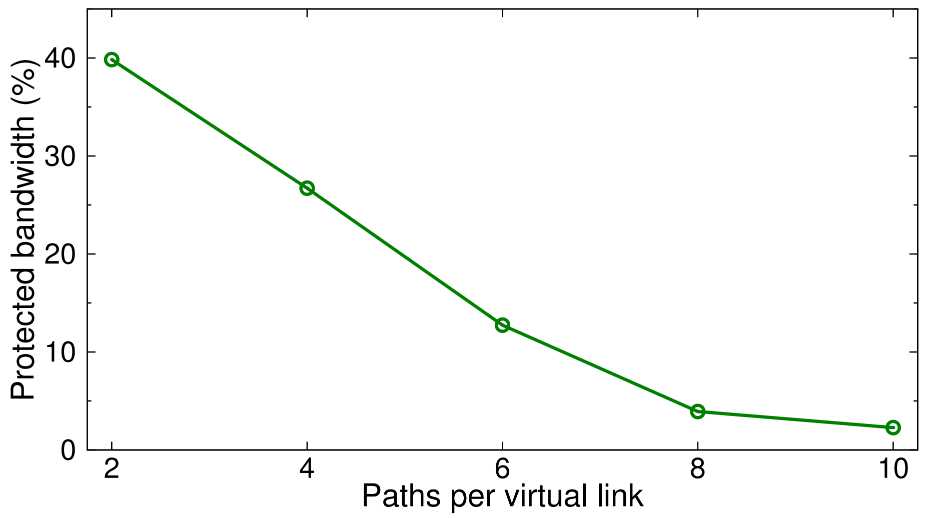


Figure 4.4: Relative gain in bandwidth protection.

links would be fully compromised. More specifically, this situation happens when all paths of a given virtual link share, at least, one substrate link. For example, if a virtual link uses two paths then, to be protected from this kind of attack, both paths have to be disjoint. Otherwise, a planned attack on a shared link would cause the virtual link to become unresponsive.

Figure 4.5 shows how the embedding approach used by ORE can decrease the vulnerability to such attacks. Vulnerability is measured by seeing how many virtual links have all their paths sharing a same substrate link. Axis x and y represent, respectively, the number of paths used in the embedding process and the percentage of virtual links vulnerable to this kind of attack; therefore, the lower the value, the better. As can be observed, when considering 2 paths per virtual link, approximately 34% of virtual links are vulnerable to a strategically planned attack in a critical link. By increasing the number of paths, the percentage of vulnerable virtual links decreases to 12% (using 3 paths), then 5% (using 4 paths), and finally stabilizes around $1\% \sim 2\%$ (using 6 to 10 paths). Based on these results, it is possible to infer that the protection against this kind of attack is proportional to the number of paths. This behavior can be further analyzed with a full view of path similarities.

The CDF (Cumulative Distribution Function) presented in Figure 4.6 quantifies virtual links similarity (i.e., the amount of paths that share a substrate link). While the y axis represents the fraction of virtual links embedded into the physical substrate, the x axis represents how many paths of a same virtual link are overlapping. Numbers indicate the amount of virtual links vulnerable to attacks in critical links (as recently discussed). The special case "0" is the equivalent to a disjoint embedding, that is, the paths selected to embed a given link do not share any substrate link. Notice that, as the number of paths increases, the likelihood of all of them being embedded on the same substrate link decreases. This characteristic is desirable because it increases the amount of physical resources an attacker has to compromise disconnect a virtual network.

Further analyzing Figure 4.6, it is possible to see that the fraction of virtual links with disjoint paths also decreases. The importance of disjoint paths is that if one or more paths share the same underling physical link, the efficiency of the opportunistic recovery strategy is limited by the capacity of the shared link. When virtual links are

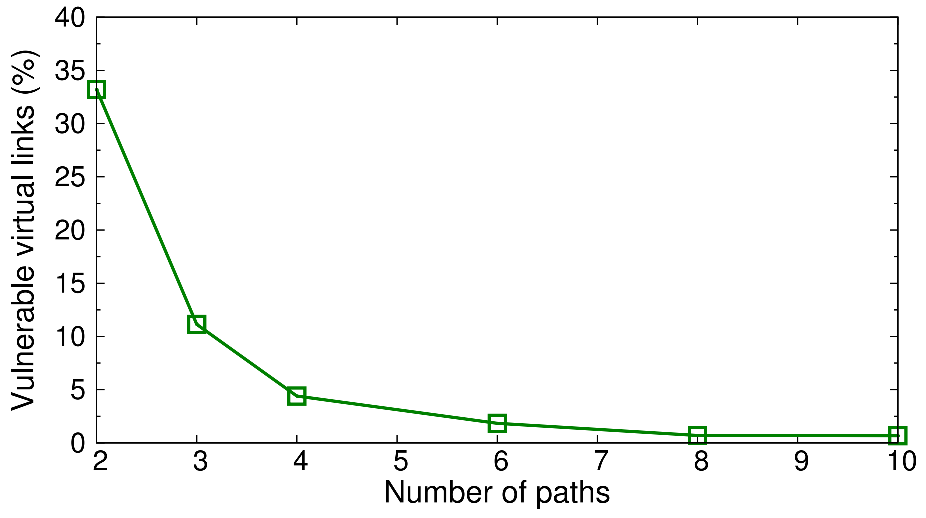


Figure 4.5: Percentage of virtual links vulnerable to attacks on critical links in a substrate network with 50 nodes and 194 links.

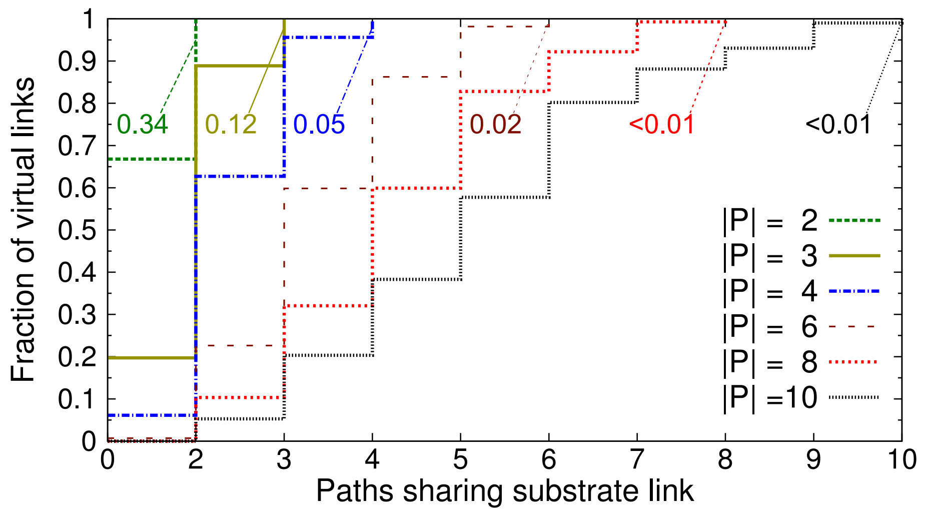


Figure 4.6: CDF displaying path similarity of virtual links in a substrate network with 50 nodes and 194 links.

embedded on 2 paths, 66% of these virtual links have disjoint paths (i.e., do not share any substrate link). This decreases to 20% when the number of paths per virtual link increases to 3, and then 6% with 4 paths per virtual link. Further, this value is reduced to less then 1% for a higher number of paths. A possible explanation for this behavior is that the amount of path diversity available in the substrate network hinders the ability of the algorithm to avoid similarity. This inference is evaluated by increasing the number of substrate links without varying the number of nodes, thus enhancing path diversity.

Figure 4.7 shows a CDF similar to the previous one for a substrate network of 50 nodes and 380 links. The number of nodes remained the same as in Figure 4.6, but the number of links almost doubled. According to the plot, the previous supposition sustains and ORE is able to reduce path similarity (i.e., the CDF is shifted towards the left). As can be observed, the fraction of virtual links embedded in disjoint paths is higher than the previous scenario; for example, with two paths per virtual link, this number increases from 0.66 (66%) to 0.84 (84%). Similarly, the fraction of virtual links in which all paths share a single substrate link is reduced. Consequently, as can be observed in Figure 4.8, the percentage of virtual links vulnerable to attacks in critical links is reduced significantly, reaching less than 1% (approximately 0.5%) when using 4 paths per virtual link (comparatively, in the previous allocation this number was more than 10 times as high).

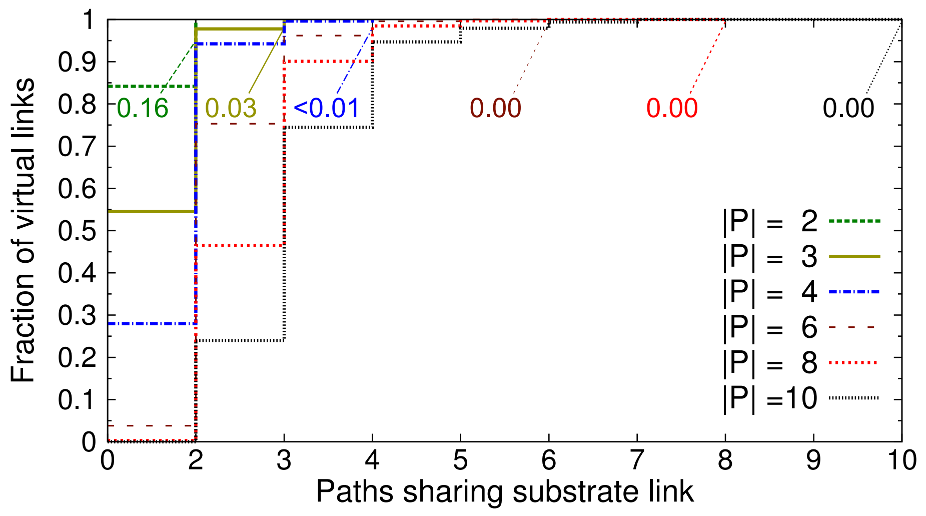


Figure 4.7: CDF displaying path similarity of virtual links in a substrate network with 50 nodes and 380 links.

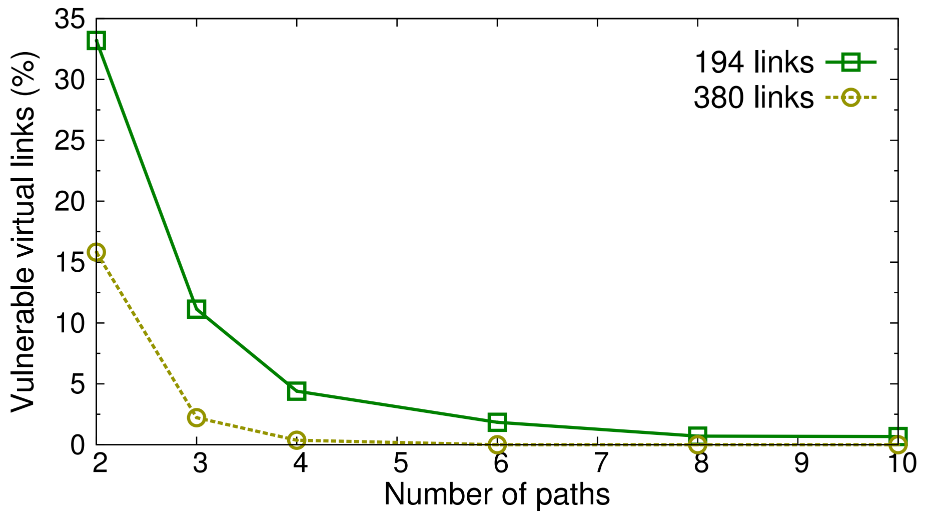


Figure 4.8: Comparison of the percentage of virtual links vulnerable to attacks on critical links, in scenarios with 194 and 380 substrate links.

The use of backup resources reduces long-term acceptance rate

Figure 4.9 illustrates and compares ORE's acceptance rate against backup-oriented schemes (denoted Bkp-X%). Intuitively, a higher number of accepted requests should increase the InP's revenue; hence, a higher value means a better result. SVNE was chosen among backup-oriented schemes (RAHMAN; AIB; BOUTABA, 2010) as it is the only approach that distributes backup resources evenly through the substrate network, that is, the allocated backup capacity is proportional to the nominal capacity of the physical link. SVNE sets a fraction of every physical link as backup (say 20%). Then, an offline phase pre-calculates paths (detours) for every physical link. Whenever a disruption on a physical link occurs, the capacity embedded in this link is reallocated in the backup resources of the detours[5].

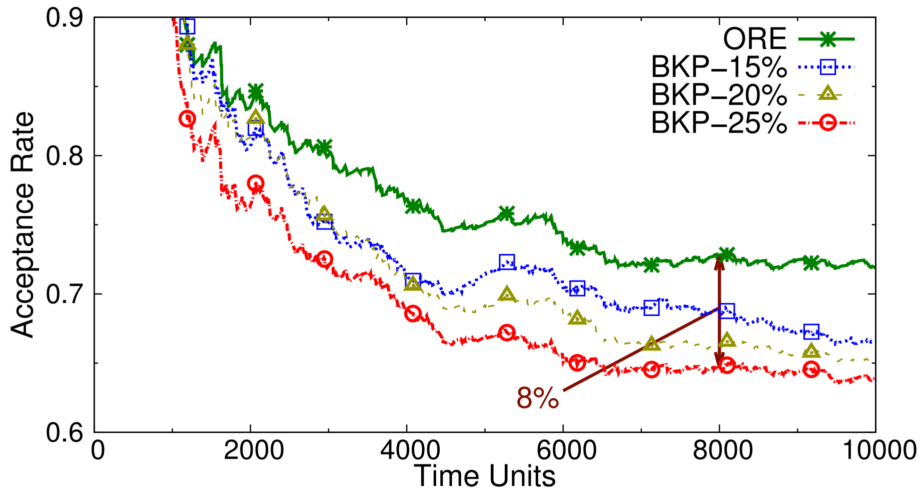


Figure 4.9: Acceptance rate for the ORE solution and backup-oriented schemes.

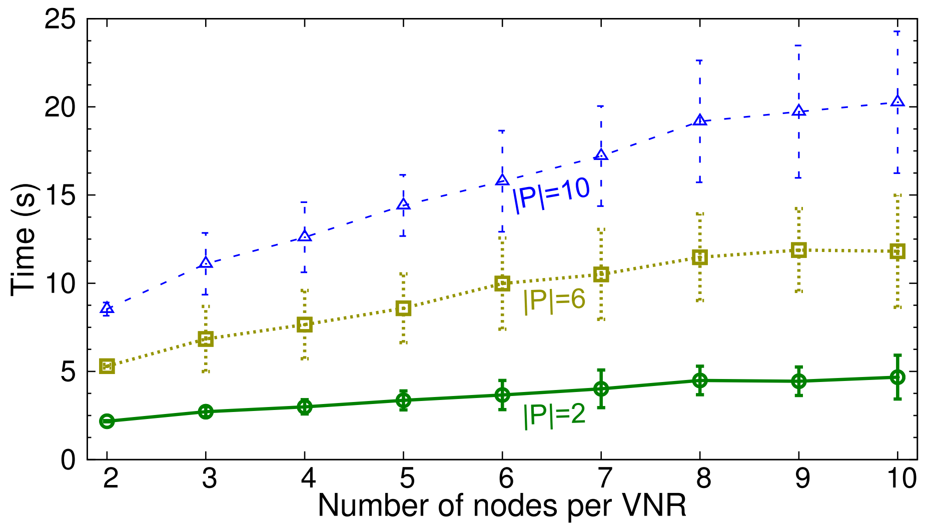
ORE's embedding algorithm was configured to use two paths per virtual link, while backup-oriented schemes used single-path embedding. Varying the number of paths caused negligible difference in acceptance rate when compared to the penalty incurred by using backups. Further, three settings were used for backup resources: 15%, 20%, and 25%. These values were chosen since the fraction of backup resources varies between 17% and 22% in previous work (RAHMAN; AIB; BOUTABA, 2010; GUO et al., 2011). As expected, the use of backup resources results in lower long-term acceptance rate. This difference tends to grow larger as more resources are used for backup purposes (e.g., 8% lower when using 25% of resources as backup).

ORE's computing time is linear with respect to the number of paths per virtual link and the size of the VNR

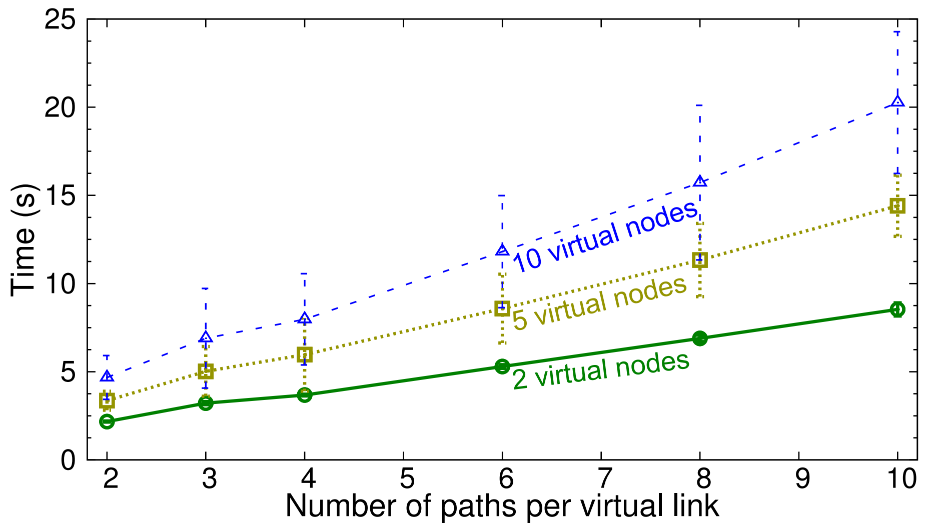
The two strategies composing ORE behave differently in respect to computing time. While the reactive strategy executed under 150 ms (0.15 s) in all experiments, the preventive strategy required much more processing time. This is because the preventive strategy requires solving the VNE problem, which is $\mathcal{NP}$-Hard (ANDERSEN, 2002). Therefore, the next experiment shows the time it takes for Simulated Annealing to perform the embedding process, demonstrating the feasibility of the proposed approach. Figure 4.10 depicts time as a function of (a) the size of virtual network requests; and (b) the number

[5]The scheme was described more deeply in Chapter 2.1 (Section 2.2.1).

of paths per virtual link. Since both parameters affect the time to compute a solution, one is kept constant in each plot. The size of VNRs is represented by the number of nodes in each request.



(a)



(b)

Figure 4.10: Mean time to compute allocation.

As observed in Figure 4.10, time is linearly proportional to the size of both inputs. Furthermore, it is possible to see that, in the selected scenario, ORE can compute a single instance of the embedding process in under a minute. As inputs grow in size, ORE is expected to degrade gracefully.

5 CONCLUSION

Network virtualization has been proposed as a means to overcome Internet Ossification. In essence, it aims to achieve this goal by allowing multiple virtual networks to coexist in the same physical substrate. Furthermore, one of its potential advantages is the use of isolation to withstand interferences among virtual networks. Nonetheless, this dissertation argued that virtual networks are still vulnerable to disruptions in the physical substrate network.

Virtual networks are mapped to the physical one in a process called Virtual Network Embedding. When this process happens, virtual networks are likely to have less diversity. In addition, the process tends to increase dependency on certain physical resources. In consequence, a compromised substrate element may affect several virtual ones. More specifically, if a physical link is compromised, all virtual links embedded in it may became inoperable.

Summary of contributions

The problem was addressed by ORE (Opportunistic Resilience Embedding), a novel approach to protect against such disruptions. ORE, unlike previous work, does not need to set aside backup resources or reallocate virtual links. Rather, it is composed of two complementary strategies. The first protects against disruptions by embedding each virtual link into multiple paths. The second attempts to recover any compromised capacity over the set of active paths (*i.e.*, those unaffected by the disruption). The set of experiments performed in this dissertation has shown interesting findings.

Preventing bandwidth loss. ORE's main goal was to protect and react against full loss of bandwidth, thus avoiding virtual links from becoming unresponsive. As stated in the evaluation section, ORE's capacity to protect bandwidth is proportional to the number of paths it uses in the embedding phase. This gain was found to be sublinear, with a tendency to converge within a few number of paths.

Reducing the severity of disruptions in critical links. ORE's embedding approach aims to avoid path similarity whenever possible. In consequence, ORE reduces the fraction of virtual links which are fully affected by a disruption in a single link. This effect is also proportional to the number of paths, and converges rapidly to a very small value (i.e., less than 1%). Moreover, it has been shown that ORE's ability to avoid path similarity is proportional to the amount of path diversity available in the substrate network.

Mitigating disruptions at a lower cost. The use of backup resources tends to reduce acceptance rate. The difference in acceptance rate between ORE and backup-oriented schemes was shown to grow larger as the use of backup resources increases. Hence, ORE can mitigate disruptions at a higher acceptance rate.

Directions for future research

ORE has demonstrated a novel virtual network embedding algorithm which aims to provide joint efficient resource utilization and opportunistic resilience to the network virtualization environment. In this context, ORE is a first step at studying lower cost alternatives for mitigating disruptions. Although ORE is efficient and feasible, opportunities for improvement exist.

Regarding resilience, experiments showed that the approach has great potential in protecting and recovering bandwidth, as well as exploring path diversity. In this context, the embedding approach could benefit from new metrics for choosing paths, thus enhancing this characteristic. Moreover, the preventive and reactive strategies work separately. Therefore, coordinating both strategies would be a desirable addition to ORE.

In terms of acceptance rate, ORE performs better than backup-oriented schemes, yet *allocation could be improved*. One possibility is to try new metrics for the quality of the solution. In addition, there exist several types of meta-heuristic algorithms; some performing better than others depending on the problem. ORE could benefit from a study in the performance of different meta-heuristics.

Finally, *ORE's capabilities could be extended* with the incorporation of a migration process. After the reactive strategy, ORE could use migration to improve resilience. Moreover, migration may be used in a regular time basis to reduce resource fragmentation, further optimizing resource allocation.

REFERENCES

ALBERT, R.; BARABÁSI, A.-L. Topology of Evolving Networks: local events and universality. **Physical Review Letters**, College Park, MD, USA, v.85, p.5234–5237, 2000.

ALKMIM, G. P.; BATISTA, D. M.; FONSECA, N. L. S. Optimal Mapping of Virtual Networks. In: IEEE GLOBAL TELECOMMUNICATIONS CONFERENCE, 54., Houston, TX, USA. **Proceedings** IEEE Globecom, 2011. p.1–6.

ALKMIM, G. P.; BATISTA, D. M.; FONSECA, N. L. S. Approximated Algorithms for Mapping Virtual Networks on Network Substrates. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, 2012., Ottawa, Canada. **Proceedings** IEEE ICC, 2012.

ANDERSEN, D. G. **Theoretical approaches to node assignment**. [S.l.: s.n.], 2002. Unpublished manuscript. Available from: `http://goo.gl/Xl1ph`. Accessed 3rd May 2013.

ANDERSON, T.; PETERSON, L.; SHENKER, S.; TURNER, J. Overcoming the Internet impasse through virtualization. **IEEE Computer**, Los Alamos, CA, USA, v.38, n.4, p.34–41, april 2005.

BARLA, I. B.; SCHUPKE, D. A.; CARLE, G. Resilient Virtual Network Design for End-to-End Cloud Services. In: BESTAK, R.; KENCL, L.; LI, L. E.; WIDMER, J.; YIN, H. (Ed.). **NETWORKING 2012, 11th International IFIP TC 6 Networking Conference**. Berlin, Germany: Springer Berlin/Heidelberg, 2012. p.161–174. (Lecture Notes in Computer Science, v.7289).

BAYS, L. R.; OLIVEIRA, R. R.; BURIOL, L. S.; BARCELLOS, M. P.; GASPARY, L. P. Security-aware Optimal Resource Allocation for Virtual Network Embedding. In: MINI-CONFERENCE OF THE INTERNATIONAL CONFERENCE ON NETWORK AND SERVICE MANAGEMENT, 8., Las Vegas, NV, USA. **Proceedings** IEEE CNSM, 2012. p.378–384.

BELBEKKOUCHE, A.; HASAN, M.; KARMOUCH, A. Resource Discovery and Allocation in Network Virtualization. **IEEE Communications Surveys & Tutorials**, New York, NY, USA, v.14, n.4, p.1114–1128, 2012.

BODÍK, P.; MENACHE, I.; CHOWDHURY, M.; MANI, P.; MALTZ, D. A.; STOICA, I. Surviving failures in bandwidth-constrained datacenters. In: ACM SIGCOMM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION, 2012., Helsinki, Finland. **Proceedings** ACM SIGCOMM, 2012. p.431–442.

BUTT, N. F.; CHOWDHURY, M.; BOUTABA, R. Topology-Awareness and Reoptimization Mechanism for Virtual Network Embedding. In: CROVELLA, M.; FEENEY, L.; RUBENSTEIN, D.; RAGHAVAN, S. (Ed.). **NETWORKING 2010, 9th International IFIP TC 6 Networking Conference**. Berlin, Germany: Springer Berlin/Heidelberg, 2010. p.27–39. (Lecture Notes in Computer Science, v.6091).

CASADO, M.; KOPONEN, T.; SHENKER, S.; TOOTOONCHIAN, A. Fabric: a retrospective on evolving sdn. In: SIGCOMM WORKSHOP ON HOT TOPICS IN SOFTWARE DEFINED NETWORKS, 1., Helsinki, Finland. **Proceedings** ACM SIGCOMM, 2012. p.85–90. (HotSDN).

CHEN, Y.; LI, J.; WO, T.; HU, C.; LIU, W. Resilient Virtual Network Service Provision in Network Virtualization Environments. In: IEEE 16TH INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED SYSTEMS, Shanghai, China. **Proceedings** IEEE ICPADS, 2010. p.51–58.

CHENG, X.; SU, S.; ZHANG, Z.; SHUANG, K.; YANG, F.; LUO, Y.; WANG, J. Virtual network embedding through topology awareness and optimization. **Computer Networks**, Amsterdam, Netherlands, v.56, n.6, p.1797–1813, 2012.

CHENG, X.; SU, S.; ZHANG, Z.; WANG, H.; YANG, F.; LUO, Y.; WANG, J. Virtual network embedding through topology-aware node ranking. **ACM SIGCOMM Computer Communication Review**, New York, NY, USA, v.41, p.38–47, April 2011.

CHO, S.; ELHOURANI, T.; RAMASUBRAMANIAN, S. Independent directed acyclic graphs for resilient multipath routing. **IEEE/ACM Transactions on Networking**, Piscataway, NJ, USA, v.20, n.1, p.153–162, 2012.

CHOWDHURY, M.; RAHMAN, M. R.; BOUTABA, R. Virtual Network Embedding with Coordinated Node and Link Mapping. In: ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES, 28., Rio de Janeiro, RJ, Brazil. **Proceedings** IEEE INFOCOM, 2009. p.783–791.

CHOWDHURY, M.; RAHMAN, M. R.; BOUTABA, R. ViNEYard: virtual network embedding algorithms with coordinated node and link mapping. **IEEE/ACM Transactions on Networking**, Piscataway, NJ, USA, v.20, n.1, p.206–219, february 2012.

CHOWDHURY, N. M. K.; BOUTABA, R. A survey of network virtualization. **Computer Networks**, Amsterdam, Netherlands, v.54, n.5, p.862–876, 2010.

CLARK, D.; WROCLAWSKI, J.; SOLLINS, K.; BRADEN, R. Tussle in cyberspace: defining tomorrow's internet. **IEEE/ACM Transactions on Networking**, Piscataway, NJ, USA, v.13, n.3, p.462–475, june 2005.

FEAMSTER, N.; GAO, L.; REXFORD, J. How to lease the internet in your spare time. **ACM SIGCOMM Computer Communication Review**, New York, NY, USA, v.37, p.61–64, January 2007.

FISCHER, A.; BOTERO, J.; BECK, M.; DE MEER, H.; HESSELBACH, X. Virtual Network Embedding: a survey. **IEEE Communications Surveys & Tutorials**, New York, NY, USA, v.15, n.1, p.1–19, 2013.

FORTZ, B.; THORUP, M. Internet traffic engineering by optimizing OSPF weights. In: ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES, 19., Tel Aviv, Israel. **Proceedings** IEEE INFOCOM, 2000. v.2, p.519–528.

FORTZ, B.; THORUP, M. Increasing Internet Capacity Using Local Search. **Computational Optimization and Applications**, Dordrecht, Netherlands, v.29, p.13–48, 2004.

GAO, P. X.; CURTIS, A. R.; WONG, B.; KESHAV, S. It's Not Easy Being Green. In: ACM SIGCOMM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION, 2012., Helsinki, Finland. **Proceedings** ACM SIGCOMM, 2012. p.211–222.

GUDE, N.; KOPONEN, T.; PETTIT, J.; PFAFF, B.; CASADO, M.; MCKEOWN, N.; SHENKER, S. NOX: towards an operating system for networks. **SIGCOMM Computer Communication Review**, New York, NY, USA, v.38, n.3, p.105–110, july 2008.

GUO, T.; WANG, N.; MOESSNER, K.; TAFAZOLLI, R. Shared Backup Network Provision for Virtual Network Embedding. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, 2011., Kyoto, Japan. **Proceedings** IEEE ICC, 2011. p.1–5.

HARTMAN, T.; HASSIDIM, A.; KAPLAN, H.; RAZ, D.; SEGALOV, M. How to split a flow? In: ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES, 31., Orlando, FL, USA. **Proceedings** IEEE INFOCOM, 2012. p.828–836.

HE, J.; REXFORD, J. Toward internet-wide multipath routing. **IEEE Network**, New York, NY, USA, v.22, n.2, p.16–21, March-April 2008.

HEEGAARD, P. E.; TRIVEDI, K. S. Network survivability modeling. **Computer Networks**, Oxford, UK, v.53, n.8, p.1215 – 1234, 2009.

HOUIDI, I.; LOUATI, W.; AMEUR, W. B.; ZEGHLACHE, D. Virtual network provisioning across multiple substrate networks. **Computer Networks**, Amsterdam, Netherlands, v.55, n.4, p.1011–1023, 2011.

HOUIDI, I.; LOUATI, W.; ZEGHLACHE, D.; PAPADIMITRIOU, P.; MATHY, L. Adaptive virtual network provisioning. In: SIGCOMM WORKSHOP ON VIRTUALIZED INFRASTRUCTURE SYSTEMS AND ARCHITECTURES, 2., New Delhi, India. **Proceedings** ACM SIGCOMM, 2010. p.41–48. (VISA).

HUANG, S.; MARTEL, C.; MUKHERJEE, B. Survivable multipath provisioning with differential delay constraint in telecom mesh networks. **IEEE/ACM Transactions on Networking**, Piscataway, NJ, USA, v.19, n.3, p.657–669, february 2011.

KATIB, I.; MEDHI, D. Network protection design models, a heuristic, and a study for concurrent single-link per layer failures in three-layer networks. **Computer Communications**, Oxford, UK, n.0, p.–, 2012.

KELLER, E.; GHORBANI, S.; CAESAR, M.; REXFORD, J. Live migration of an entire network (and its hosts). In: ACM WORKSHOP ON HOT TOPICS IN NETWORKS, 11., Redmond, Washington. **Proceedings** ACM, 2012. p.109–114. (HotNets-XI).

KHAN, A.; ZUGENMAIER, A.; JURCA, D.; KELLERER, W. Network virtualization: a hypervisor for the internet? **IEEE Communications Magazine**, New York, NY, USA, v.50, n.1, p.136–143, 2012.

KINI, S.; RAMASUBRAMANIAN, S.; KVALBEIN, A.; HANSEN, A. Fast Recovery From Dual-Link or Single-Node Failures in IP Networks Using Tunneling. **IEEE/ACM Transactions on Networking**, Piscataway, NJ, USA, v.18, n.6, p.1988–1999, 2010.

KOPONEN, T.; CASADO, M.; GUDE, N.; STRIBLING, J.; POUTIEVSKI, L.; ZHU, M.; RAMANATHAN, R.; IWATA, Y.; INOUE, H.; HAMA, T.; SHENKER, S. Onix: a distributed control platform for large-scale production networks. In: USENIX CONFERENCE ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION, 9., Vancouver, BC, Canada. **Proceedings** USENIX OSDI, 2010. p.1–6.

KOPONEN, T.; SHENKER, S.; BALAKRISHNAN, H.; FEAMSTER, N.; GANICHEV, I.; GHODSI, A.; GODFREY, P. B.; MCKEOWN, N.; PARULKAR, G.; RAGHAVAN, B.; REXFORD, J.; ARIANFAR, S.; KUPTSOV, D. Architecting for innovation. **ACM SIGCOMM Computer Communication Review**, New York, NY, USA, v.41, n.3, p.24–36, July 2011.

MAGNIEN, C.; LATAPY, M.; GUILLAUME, J.-L. Impact of random failures and attacks on Poisson and power-law random networks. **ACM Computing Surveys**, New York, NY, USA, v.43, n.3, p.13:1–13:31, Apr. 2011.

MARQUEZAN, C. C.; NOBRE, J. C.; GRANVILLE, L. Z.; NUNZI, G.; DUDKOWSKI, D.; BRUNNER, M. Distributed reallocation scheme for virtual network resources. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, 2009., Dresden, Germany. **Proceedings** IEEE ICC, 2009. p.2309–2313.

MARQUEZAN, C.; GRANVILLE, L.; NUNZI, G.; BRUNNER, M. Distributed autonomic resource management for network virtualization. In: IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, Osaka, Japan. **Proceedings** IEEE NOMS, 2010. p.463–470.

MATTOS, D.; FERNANDES, N. C.; DUARTE, O. C. M. B. XenFlow: um sistema de processamento de fluxos robusto e eficiente para migração em redes virtuais. In: XXIX SIMPóSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUíDOS, Campo Grande, MS, Brazil. **Proceedings** SBC SBRC, 2011.

MEDHI, D. Network Restoration. In: RESENDE, M. G. C.; PARDALOS, P. M. (Ed.). **Handbook of Optimization in Telecommications**. New York, NY, USA: Springer US, 2006. p.801–836.

PAPADIMITRIOU, P.; MAENNEL, O.; GREENHALGH, A.; FELDMANN, A.; MATHY, L. Implementing Network Virtualization for a Future Internet. In: ITC SPECIALIST SEMINAR ON NETWORK VIRTUALIZATION - CONCEPT AND PERFORMANCE ASPECTS, 20., Hoi An, Viet Nam. **Proceedings** ITC, 2009.

PAUL, S.; PAN, J.; JAIN, R. Architectures for the future networks and the next generation Internet: a survey. **Computer Communications**, Amsterdam, Netherlands, v.34, n.1, p.2–42, 2011.

PISA, P.; FERNANDES, N.; CARVALHO, H.; MOREIRA, M.; CAMPISTA, M.; COSTA, L.; DUARTE, O. OpenFlow and Xen-Based Virtual Network Migration. In: **Communications**: wireless in developing countries and networks of the future. Boston, MA, USA: Springer Boston, 2010. p.170–181. (IFIP Advances in Information and Communication Technology, v.327).

RAHMAN, M. R.; AIB, I.; BOUTABA, R. Survivable Virtual Network Embedding. In: CROVELLA, M.; FEENEY, L.; RUBENSTEIN, D.; RAGHAVAN, S. (Ed.). **NETWORKING 2010, 9th International IFIP TC 6 Networking Conference**. Berlin, Germany: Springer Berlin/Heidelberg, 2010. p.40–52. (Lecture Notes in Computer Science, v.6091).

REITBLATT, M.; FOSTER, N.; REXFORD, J.; SCHLESINGER, C.; WALKER, D. Abstractions for Network Update. In: ACM SIGCOMM 2012 CONFERENCE, Helsinki, Finland. **Proceedings** ACM SIGCOMM, 2012. v.42, n.4, p.323–334.

SCHUPKE, D. Multilayer and Multidomain Resilience in Optical Networks. **Proceedings of the IEEE**, Piscataway, NJ, USA, v.100, n.5, p.1140–1148, 2012.

SRIVASTAVA, A.; ACHARYA, S.; ALICHERRY, M.; GUPTA, B.; RISBOOD, P. Differential delay aware routing for Ethernet over SONET/SDH. In: ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES, 24., Miami, FL, USA. **Proceedings** IEEE INFOCOM, 2005. v.2, p.1117–1127.

STUCKMANN, P.; ZIMMERMANN, R. European research on future Internet design. **IEEE Wireless Communications**, New York, NY, USA, v.16, n.5, p.14–22, october 2009.

TURNER, J.; TAYLOR, D. Diversifying the Internet. In: IEEE GLOBAL TELECOMMUNICATIONS CONFERENCE, 48., St. Louis, MO, USA. **Proceedings** IEEE GLOBECOM, 2005. v.2, p.755–760.

WANG, N.; HO, K.; PAVLOU, G.; HOWARTH, M. An overview of routing optimization for internet traffic engineering. **IEEE Communications Surveys & Tutorials**, New York, NY, USA, v.10, n.1, p.36–56, quarter 2008.

WANG, Y.; KELLER, E.; BISKEBORN, B.; MERWE, J. van der; REXFORD, J. Virtual routers on the move: live router migration as a network-management primitive. In: ACM SIGCOMM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION, 2008., Seattle, WA, USA. **Proceedings** ACM SIGCOMM, 2008. p.231–242.

YEOW, W.-L.; WESTPHAL, C.; KOZAT, U. C. Designing and embedding reliable virtual infrastructures. In: SIGCOMM WORKSHOP ON VIRTUALIZED INFRASTRUCTURE SYSTEMS AND ARCHITECTURES, 2., New Delhi, India. **Proceedings** ACM SIGCOMM, 2010. p.33–40. (VISA).

YU, H.; ANAND, V.; QIAO, C.; SUN, G. Cost Efficient Design of Survivable Virtual Infrastructure to Recover from Facility Node Failures. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, 2011., Kyoto, Japan. **Proceedings** IEEE ICC, 2011. p.1–6.

YU, H.; QIAO, C.; ANAND, V.; LIU, X.; DI, H.; SUN, G. Survivable Virtual Infrastructure Mapping in a Federated Computing and Networking System under Single Regional Failures. In: IEEE GLOBAL TELECOMMUNICATIONS CONFERENCE, 53., Miami, FL, USA. **Proceedings** IEEE Globecom, 2010. p.1–6.

YU, M.; YI, Y.; REXFORD, J.; CHIANG, M. Rethinking virtual network embedding: substrate support for path splitting and migration. **ACM SIGCOMM Computer Communication Review**, New York, NY, USA, v.38, p.17–29, March 2008.

ZHANG, W.; TANG, J.; WANG, C.; DE SOYSA, S. Reliable adaptive multipath provisioning with bandwidth and differential delay constraints. In: ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES, 29., San Diego, CA, USA. **Proceedings** IEEE INFOCOM, 2010. p.2178–2186.

APPENDIX A - RESUMO EXPANDIDO EM PORTUGUÊS

A.1 Introdução

Desde sua popularização, a Internet desencadeou uma grande onda de inovação. Isso pode ser observado, por exemplo, na vasta quantidade de serviços e aplicações fundamentados na Internet, bem como no pluralidade de dispositivos de rede disponíveis. Todas essas inovações possibilitaram uma expansão sem precedentes nos limites da comunicação humana (KOPONEN et al., 2011).

No entanto, o atual sucesso da Internet vem inibindo sua própria inovação. Mais especificamente, qualquer modificação no núcleo da rede requer comum acordo entre diversas partes (CHOWDHURY; BOUTABA, 2010). Dessa forma, tais modificações acabam restringindo-se a "remendos" em protocolos já existentes (TURNER; TAYLOR, 2005). Face a isso, a Virtualização de Redes vem sendo proposta como um atributo diversificador para a Internet (ANDERSON et al., 2005).

A.1.1 Contexto

A virtualização de redes consiste na instanciação de múltiplas redes virtuais sobre um mesmo substrato de rede física. Tal paradigma promove o desenvolvimento de novas arquiteturas e protocolos por meio da criação de múltiplas redes virtuais sobrepostas em um mesmo substrato físico (PAUL; PAN; JAIN, 2011; CHOWDHURY; BOUTABA, 2010). Ademais, redes virtuais instanciadas sobre uma mesma rede física podem ser isoladas mutuamente, propiciando a independência funcional entre as mesmas. Com isso, uma de suas mais promissoras vantagens é a capacidade de limitar o escopo de ataques, através da organização de uma infraestrutura em múltiplas redes virtuais, isolando o tráfego das mesmas (KHAN et al., 2012).

Em um ambiente de virtualização de redes, as responsabilidades de um Provedor de Acesso à Internet são absorvidas por duas entidades principais: i) o Provedor de Infraestrutura, responsável pelo gerenciamento dos dispositivos físicos que compõem a rede, e ii) o Provedor de Serviço, responsável pela criação de redes virtuais, visando a oferta de serviços personalizados. Essa separação permite que os InPs consigam multiplexar melhor o uso de seus recursos. Além disso, os SPs podem reduzir investimentos em infraestrutura instanciando suas redes virtuais em recursos físicos alugados de InPs (FEAMSTER; GAO; REXFORD, 2007; PAPADIMITRIOU et al., 2009).

A instanciação das redes virtuais nos substratos de rede é conhecida na literatura como "Problema do Mapeamento de Redes Virtuais". Nesse problema, redes virtuais são requisitadas sob-demanda e mapeadas sobre os dispositivos físicos da seguinte forma: cada roteador virtual é alocado em um roteador físico; e cada enlace virtual é alocado em um (ou mais) caminho(s) do substrato (CHOWDHURY; RAHMAN; BOUTABA, 2012;

CHENG et al., 2012; BUTT; CHOWDHURY; BOUTABA, 2010). Para guiar a alocação, algoritmos de mapeamento atuais buscam minimizar o uso de dispositivos físicos, de forma a evitar desperdício de recursos (FISCHER et al., 2013).

A.1.2 Motivação

De maneira geral, o mapeamento de redes virtuais, ao tentar evitar o desperdício de recursos, restringe a alocação de dispositivos à um subconjunto mínimo. Esse processo tende a: (a) aumentar a dependência de certos recursos físicos; e (b) diminuir a diversidade de caminhos para determinados pares de nós fonte-destino. Consequentemente, um atacante pode realizar negação de serviço às redes virtuais, comprometendo para isso roteadores e/ou enlaces do substrato físico subjacente. Mais especificamente, caso determinado enlace do substrato seja comprometido, um número potencialmente grande de enlaces virtuais sobrepostos (ou seja, alocados neste) serão afetados.

Para lidar com esse problema, a literatura propõe dois tipos de estratégias: as que reservam recursos adicionais do substrato como sobressalentes, protegendo contra disrupções; e as que utilizam migração em tempo real para realocar recursos virtuais comprometidos. Ambas estratégias acarretam compromissos: o uso de recursos sobressalentes tende a tornar-se custoso ao provedor de infraestrutura, enquanto a migração de recursos demanda um período de convergência e pode deixar as redes virtuais inoperantes durante o mesmo.

A.1.3 Contribuições

Esta dissertação apresenta ORE (Opportunistic Resilience Embedding – Mapeamento com Resiliência Oportunística), uma nova abordagem de mapeamento de redes para proteger enlaces virtuais contra disrupções no substrato físico. ORE é composto por duas estratégias: uma *proativa*, na qual enlaces virtuais são alocados em *múltiplos caminhos* para mitigar o impacto de uma disrupção; e uma *reativa*, a qual tenta recuperar, *parcial ou integralmente*, a capacidade perdida nos enlaces virtuais afetados. Ambas são modeladas como problemas de otimização. Ademais, como o mapeamento de redes virtuais é $\mathcal{NP}$-Difícil, ORE faz uso de uma meta-heurística baseada em Simulated Annealing para resolver o problema de forma eficiente. Resultados numéricos mostram que ORE pode prover resiliência a disrupções por um custo mais baixo.

A.2 Abordagem Proposta

Nesta seção são definidas as duas estratégias, proativa e reativa, para oferecer resiliência contra disrupções na rede física. A primeira estratégia consiste no mapeamento dos enlaces virtuais em múltiplos caminhos, de forma que o comprometimento de um ou mais enlaces físicos não afete completamente os enlaces virtuais. A segunda estratégia é utilizada quando um ataque obtém sucesso em derrubar um enlace. Essa estratégia tenta realocar a capacidade perdida nos caminhos que não foram afetados.

A.2.1 Alocação de Redes Virtuais com Resiliência

A estratégia proativa realiza necessita realizar o mapeamento de redes virtuais em um substrato físico. Um modelo ótimo para resolver esse problema não escala, uma vez que esse mapeamento é um problema $\mathcal{NP}$-Difícil. Portanto, para solucionar o problema de forma eficiente utiliza-se um algoritmo baseado na meta-heurística Simulated Annealing (SA). Soluções obtidas pelo algoritmo são potencialmente sub-ótimas, no entanto, as mesmas são computadas em tempo pragmático.

O pseudo-código do SA é descrito no Algoritmo A.1. O procedimento inicia com a geração de uma solução inicial. A seguir, a mesma é modificada de forma iterativa até alcançar um número máximo de passos (ou seja, K). A cada passo (k), o SA tenta melhorar a solução atual gerando diversas soluções similares (R) – também chamadas de soluções vizinhas(S'). Adicionalmente, SA utiliza o conceito de temperatura (T) e coeficiente de resfriamento ($0 < c < 1$) para escapar de um mínimo local. Essa técnica seleciona soluções piores com uma probabilidade proporcional à temperatura ($e^{-[cost(S')-cost(S)]/T}$). Como a temperatura é decrescida a cada passo (k), essa ação torna-se menos provável nos últimos passos do SA, quando a solução potencialmente converge para um mínimo global. Finalmente, ao terminar, o algoritmo retorna a melhor solução obtida (S^*).

Algorithm A.1: Pseudo-código do Simulated Annealing

```
Input: $K$                                   // número máximo de passos
Input: $R$                                   // número máximo de sub-passos
Input: $T$                                   // temperatura
Input: $c$                                   // coeficiente de resfiramento
Output: $S^*$                                // melhor solução global

$S \leftarrow$ initial_solution();
$S^* \leftarrow S$;
for $k \leftarrow 1$ upto $K$ do
    for $r \leftarrow 1$ upto $R$ do
        $S' \leftarrow$ neighbor($S$);
        if cost($S'$) $<$ cost($S^*$) then
            $S^* \leftarrow S'$;
        end
        if cost($S'$) $<$ cost($S$) then
            $S \leftarrow S'$;
        else
            with probability $e^{-[\texttt{cost}(S')-\texttt{cost}(S)]/T}$, do $S \leftarrow S'$;
        end
    end
    $T \leftarrow T \cdot c$;
end
```

O algoritmo depende de três componentes principais: (i) a função de custo; (ii) a solução inicial; e (iii) a geração de vizinhos. Esses componentes serão descritos a seguir.

Função de custo. O objetivo de alocação considerado neste trabalho é o balanceamento da carga dos enlaces do substrato físico. Esse objetivo de otimização tende a gerar boas alocações a longo prazo, aumentando a taxa de aceitação de redes virtuais (CHOWDHURY; RAHMAN; BOUTABA, 2012). Para gerar esse balanceamento utiliza-se uma função de engenharia de tráfego clássica da literatura (FORTZ; THORUP, 2004).

Solução inicial. Para calcular um mapeamento inicial utiliza-se um algoritmo guloso composto por três fases: (i) mapeamento de nós; (ii) mapeamento de enlaces; e (iii)

atribuição de capacidade. Na primeira fase, cada nó virtual é posicionado aleatoriamente em nós do substrato. Na segunda, um algoritmo de busca escolhe até k caminhos para mapear cada enlace virtual. Por fim, a terceira fase consiste em distribuir a demanda por largura de banda entre os caminhos escolhidos durante a alocação. Durante as duas primeiras fases (posicionamento e busca), as escolhas seguem probabilidades de forma a gerar soluções distintas.

Vizinhança. Soluções vizinhas são geradas aplicando-se modificações simples na solução corrente. No algoritmo proposto, esse processo consiste em reposicionar um nó virtual e recalcular as duas últimas fases do algoritmo anterior, ou seja, a fase de mapeamento de enlaces e a fase de atribuição de capacidade.

A.2.2 Recuperação Oportunística de Capacidade

Uma contribuição chave do presente trabalho é o que cunhamos de *recuperação oportunística*. Ao invés de reservar recursos sobressalentes, a capacidade perdida após uma disrupção é redistribuída sobre a banda disponível nos caminhos remanescentes dos enlaces virtuais afetados (denominados de *caminhos ativos*).

Quando um enlace físico torna-se inacessível, a capacidade dos enlaces virtuais sobrepostos é totalmente ou parcialmente comprometida. Caso determinado enlace virtual seja parcialmente afetado, ele ainda possuirá um conjunto de caminhos ativos no substrato. Dessa forma, é possível utilizar qualquer banda disponível nesses caminhos para tentar recuperar a capacidade do enlace virtual. A formulação da estratégia de recuperação é dada conforme segue:

Variável:

- $\mathcal{Q}_{p,e^V} \in \mathbb{R}$ no intervalo $[0;1]$: indica a taxa de fluxo do enlace virtual $e^V \in \dot{E}^V$ utilizada no caminho p. Essa taxa é relativa apenas ao fluxo afetado, visto que o restante do fluxo do enlace virtual não é redistribuído.

Entrada:

- $\dot{E}^S \subset E^S$: conjunto de enlaces físicos disponíveis após a disrupção.
- P_{e^V}: conjunto de caminhos ativos para cada enlace virtual após a disrupção.
- $P_{p,e^S,e^V} \in \mathbb{B}$: indica que o enlace físico e^S está sendo utilizado pelo caminho ativo p do enlace virtual e^V.
- $\dot{E}^V \subset E^V$: indica o conjunto de enlaces virtuais afetados.

Objetivo: *maximizar a capacidade de recuperação após uma disrupção.* Dessa forma, a função

$$\max T = \sum_{\forall e^V \in E^V} \sum_{\forall p \in P_{e^V}} \mathcal{Q}_{p,e^V} \tag{A.1}$$

visa realocar todo o tráfego afetado sobre o conjunto de caminhos ativos.

Restrições:

$$\sum_{\forall p \in P_{e^V}} \mathcal{Q}_{p,e^V} \leq F_{e^V} \qquad \forall e^V \in \dot{E}^V \tag{A.2}$$

$$\sum_{\forall e^V \in E^V} \sum_{\forall p \in P_{e^V}} P_{p,e^S,e^V} \cdot \mathcal{Q}_{p,e^V} \cdot b(e^V) \leq b(e^S) \qquad \forall e^S \in \dot{E}^S \tag{A.3}$$

A primeira restrição [Equação (A.2)] impede que o modelo aloque mais banda do que o necessário (ou seja, somente aquela afetada pelo ataque). A segunda restrição [Equação (A.3)] assegura que as capacidades de banda dos enlaces físicos serão respeitadas.

A.3 Avaliação

Esta seção avalia a abordagem proposta através da quantidade de enlaces virtuais afetados por ataques, com e sem proteção, e da quantidade de requisições aceitas na alocação determinada pela abordagem. Todos os experimentos foram executados em um Intel Core i7 com 8 núcleos de 2,93 GHz e 8GB de memória ram. As condições dos experimentos e seus parâmetros são descritos a seguir.

A.3.1 Configuração dos Experimentos

Durante os experimentos o tempo foi discretizado, organizado em rodadas. Cada rodada pode receber uma ou mais requisições e executar o algoritmo de alocação para cada uma delas. Ademais, caso ocorra um evento de disrupção, o algoritmo de recuperação também é executado. A rodada é encerrada somente após a conclusão de todas as etapas pertinentes à alocação e/ou recuperação.

As topologias de rede dos experimentos foram geradas sinteticamente utilizando o *software* BRITE com o modelo de redes Barabási-Albert (BA-2). As configurações das redes e da carga de trabalho são similares às definidas em trabalhos anteriores (CHENG et al., 2012; CHOWDHURY; RAHMAN; BOUTABA, 2009).

A *rede física* é composta por 50 nós e 194 enlaces. As capacidades de processamento dos nós e de largura de banda dos enlaces são uniformemente distribuídas no intervalo [50,100]. *Redes virtuais* variam em tamanho; o número de nós de uma requisição de criação de rede virtual é gerado uniformemente entre 2 e 10. A capacidade dos nós e enlaces é uniformemente distribuída nos intervalos [5,15] e [15,30], respectivamente.

Existem dois tipos de eventos nesse ambiente: requisições de rede virtual e disrupções na rede física. A chegada de requisições é dada por um processo de Poisson com média de 70 para cada 100 rodadas. A duração de cada rede virtual segue uma distribuição geométrica com média de 100 rodadas. Por sua vez, disrupções são modeladas por um processo de Poisson com média de 1 para cada 100 rodadas e duração de 5 rodadas. Cada disrupção ocorre no nó que possui a maior alocação de largura de banda em seus enlaces, no intuito de maximizar o efeito causado (MAGNIEN; LATAPY; GUILLAUME, 2011). Como trata-se do pior caso à vítima, pode-se avaliar de forma mais completa o potencial da abordagem proposta em recuperar o conjunto de redes virtuais afetadas.

Os experimentos executaram durante 10000 rodadas. Dessa forma, durante cada experimento foram geradas, em média, 7000 requisições e 100 ataques.

A.3.2 Avaliação dos Resultados

ORE ajuda a mitigar o impacto de disrupções ao prevenir perda completa de largura de banda

A Figura A.1 ilustra o efeito da disrupção da rede física utilizando múltiplos caminhos e recuperação oportunística. Os eixos x e y representam respectivamente o número de caminhos a ser usado entre dois nós e a perda média de largura de banda (proporcional) para cada enlace virtual causada pela disrupção. Intuitivamente, um número maior de caminhos deve tornar a rede mais resiliente, e quanto menor o valor da curva, melhor.

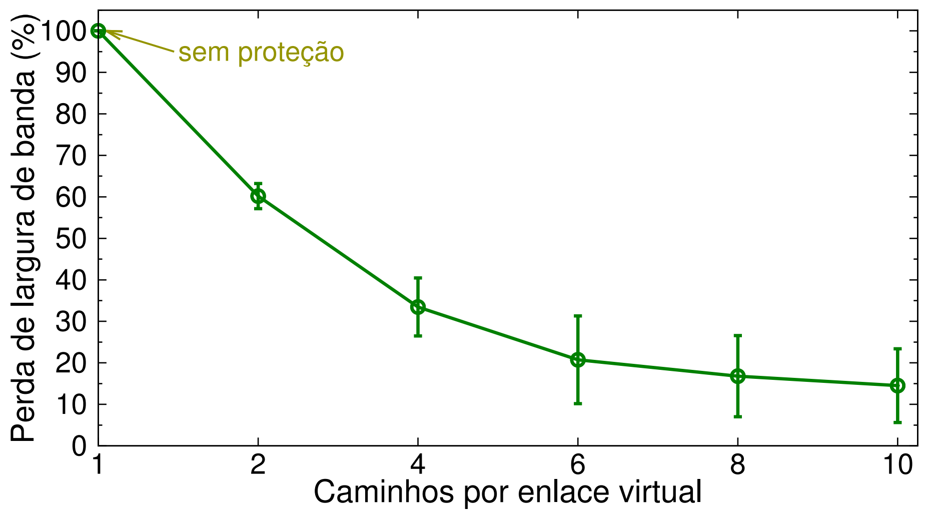


Figure A.1: Perda de largura de banda para cenários com 1 a 10 caminhos por enlace virtual.

Conforme esperado, a eficiência de abordagem proposta é proporcional ao número de caminhos por enlace virtual. Considerando o cenário com 2 caminhos por enlace virtual, é possível perceber que ORE consegue proteger, em média, 39% da capacidade do enlace. Ao incrementar o número de caminhos para 4 e 6, a capacidade protegida cresce para 65% e 79%, respectivamente. O potencial de proteção estabiliza em aproximadamente 85%.

O uso de recursos sobressalentes reduz a taxa de aceitação

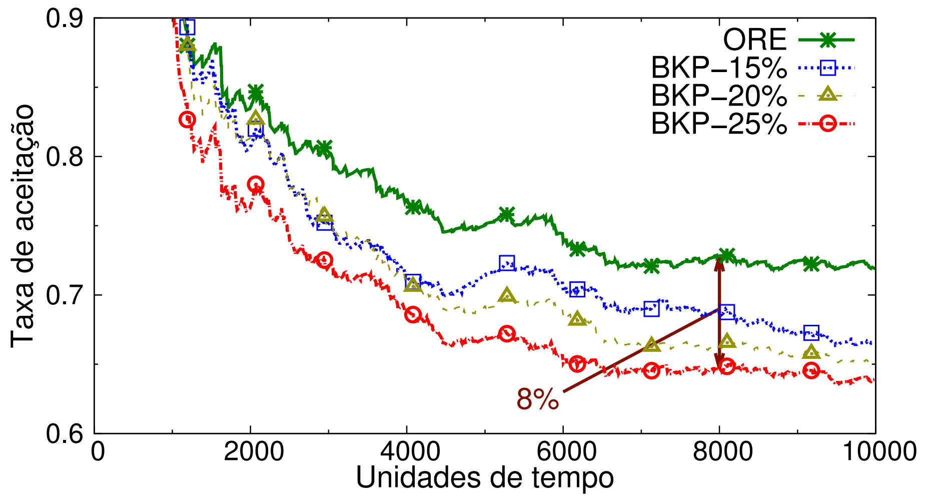


Figure A.2: Comparação da taxa de aceitação entre ORE e esquemas de reserva de recursos sobressalentes.

A Figura A.2 ilustra e comprara a taxa de aceitação de redes virtuais de ORE com esquemas de reserva de recursos sobressalentes (denominados Bkp-X%). Quanto maior for o número de redes virtuais aceitas, maior a receita do provedor de infraestrutura; logo, quanto maior a taxa, melhor. O algoritmo utilizado para a comparação foi SVNE (RAHMAN; AIB; BOUTABA, 2010). Esse algoritmo reserva um percentual de todos recursos disponíveis no substrato como sobressalentes. As configurações de reserva utilizadas foram: 15%, 20% e 25%. Esses valores foram escolhidos pois eles representam a faixa de valores utilizadas em trabalhos anteriores (17% a 22%) (GUO et al., 2011; RAHMAN; AIB; BOUTABA, 2010).

O algoritmo de alocação proposto foi configurado para utilizar dois caminhos por enlace virtual, enquanto os esquemas de reserva de recursos sobressalentes utilizam somente um caminho. Quando comparada com o impacto causado pela reserva de recursos adicionais, a variação no número de caminhos causa impacto negligenciável na taxa de aceitação. Conforme esperado, o uso de recursos sobressalentes resulta em um menor taxa de aceitação. Esta diferença cresce com o percentual de recursos utilizados como sobressalente.

A.4 Conclusão

Apesar das potenciais vantagens, como o isolamento, redes virtuais estão sujeitas a disrupções na rede física. Este trabalho apresentou uma abordagem inovadora, denominada ORE (Opportunistic Resilience Embedding – Mapeamento com Resiliência Oportunística), cujo objetivo é mitigar o impacto que esse tipo de evento causa às redes virtuais. Entre suas contribuições, destaca-se a que a abordagem de alocação proposta, diferentemente dos trabalhos anteriores, não precisa reservar recursos sobressalentes ou migrar recursos virtuais. ORE é composto por duas estratégias complementares. A primeira estratégia visa prevenir os ataques pelo uso de múltiplos caminhos. A segunda visa recuperar a capacidade comprometida utilizando os caminhos ativos (ou seja, aqueles não afetados pelo ataque).

Conforme discutido na seção de avaliação, a solução proposta oferece um ganho proporcional ao número de caminhos utilizados durante a etapa de alocação. Ademais, a estratégia tende a aumentar a taxa de aceitação de redes virtuais quando comparada a estratégias de reserva de recursos sobressalentes.

Quanto aos trabalhos futuros, serão estudados os comportamentos de outras heurísticas e meta-heurísticas para melhorar os resultados. Ademais, vislumbra-se a definição de novas estratégias complementares para o processo de recuperação, como a migração de nós, as quais possibilitarão tornar a rede ainda mais robusta a disrupção.

APPENDIX B - LIST OF PUBLICATIONS

- *OLIVEIRA, R. R.*; MARCON, D. S.; BAYS, L. R.; NEVES, M. C.; BURIOL, L. S.; GASPARY, L. P.; BARCELLOS, M. P. No More Backups: Toward Efficient Embedding of Survivable Virtual Networks. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, 2013., Budapest, Hungary. **Proceedings** IEEE ICC, 2013. pp. 1–5.

- *OLIVEIRA, R. R.*; BAYS, L. R.; MARCON, D. S.; NEVES, M. C.; BURIOL, L. S.; GASPARY, L. P.; BARCELLOS, M. P. DoS-Resilient Virtual Networks through Multipath Embedding and Opportunistic Recovery. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 28., Coimbra, Portugal. **Proceedings** ACM SAC, 2013. pp. 597–602.

- *OLIVEIRA, R. R.*; BAYS, L. R.; MARCON, D. S.; NEVES, M. C.; BURIOL, L. S.; GASPARY, L. P.; BARCELLOS, M. P. Redes Virtuais Seguras: Uma Nova Abordagem de Mapeamento para Proteger contra Ataques de Disrupção na Rede Física. In: SIMPÓSIO BRASILEIRO DE SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS, XII., Curitiba, PR, Brazil. **Anais** SBSeg, 2012. pp. 235–248.

- MARCON, D. S.; *OLIVEIRA, R. R.*; NEVES, M. C.; BURIOL, L. S.; GASPARY, L. P.; BARCELLOS, M. P. Trust-based Grouping for Cloud Datacenters: improving security in shared infrastructure. In: INTERNATIONAL CONFERENCE ON NETWORKING, 12., New York, NY, USA. **Proceedings** IFIP NETWORKING, 2013. pp. 1–9.

- NEVES, M. C.; MARCON, D. S.; *OLIVEIRA, R. R.*; BAYS, L. R.; GASPARY, L. P.; BARCELLOS, M. P. IoNCloud: uma abordagem não entrópica orientada a tráfego para reserva e isolamento de recursos em nuvens. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS, XXXI., Brasília, DF, Brazil. **Anais** SBRC, 2013, pp. 1–14.

- BAYS, L. R.; *OLIVEIRA, R. R.*; BURIOL, L. S.; BARCELLOS, M. P.; GASPARY, L. P. Um Modelo para Mapeamento Ótimo de Redes Virtuais com Requisitos de Segurança. In: SIMPÓSIO BRASILEIRO DE SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS, XII., Curitiba, PR, Brazil. **Anais** SBSeg, 2012. pp. 249–262.

- MARCON, D. S.; NEVES, M. C.; *OLIVEIRA, R. R.*; BURIOL, L. S.; GASPARY, L. P.; BARCELLOS, M. P. Mitigando Ataques de Egoísmo e Negação de Serviço em Nuvens via Agrupamento de Aplicações. In: SIMPÓSIO BRASILEIRO DE SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS, XII., Curitiba, PR, Brazil. **Anais** SBSeg, 2012. pp. 154–167.

- BAYS, L. R.; *OLIVEIRA, R. R.*; BURIOL, L. S.; BARCELLOS, M. P.; GASPARY, L. P. Security-aware Optimal Resource Allocation for Virtual Network Embedding. In MINI-CONFERENCE OF INTERNATIONAL CONFERENCE ON NETWORK AND SERVICE MANAGEMENT, 8., Las Vegas, NV, USA. **Proceedings** IEEE CNSM, 2012. pp. 378–384.

- BAYS, L. R.; *OLIVEIRA, R. R.*; GASPARY, L. P.; BARCELLOS, M. P., MADEIRA, E. R. M.; FONSECA, N. L. S. Capítulo 2: Segurança de Redes Virtuais: Fundamentos, Tecnologias e Tendências. In: ABELÉM, A. J. G.; ALMEIDA, J. M.; NETO, D. O. G. (Ed.). **SBRC 2012, MINICURSOS LIVRO TEXTO**. Porto Alegre, RS, Brazil: SBC, April, 2012, v. 1, pp. 59–98.

APPENDIX C - IEEE ICC ARTICLE

- Title: No More Backups: Toward Efficient Embedding of Survivable Virtual Networks
- Conference: 2013 IEEE International Conference on Communications (ICC 2013)
- URL: `www.ieee-icc.org/2013`
- Date: 9 – 13, june, 2013
- Location: Budapest, Hungary

No More Backups: Toward Efficient Embedding of Survivable Virtual Networks

Rodrigo R Oliveira, Daniel S Marcon, Leonardo R Bays, Miguel C Neves,
Luciana S Buriol, Luciano P Gaspary, Marinho P Barcellos
Federal University of Rio Grande do Sul, Institute of Informatics, Porto Alegre, RS, Brazil
Email: {ruas.oliveira, daniel.stefani, lrbays, mcneves, buriol, paschoal, marinho}@inf.ufrgs.br

Abstract—**Although network virtualization can improve security by isolating traffic from different networks, routers and links are still vulnerable to attacks on the underlying network. High capacity physical links, in particular, constitute good targets since they may be important for a large number of virtual networks. Previous work protects virtual networks by setting aside backup resources. Although effective, this solution increases the cost to infrastructure providers. In this paper, we present a virtual network embedding approach which enables resilience to attacks and efficiency in resource utilization. Our approach is two-folded: while a preventive strategy embeds virtual links into multiple substrate paths, a reactive strategy attempts to reallocate any capacity affected by an underlying DoS attack. Since the embedding problem is NP-Hard, we devise a Simulated Annealing meta-heuristic to solve it efficiently. Results show our solution can provide resilience to attacks at a lower cost.**

I. Introduction

Network virtualization enables the creation of specifically tailored network infrastructures, promoting the instantiation of favorable environments for the development of new architectures and protocols [1]. In addition, applications running over the same physical network can be isolated from each other, thus allowing them to coexist independently. Hence, one of the main advantages of this paradigm is the use of isolation to limit the scope of attacks. This can be achieved by creating different, isolated virtual networks for each task, so traffic from one virtual network does not interfere with the others [2].

In a network virtualization environment, virtual networks are requested *on-demand* by different Service Providers (SPs), and *embedded*[1] in the physical resources of an Infrastructure Provider (InP) [3]. The Virtual Network Embedding (VNE) phase consists of assigning shares of physical resources to the overlaid virtual nodes and links [4]. This characteristic tends to increase the dependency on certain physical resources, allowing an attacker to launch DoS attacks on virtual networks by compromising nodes and links of the underlying physical substrate. Specifically, a successful attack (or a failure) on a physical link affects all embedded virtual links.

Previous research tackled this problem by setting aside additional resources as backup [5]–[9]. Although effective, this strategy may be unaffordable since backup resources cannot be used for new allocations, thus reducing the ability of the network to take new requests. Therefore, these solutions may be of limited applicability, pragmatically leaving virtual networks vulnerable to DoS attacks.

[1]The terms *embed*, *map* and *allocate* will be used interchangeably.

In this paper, we propose a novel embedding approach that optimistically improves the resilience of virtual networks without expending additional resources. Our approach is composed of two complementary optimization strategies, one **preventive** and the other **reactive**. The preventive strategy attempts to mitigate the initial impact of an attack by embedding each virtual link into *multiple paths*, thus preventing the virtual links from losing all their capacity. The reactive strategy aims at partially or fully recovering any capacity compromised by attacks. To achieve this goal, the set of unaffected paths (if such paths exist) is used to *opportunistically recover* the compromised capacity.

The first strategy requires solving the VNE problem. In a previous paper, we provide a full mixed-integer program of a variation of this strategy [10]. Yet, since the VNE problem is NP-Hard [4], [11]–[13], MIP models cannot scale. Hence, in order to achieve solutions within reasonable timeframes, we use a Simulated Annealing meta-heuristic. It runs efficiently and provides near-optimal solutions, converging towards a global optimum. The second strategy requires solving a max-flow problem over a set of pre-computed paths (i.e., those computed by the first strategy). We present a linear program which solves this problem.

Overall, the contributions of this paper are three-fold: (i) we devise a novel embedding approach aimed at providing efficient embedding and resilience to virtual networks; (ii) we provide a Simulated Annealing-based algorithm for the embedding process, thus achieving near-optimal solutions at practical computing time; and (iii) we perform experiments to measure our approach and compare it's benefit against traditional, backup-oriented schemes.

The rest of the paper is organized as follows: Section II defines the attack model, the implications in the use of multipath embedding, and the main concepts related to virtual network embedding. Section III presents our solution, that is, the preventive and the reactive strategies. Section IV discusses numerical results. Related work is reviewed in Section V, and Section VI concludes this paper.

II. Preliminaries

A. Attack Model

We focus on the disruption of the communication between virtual nodes caused by attacks (or failures) compromising parts of the substrate network.

Attacks to physical routers and links. A DoS attack to a physical device may occur by obtaining physical access, or by exploiting vulnerabilities. The former happens when an attacker identifies the location of an optical fiber or a router and interrupts these devices (e.g., strategically planed fiber cuts have been known to affect thousands of customers[2]). The later occurs when an attacker exploits some vulnerability on the control software or protocol to compromise a device (recent news suggest that numerous networking equipment on the Internet operate with insecure firmware[3]).

Assumptions. We assume the attacker might (partially) infer the topology, identifying the most critical elements. The worst case scenario corresponds to an attack compromising the node with most aggregated bandwidth embedded on its incident links. We also consider that the disruption of a node can be reduced to the disruption of multiple links, that is, those directly connected to the compromised node.

B. Multipath Concerns

There exists two primary concerns when splitting the capacity of a virtual link over multiple paths. First, the selected paths may present different propagation delays, which aggravates out-of-order delivery and may hurt application performance. Second, each path requires additional resources (table entries) on routers and periodic maintenance (e.g., availability checks).

Addressing out-of-order delivery. In non-virtual IP networks, this problem is generally prevented by multiplexing network traffic at flow-level granularity. This is achieved by assigning each flow to a specific path based on its "flow-id", that is, five fields in the packet header (namely the src/dst IP addresses, the transport protocol, and the src/dst ports) [14]. To utilize such techniques, a virtual network environment would need to be able to identify from which flow belongs each packet. This should be straightforward, as non-IP virtual networks would only be required to inform which bits in the packet header should be used to identify a flow [13].

Limiting overhead. To tackle the second problem, the number of paths should be kept as low as possible, thus reducing the overhead caused by additional table entries and control messages [15]. A practical approach for embedding a virtual link into multiple paths can be achieved by using a multi-commodity flow formulation. Yet, although such approaches yield good solutions, they generate an unpredictable number of paths [15]. Therefore, any solution for assigning virtual links to multiple paths should be upper bounded, thus allowing better control over the outcome of the embedding phase.

C. Virtual Network Embedding

Virtual network embedding consists of allocating virtual nodes and links, respectively, on substrate nodes and paths. Each virtual node is embedded in a single physical node, and each virtual link is embedded into *one or more paths* of the substrate network. Additionally, virtual nodes of the *same* virtual network should be embedded on *distinct* physical nodes. The embedding of virtual networks is performed *on-demand*, as requests arrive. Hence, it is not possible to determine beforehand which virtual networks should be embedded. This definitions are similar to [4], [12], [13].

[2]http://goo.gl/8KWXG, http://goo.gl/NeupI
[3]http://goo.gl/w3yNB, http://goo.gl/dgmtn

Substrate. The substrate network is given by a weighted directed graph $G^S(N^S, E^S, A^S)$, where N^S is the set of nodes, E^S is the set of links and A^S is the set of node and link attributes. We focus on the following attributes: $c(n^S)$ is the *CPU capacity* of node $n^S \in N^S$, and $b(e^S)$ is the *bandwidth capacity* of link $e^S \in E^S$.

Virtual network request. Each request is modeled as a weighted directed graph $G^V(N^V, E^V, A^V)$, where sets N^V, E^V and A^V are analogous to the aforementioned ones.

Mapping. The bound in number of paths per virtual link is denoted by $|P|$, where set P enumerates the paths from 1 to $|P|$. Binary variables $\mathcal{M}(n^V, n^S)$ and $\mathcal{P}(p, e^V, e^S)$ denote node and link assignments, respectively. $\mathcal{M}(n^V, n^S)$ indicates if a virtual node $n^V \in N^V$ is mapped into physical node $n^S \in N^S$, whereas $\mathcal{P}(p, e^V, e^S)$ indicates if link $e^S \in E^S$ is used in the mapping of path $p \in P$ from virtual link $e^V \in E^V$. In addition, functions $alloc(.)$ and $resid(.)$ denote, respectively, the amount of allocated and residual resources of a given element (e.g., $alloc(e^S)$ indicates the amount of bandwidth already allocated on link e^S).

III. Mitigating Disruptions

In this section, we present an approach for improving the resilience of virtual networks against attacks that compromise physical devices in an attempt to disrupt the embedded virtual networks. Toward this end, we use two complementary strategies, one preventive (§ III-A) and the other reactive (§ III-B).

A. Resilient Virtual Network Embedding

This strategy attempts to embed virtual network requests on a given substrate network with respect to the previously mentioned constraints. Our goal is to provide resilience for virtual networks and efficient allocation of substrate resources.

Resilience is attained by distributing all the capacity of a virtual link over a set of substrate paths. This way, if a device embedding virtual links is compromised, only a part of the capacity of these links is lost. However, unless we select paths which are sufficiently different, a virtual link will still be vulnerable when a single substrate link embeds most of the virtual link capacity. Thus, we choose to avoid path similarity when selecting paths to embed virtual links.

Efficient allocation is achieved by performing load balance over the set of substrate links. As mentioned earlier, virtual network requests are received on-demand. This implies that, on the long-run, a set of physical substrate links may become saturated. When this happens, virtual network requests that need to use a given substrate link (or a set of substrate links) may be rejected. Therefore, we try to avoid the creation of bottlenecks in order to improve acceptance ratio.

To solve this problem efficiently, we use a Simulated Annealing (SA) meta-heuristic, as depicted in Fig. 1. Instead of exploring the entire solution space for the optimal mapping, this algorithm iteratively generates possible mappings until a maximum number of steps (i.e., K) is reached. In each step (k), SA tries to improve the current solution by comparing it to several (R) similar solutions (S' – neighbors). In addition, SA uses the concept of temperature (T) and cooling factor ($0 < c < 1$) to escape local minima. This works by

```
$S \leftarrow initial\_solution()$
$S^* \leftarrow S$
for $k \leftarrow 1$ upto $K$ do
    for $r \leftarrow 1$ upto $R$ do
        $S' \leftarrow neighbor(S)$
        if $cost(S') < cost(S^*)$ then
            $S^* \leftarrow S'$
        end if
        if $cost(S') < cost(S)$ then
            $S \leftarrow S'$
        else
            with probability $e^{-[cost(S')-cost(S)]/T}$, do $S \leftarrow S'$
        end if
    end for
    $T \leftarrow T \cdot c$
end for
return $S^*$
```

Fig. 1. Pseudo-code of Simulated Annealing

selecting slightly worse solutions with the probability being directly proportional to the temperature ($e^{-[cost(S')-cost(S)]/T}$). Since temperature is decreased at each step (k), this action is less likely to happen at the last few steps of SA, when the solution has probably converged to a global minimum. Finally, when SA terminates, the best overall attained solution (S^*) is returned.

The parameters of SA, namely the number of steps (K), the number of neighbors per step (R), the temperature (T), and the cooling factor (c), were tuned during experimentation. Therefore, we defer their discussion to Section IV. Next, we present the key components of Simulated Annealing, that is: (i) the *initial solution*, which is iteratively modified in an attempt to converge to the global optimum; (ii) the *neighbor* function, used to modify a solution; and (iii) the *cost* function that measures the quality of a solution.

Initial solution. Fig. 2 describes the procedure for generating an initial solution. This procedure is composed of three steps: the node embedding phase, the link embedding phase, and the capacity assignment phase. In the node embedding phase, each virtual node is randomly placed on substrate nodes with enough capacity to hold them. Since virtual nodes from the same request cannot share the same physical node, the probability of choosing a physical node (p_n) is inversely proportional to the number virtual nodes already allocated:

$$p_{n^S} = \frac{1}{1 + \sum\limits_{\forall n^V \in N^V} \mathcal{M}(n^V, n^S)},$$

where p_{n^S} is normalized so that $\sum\limits_{\forall n^S \in V} p_{n^S} = 1$.

The link embedding phase uses a breadth-first search to allocate the paths of each virtual link. Instead of selecting edges deterministically, we use a probability p_e based on two metrics: (a) link utilization; and (b) path similarity. The higher the value of one of these metrics (or both), the lower the probability to select the link:

$$p_{e^S} = \frac{1}{1 + alloc(e^S)} + \frac{1}{1 + \sum\limits_{\forall p^V \in |P|} \mathcal{P}(p, e^V, e^S)}.$$

Similarly to p_{n^S}, this probability is normalized so that p_{e^S} of all neighboring links sum to 1.

```
***Node embedding phase:***
for $n^V \in N^V$ do
    $V \subseteq N^S \mid n^S \in V \iff c(n^S) \geq c(n^V)$
    choose $n^S \in V$ with probability $p_{n^S}$
end for
***Link embedding phase:***
for $e^V \in E^V$ do
    breadth-first search for each $p \in |P|$
    for $p \in |P|$ do
        choose $e^S \in E^S$ with probability $p_{e^S}$
    end for
end for
***Capacity assignment phase:***
for $e^V \in E^V$ do
    for $p \in |P|$ do
        Proportionally assign $x_p\%$ of $b(e^V)$ to path $p$
    end for
end for
```

Fig. 2. Pseudo-code of the Initial Solution procedure

```
Randomly select a virtual node $n^V \in N^V$
Perform ***Node embedding phase*** for selected $n^V$
Perform ***Link embedding phase*** as necessary
Perform ***Capacity assignment phase*** as necessary
```

Fig. 3. Pseudo-code of the Generate Neighbor procedure

Finally, in the capacity assignment phase consists of assigning a fraction of the virtual link capacity to each path. In this phase, each path will receive a fraction proportional to its available bandwidth, as follows:

$$x_p\% = \frac{resid(p)}{\sum\limits_{\forall q \in P} resid(q)}.$$

Notice that before actually assigning a capacity to a path, it is necessary to check whether the path has enough bandwidth. Thus, the amount of bandwidth allocated to a path p is $\min(resid(p), x_p\% \cdot b(e^V))$.

Generating neighbors. SA uses neighbors in an attempt to improve the current solution or to escape a local minimum. A neighbor is generated by applying simple modifications to the current solution, resulting on a similar (potentially better) solution. Fig. 3 shows a pseudocode of this procedure, which consists of moving a randomly selected virtual node to another physical node and recalculating paths affected by this action.

Cost function. Finally, the quality of attained solutions is compared by measuring link saturation over the network. This can be achieved by employing traffic engineering functions. Thus, the link saturation metric uses the classic TE function proposed in [16] with the same values defined by the authors.

B. Opportunistic Capacity Recovery

One of the key contributions of this paper is what we call *opportunistic recovery*. Instead of allocating backup resources, the capacity compromised by a successful DoS attack is reallocated over the available bandwidth along the set of unaffected paths. The only requirement is that the path was previously selected to allocate a virtual link and remained operational after an attack. From this point onwards, we will refer to such paths as *active paths*.

Whenever a physical link becomes inactive, one or more paths of a virtual link may disappear. A virtual link requires at least one active path, otherwise it means the virtual link has been fully compromised. As long as one or more active paths remain available, any spare bandwidth remaining on these paths can be used to restore the capacity of the virtual link. The formulation of this strategy is given as follows:

Variable:

- $\mathcal{X}_{p,e^V} \in \mathbb{R}$ within $[0,1]$: the fraction of capacity each virtual link $e^V \in \dot{E}^V$ reallocates to path p.

Input:

- $\hat{E}^S \subset E^S$: substrate links that remain available after a disruption event;
- $\dot{E}^V \subset E^V$: virtual links affected by the disruption;
- P_{e^V}: active paths of virtual link e^V after disruption;
- F_{e^V}: compromised capacity of virtual link e^V;
- $P_{p,e^S,e^V} \in \mathbb{B}$: indicates if physical link e^S is being used by active path p on virtual link e^V.

Objective: *maximize the amount of bandwidth recovered after disruption events.*

$$\max Z = \sum_{\forall e^V \in E^V} \sum_{\forall p \in P_{e^V}} \mathcal{X}_{p,e^V} \tag{1}$$

Constraints:

$$\sum_{\forall p \in P_{e^V}} \mathcal{X}_{p,e^V} \leq F_{e^V}, \qquad \forall e^V \in \dot{E}^V \tag{2}$$

$$\sum_{\forall e^V \in E^V} \sum_{\forall p \in P_{e^V}} P_{p,e^S,e^V} \cdot \mathcal{X}_{p,e^V} \cdot b(e^V) \leq b(e^S), \quad \forall e^S \in \hat{E}^S \tag{3}$$

Remarks: This Linear Program requires two constraints. The first constraint [Eq. (2)] ensures that the amount of bandwidth reallocated is not greater than the amount compromised by the attack; otherwise, the objective function [Eq. (1)] would try to exhaust all spare bandwidth of the active paths. The second constraint [Eq. (3)] guarantees that the allocated bandwidth stays within the available capacity of physical links.

IV. Evaluation

A. Evaluation Settings

The choice of settings is similar to [4], [12]. Experiments where executed during 10000 rounds. The network topologies were synthetically generated using BRITE with the Barabási-Albert (BA-2) network model.

Substrate. The physical network is composed of 50 nodes and around 200 links. The CPU and bandwidth capacity of nodes and links are uniformly distributed within [50,100][4].

Virtual networks. The number of nodes is uniformly distributed between 2 and 10. The capacity of nodes and links is uniformly distributed within [5,15] and [15,30], respectively.

Workload. The arrival rate of each request is given by a Poisson process with an average of 7 requests per 100 rounds. The duration of each request is geometrically distributed with an average of 1000 rounds. The attacks are modeled by a Poisson process with mean of 1 attack per 100 rounds and duration of 10 rounds. Thus, each experiment generated, approximately, 700 requests and 100 attacks.

SA parameters. The main idea behind these parameters is that by increasing any of them, we achieve better solutions at the cost of more computation. Thus, we increased their values until there was no sensitive improvement in the quality of the solutions. The set of parameters used in our experiments were the following: The number of steps (K) was set to 80, the number of neighbors per step (R) to 2000, the temperature (T) to 3000, and the cooling factor (c) to 0.96. Notice that K and R have multiplicative cost, that is, increasing K by 1 results in R additional substeps.

B. Comparison Method

To measure the efficiency of our approach, we need to compare it with backup-based schemes. Among possible choices we choose [5] as it generalizes the main concept of backup-based schemes. Authors propose setting a fraction of every physical link as backup (say 20%). Then, an offline phase pre-calculates paths (detours) for every physical link. Whenever a disruption on a physical link occurs, the capacity embedded in this link is reallocated in the backup resources of the detours.

C. Evaluation Results

The use of backup resources reduces long-term acceptance rate. Fig. 4 shows the comparison in acceptance rate between our embedding approach (denoted as No-Bkp) against backup-oriented schemes (denoted Bkp-X%). We choose to configure our solution to use two paths per virtual link, while backup-oriented schemes used single-path embedding. Varying the number of paths caused negligible difference in acceptance rate when compared to the penalty incurred by using backups. We used three settings for backup resources: 15%, 20%, and 25%. These values were chosen since the fraction of backup resources varies between 17% and 22% in previous work [5], [6]. As expected, the use of backup resources results in lower long-term acceptance rate. This difference tends to grow larger as more resources are used for backup purposes (e.g., 8% lower when using 25% of resources as backup).

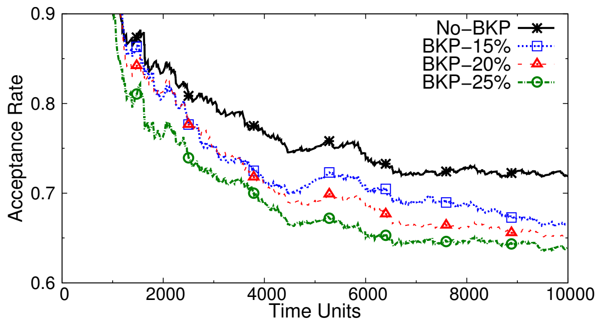


Fig. 4. Acceptance rate for our solution and backup-oriented schemes.

Improving resilience through multipath embedding and opportunistic recovery. Fig. 5 shows the bandwidth lost after disruption events. This loss is reduced by incrementing the number of paths, stabilizing at approximately 15%. The observed non-linear behavior is due to the capacity of the

[4]Similarly to previous work, we kept units abstract; they could represent Mbps on links or time slices on nodes.

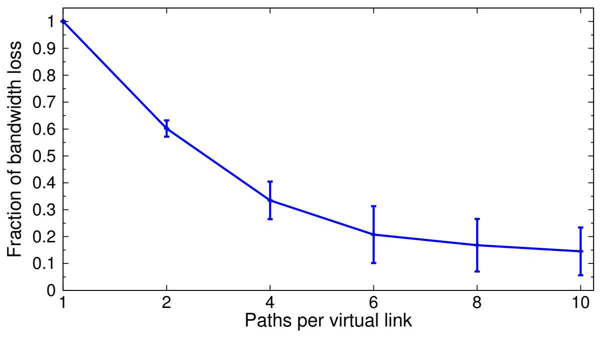


Fig. 5. Bandwidth loss on scenarios with 1 to 10 paths per virtual link.

substrate network to provide disjoint end-to-end paths. As the number of paths increase, so does the likelihood of paths being mapped onto the same physical link.

V. Related Work

Protection and Restoration in Optical Networks. Most network survivability optimization problems are formulated at the network design, thus assuming traffic demands are given as a parameter of the model (i.e., offline). In contrast, in network virtualization, requests are received in an on-demand basis (i.e., online). The most relevant work in this area is [17], which considers on-demand connection requests on SONET/SDH networks with pre-defined source/sink nodes. However, their work is unsuitable for network virtualization as it does not consider node embedding (i.e., placement and requirements).

Survivable Virtual Network Embedding. Generally, survivability in virtual networks is provided by reserving backup resources [5]–[9]. [5] preallocates a fraction of bandwidth on each substrate link as backup and allows flows to take detours. [6] allocates disjoint backup paths with sufficient additional resources to restore each primary path. [7], [8] attempt to protect virtual networks by allocating physically disjoint backup nodes and backup paths between these nodes. [9] aims to protect virtual networks against regionalized disruption events by replicating virtual nodes and by making backup paths traverse different locations. All these solutions, despite being effective, may became too expensive as backup resources hinder the ability of the substrate network to accept future requests.

In contrast, [18] attempts to recalculate virtual link embeddings when disruptions occur. Although schemes like that do not rely on backup resources, reassigning paths may demand a long convergence time, leaving virtual networks inoperable during such periods.

Our approach differs from previous work because (a) resilience is improved without the need to reserve backup resources; and (b) recoverability is achieved without the need to recalculate and reallocate end-to-end paths.

VI. Conclusions

In this paper, we presented a novel virtual network embedding algorithm based on the Simulated Annealing metaheuristic for protecting virtual networks against DoS attacks and failures in the physical substrate. Unlike previous work, our approach does not need to set aside backup resources or recalculate end-to-end paths to reallocate virtual links. Rather, it employs multiple path virtual link embedding and opportunistic recovery to mitigate the impact of attacks. Results show that the proposed approach provides resilience to virtual networks with a higher long-term acceptance rate.

In ongoing work, we are studying techniques to protect different types of virtual network topologies against connectivity attacks. Moreover, we aim at investigating multi-layer optimization to avoid overlapping protection mechanisms.

Acknowledgment

This work has been supported by FP7/CNPq (Project SecFuNet, FP7-ICT-2011-EU-Brazil), RNP-CTIC (Project ReVir), as well as PRONEM/FAPERGS/CNPq (Project NPRV).

References

[1] M. Chowdhury and R. Boutaba, “A survey of network virtualization,” *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010.

[2] A. Khan, A. Zugenmaier, D. Jurca, and W. Kellerer, “Network virtualization: a hypervisor for the internet?” *Communications Magazine, IEEE*, vol. 50, no. 1, pp. 136–143, 2012.

[3] A. Belbekkouche, M. Hasan, and A. Karmouch, “Resource discovery and allocation in network virtualization,” *Communications Surveys & Tutorials, IEEE*, vol. 14, no. 4, pp. 1114–1128, 2012.

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

[5] M. R. Rahman, I. Aib, and R. Boutaba, “Survivable virtual network embedding,” in *NETWORKING 2010*, ser. LNCS, M. Crovella, L. Feeney, D. Rubenstein, and S. Raghavan, Eds. Springer Berlin / Heidelberg, 2010, vol. 6091, pp. 40–52.

[6] T. Guo, N. Wang, K. Moessner, and R. Tafazolli, “Shared backup network provision for virtual network embedding,” in *Proc. ICC, IEEE*, 2011, pp. 1–5.

[7] W.-L. Yeow, C. Westphal, and U. C. Kozat, “Designing and embedding reliable virtual infrastructures,” in *2nd SIGCOMM VISA workshop*. ACM, 2010, pp. 33–40.

[8] H. Yu, V. Anand, C. Qiao, and G. Sun, “Cost efficient design of survivable virtual infrastructure to recover from facility node failures,” in *Proc. ICC, IEEE*, 2011, pp. 1–6.

[9] H. Yu, C. Qiao, V. Anand, X. Liu, H. Di, and G. Sun, “Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures,” in *Proc. GLOBECOM, IEEE*, 2010, pp. 1–6.

[10] R. R. Oliveira, L. R. Bays, D. S. Marcon, M. C. Neves, L. S. Buriol, L. P. Gaspary, and M. P. Barcellos, “Dos-resilient virtual networks through multipath embedding and opportunistic recovery,” in *Proc. SAC, ACM*, 2013, pp. 597–602.

[11] D. G. Andersen, “Theoretical approaches to node assignment,” 2002. [online]. Available: http://www.cs.cmu.edu/~dga/papers/andersen-assign.ps.

[12] X. Cheng, S. Su, Z. Zhang, K. Shuang, F. Yang, Y. Luo, and J. Wang, “Virtual network embedding through topology awareness and optimization,” *Computer Networks*, vol. 56, no. 6, pp. 1797–1813, 2012.

[13] M. Yu, Y. Yi, J. Rexford, and M. Chiang, “Rethinking virtual network embedding: substrate support for path splitting and migration,” *SIGCOMM CCR, ACM*, vol. 38, pp. 17–29, March 2008.

[14] J. He and J. Rexford, “Toward internet-wide multipath routing,” *Network, IEEE*, vol. 22, no. 2, pp. 16–21, March-April 2008.

[15] T. Hartman, A. Hassidim, H. Kaplan, D. Raz, and M. Segalov, “How to split a flow?” in *Proc. INFOCOM, IEEE*, 2012, pp. 828–836.

[16] B. Fortz and M. Thorup, “Increasing internet capacity using local search,” *Computational Optimization and Applications*, vol. 29, pp. 13–48, 2004.

[17] S. Huang, C. Martel, and B. Mukherjee, “Survivable multipath provisioning with differential delay constraint in telecom mesh networks,” *Transactions on Networking, IEEE/ACM*, vol. 19, no. 3, pp. 657–669, 2011.

[18] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, and L. Mathy, “Adaptive virtual network provisioning,” in *2nd SIGCOMM VISA workshop*. ACM, 2010, pp. 41–48.

APPENDIX D - ACM SAC ARTICLE

- Title: DoS-Resilient Virtual Networks through Multipath Embedding and Opportunistic Recovery
- Conference: 28th ACM Symposium On Applied Computing (SAC 2013)
- URL: `http://www.acm.org/conferences/sac/sac2013`
- Date: 18 – 22, march, 2013
- Location: Coimbra, Portugal

DoS-Resilient Virtual Networks through Multipath Embedding and Opportunistic Recovery

Rodrigo R Oliveira, Leonardo R Bays, Daniel S Marcon, Miguel C Neves
Luciana S Buriol, Luciano P Gaspary, Marinho P Barcellos
Federal University of Rio Grande do Sul (UFRGS)
Institute of Informatics
Porto Alegre, RS, Brazil
{ruas.oliveira, lrbays, daniel.stefani, mcneves, buriol, paschoal, marinho}@inf.ufrgs.br

ABSTRACT

Network virtualization can potentially limit the impact of attacks by isolating traffic from different networks. However, routers and links are still vulnerable to attacks on the underlying network. Specifically, should a physical link be compromised, all embedded virtual links will be affected. Previous work protects virtual networks by setting aside backup resources. Although effective, this solution tends to be expensive as backup resources usually remain idle. In this paper, we present a novel virtual network allocation approach which explores the trade-off between resilience to attacks and efficiency in resource utilization. Our approach is composed of two complementary strategies, one *preventive* and the other *reactive*. The former embeds virtual links into multiple substrate paths, while the latter attempts to reallocate any capacity affected by an underlying DoS attack. Both strategies are modeled as optimization problems. Numerical results show the level of resilience to attacks and the low cost demanded by our approach.

Categories and Subject Descriptors

C.2.0 [**Computer-communication networks**]: General–*Security and Protection*

General Terms

Design, Optimization, Security

Keywords

Network Virtualization, Resource Allocation, Denial-of-Service

1. INTRODUCTION

Network virtualization allows multiple virtual networks to be embedded on the same physical substrate. In addition, virtual networks can be isolated from each other, thus allowing them to coexist independently. Hence, one of the main advantages of this paradigm is the use of isolation to limit the scope of attacks. This can be achieved by creating different, isolated virtual networks for each task, so traffic from one virtual network does not interfere with the others [11].

In a network virtualization environment, virtual networks are requested *on-demand* by different Service Providers (SPs), and *embedded* in the physical resources of an Infrastructure Provider (InP) [2]. The Virtual Network Embedding (or Allocation/Mapping) phase consists of assigning shares of physical resources to the overlaid virtual nodes and links [5]. This characteristic tends to increase dependency on certain physical resources, allowing an attacker to launch DoS attacks on virtual networks by compromising nodes and links of the underlying physical substrate. Specifically, a successful attack on a physical link affects all embedded virtual links. Note that, although this article focuses on attacks, the proposed solutions can also be applied to failures.

Previous research tackled this problem by setting aside additional resources as backup [12, 3, 8, 13, 14]. Although effective, this strategy may be too expensive since backup resources remain idle (i.e. are wasted) when there are no disruptions. Therefore, these solutions may be of limited applicability, pragmatically leaving virtual networks vulnerable to DoS attacks.

In this paper, we propose a novel embedding approach that improves the resilience of virtual networks without expending additional resources. Our approach is composed of two complementary strategies, one **preventive** and the other **reactive**. The preventive strategy, modeled as a Mixed-Integer Program (MIP), attempts to mitigate the initial impact of an attack by embedding each virtual link into *multiple paths*, thus preventing the virtual links from losing all their capacity. The reactive strategy, modeled as a Linear Program (LP), aims at partially or fully recover any capacity compromised by attacks. To achieve this goal, it uses any unaffected path (if such a path exists) to *opportunistically recover* the compromised capacity.

The proposed approach has to deal with three fundamental problems: (*i*) similarity among paths embedding the same virtual link can be exploited to increase the impact of an attack, thus reducing the effectiveness of our approach; (*ii*) online resource allocation may overload some physical links, causing virtual network requests that depend on those resources to be dropped; and (*iii*) the selection of paths with distinct end-to-end propagation delays aggravates out-of-order packet delivery. These problems are addressed with penalty functions that aim at achieving path disjointness, load balancing and differential delay minimization (§3.3). Overall, the contributions of this paper are three-fold:

- **Joint modeling of the three factors of the problem (§3).** We combine the relative impact of path disjointness, load balancing and differential delay, thus achieving equilibrium between virtual network resilience and efficient resource allocation.
- **Formulation of the strategies as optimization problems (§4).** The first strategy (*multipath embedding*) is modeled as a MIP, while the second (*opportunistic capacity recovery*) is modeled as a LP. These formulations generate *optimal solutions* in accordance to the predefined priorities (see above).
- **Analysis of the trade-off addressed by our solution (§5).** Experiments show that the proposed approach can substantially mitigate DoS by decreasing bandwidth lost to 25% and reducing critical targets (single point of failure) to 1%. These benefits demand a low cost in acceptance rate (at most 14% less in our experiments).

The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 defines the attack model, the main concepts related to virtual network embedding, and the three factors considered in our optimization model. Section 4 presents the MIP/LP optimization models for the two proposed strategies. Numerical results are discussed in Section 5, and Section 6 concludes this paper.

2. RELATED WORK

We focus on mitigating attacks that compromise physical resources to cause DoS on virtual networks. Since this problem is closely related to network survivability, we discuss related work in both conventional and virtualized networks.

Most scenarios on network survivability are considered at the network design, that is, with a given demand matrix. This is not the case for network virtualization since requests are received in an on-demand basis. The most relevant work in this area is that of Huang et al. [10], which considers on-demand connection requests on SONET/SDH networks with pre-defined source/sink nodes. Their work however is unsuitable for network virtualization as node embedding (i.e., placement and requirements) should be considered in addition to link embedding.

Generally, resilience in virtual networks is provided by reserving backup resources. Rahman et al. [12], Chen et al. [3], and Guo et al. [8] try to deal with disruptions on substrate links. While [12] set aside a percentage of bandwidth on each substrate link, [3] and [8] allocate disjoint backup paths with sufficient additional resources to restore each primary path. Yeow et al. [13] attempt to protect virtual networks by allocating physically disjoint backup nodes and backup paths between this nodes. Yu et al. [14] consider regionalized disruption events on the substrate network (i.e., when a single event compromises multiple devices at the same location). In order to survive such events, each virtual node is replicated on a separate location, with backup paths passing through different locations. All these solutions may became too expensive as backup resources remain idle when no disruption occurs.

In contrast, some work attempt to recalculate virtual link embeddings when disruptions occur [9]. Although these schemes do not rely on backup resources, reassigning paths may demand a long convergence time, leaving virtual networks inoperable during such periods.

Our approach differs from previous work for the following reasons: *i*) resilience is improved without the use of backup resources; *ii*) recoverability is achieved without the need to recalculate and reallocate end-to-end paths; and *iii*) constraints are relaxed, which leads to a less expensive strategy for the Infrastructure Provider.

3. PRELIMINARIES

We begin this section with the attack model and related assumptions. Then, we describe a network virtualization environment and the virtual network embedding problem. Finally, we present the objectives used by our multipath embedding formulation.

3.1 Attack Model

The main threat considered in this article is the disruption of the communication between virtual nodes caused by attacks (or failures) compromising parts of the underlying substrate network.

Attacks to physical routers and links. A DoS attack to a physical device may occur by obtaining physical access, or by exploiting vulnerabilities. The former happens when an attacker identifies the location of an optical fiber or a router and interrupts these devices (e.g., strategically planed fiber cuts have been known to affect thousands customers[1]). The later happens when an attacker exploits some vulnerability on the control software or protocol to compromise a device (recent news suggest that several networking equipment on the Internet operate with insecure firmware[2]).

Assumptions. The scope of the paper is defined by the following set of assumptions:

- The disruption of a node can be reduced to the disruption of multiple links, that is, those directly connected to the disrupted node.
- The attacker might infer the topology or at least part of it, identifying the most critical elements. The worst case scenario corresponds to an attack compromising the node with most aggregated bandwidth embedded on its incident links.
- Virtual networks are isolated among each other. Therefore, a virtual network cannot prejudice another by consuming more resources than it was granted.

3.2 Virtual Network Embedding

Virtual network embedding consists of allocating virtual nodes and links, respectively, on substrate nodes and paths. Each virtual node is embedded in a single physical node, and each virtual link is embedded into *one or more paths* of the substrate network. Moreover, virtual nodes of the *same* virtual network should be embedded on *distinct* physical nodes. The embedding of virtual networks is performed *on-demand*, as requests arrive. Hence, it is not possible to determine beforehand which virtual networks should be embedded. These definitions are similar to [6, 4].

Substrate. The substrate network is given by a weighted directed graph $G^S(N^S, E^S, A^S)$, where N^S is the set of nodes, E^S is the set of links and A^S is the set of node and link attributes. We focus on the following attributes: *CPU capacity* of nodes, and *bandwidth capacity* and *delay* of links.

Virtual network request. Each request is modeled as a weighted directed graph $G^V(N^V, E^V, A^V)$, where sets N^V, E^V and A^V are analogous to the aforementioned ones.

[1]`http://goo.gl/8KWXG`, `http://goo.gl/NeupI`
[2]`http://goo.gl/w3yNB`, `http://goo.gl/dgmtn`

3.3 Optimization Objectives

As previously stated, our approach aims at improving the resilience of virtual networks in a way that can be cost-effective to the substrate network. This design goal requires dealing with three main factors: path disjointness, load-balancing, and differential delay minimization. Thus, in this section, we describe each factor and *what* we do to tackle them. Section 4.1 describes *how* we implement these factors in our formulations.

Prioritizing disjointness. Embedding a virtual link into multiple substrate paths tends to increase resilience to disruptions in the substrate network. However, unless we select paths which are sufficiently different, a virtual link will still be vulnerable when a single substrate link embeds most of the virtual link capacity. In fact, in a worst-case scenario a single link disruption could lead multiple virtual links to lose their entire capacity. Therefore, to address this problem, we define $\Xi = \sum_{\forall e^S \in E^S} \sum_{\forall e^V \in E^V} \xi_{e^S, e^V}$ as a function which penalizes path similarity. For now, assume that ξ_{e^S, e^V} grows proportionally to the number of paths of virtual link e^V that share a given substrate link e^S.

Load balancing. As mentioned earlier, virtual network requests are received *on-demand*. This implies that, on the long-run, a set of physical substrate links may become saturated. When this happens, virtual network requests that need to use a given substrate link (or a set of substrate links) may be rejected. Thus, to avoid this problem we define function $\Phi = \sum_{\forall e^S \in E^S} \phi_{e^S}$, which penalizes saturation of the substrate links. By avoiding saturation, this function tends to balance the overall load of the substrate network and avoid the creation of bottlenecks.

Minimizing differential delay. When splitting the capacity of a virtual link over multiple paths, it may happen that the selected paths present different propagation delays. This aggravates out-of-order packet delivery and hurts application performance. Tackling such problem may require an additional cost in memory or processing capacity for border routers or end-hosts. This additional cost is proportional to the difference of end-to-end delays among paths (i.e., the *differential delay*). Thus, similarly to [15], we define *differential delay minimization* as an optimization objective.

Consider P_{e^V} to be the set of substrate paths selected to embed a given virtual link e^V. Also consider D_p to be the delay of a given path p and $\delta_{e^V} = \max_{\forall p,q \in P_{e^V}} (\mid D_p - D_q \mid)$ to be the maximum difference of propagation delays among the set of paths P_{e^V}. Therefore, $\Delta = \sum_{\forall e^V \in E^V} \delta_{e^V}$ penalizes the differential delay of each virtual link.

4. MITIGATING DISRUPTIONS

In this section, we present an approach for improving the resilience of virtual networks against denial of service attacks. Our focus is to protect against attacks that compromise physical devices in an attempt to disrupt the embedded virtual networks. Toward this end, we use two complementary strategies, one preventive (§ 4.1) and the other reactive (§ 4.2), as follows.

4.1 Resilient Virtual Network Embedding

This strategy aims at providing resilience by embedding each virtual link into a set of substrate paths such that: (i) all capacity of those links are distributed along the paths; and (ii) substrate paths of the same virtual link have little or no similarity. Our formulation is given as follows.

Variables:

- $\mathcal{X}_{n^S, n^V} \in \mathbb{B}$: indicates if substrate node $n^S \in N^S$ is being used to embed virtual node $n^V \in N^V$;
- $\mathcal{P}_{p, e^S, e^V} \in \mathbb{B}$: indicates if path p uses substrate link $e^S \in E^S$ to embed the virtual link $e^V \in E^V$;
- $\mathcal{F}_{p, e^S, e^V} \in \mathbb{R}$ within $[0, 1]$: indicates the amount of capacity of the virtual link $e^V \in E^V$ to be allocated along substrate link e^S of path p.

Input:

- $c(.) \in A^*$: CPU capacity of a (physical/virtual) node;
- $b(.) \in A^*$: bandwidth of a (physical/virtual) link;
- $d(.) \in A^S$: propagation delay of a physical link;
- $P, |P|$: represent, respectively, the set of path indices and the cardinality of this set (i.e., maximum number of paths per virtual link).

Objective: *minimize the impact of the three factors of the problem* (§3.3),

$$\min Z = w_\Xi \cdot \Xi + w_\Phi \cdot \Phi + w_\Delta \cdot \Delta, \tag{1}$$

where w_Φ, w_Ξ, and w_Δ indicate the relative importance of each part of the objective function. The definition of Ξ, Φ, and Δ will be given after the constraints.

Constraints:

$$\sum_{\forall n^S \in N^S} \mathcal{X}_{n^S, n^V} = 1, \qquad \forall n^V \in N^V; \tag{2}$$

$$\sum_{\forall n^V \in N^V} \mathcal{X}_{n^S, n^V} \leq 1, \qquad \forall n^S \in N^S; \tag{3}$$

$$\sum_{\forall n^V \in N^V} \mathcal{X}_{n^S, n^V} \cdot c(n^V) \leq c(n^S), \qquad \forall n^S \in N^S; \tag{4}$$

$$\sum_{\forall p \in P} \sum_{\forall e^V \in E^V} \mathcal{F}_{p, e^S, e^V} \cdot b(e^V) \leq b(e^S), \qquad \forall e^S \in E^S; \tag{5}$$

$$\sum_{\substack{\forall b^S \in N^S: \\ (a^S, b^S) \in E^S}} \mathcal{P}_{p, (a^S, b^S), e^V} - \sum_{\substack{\forall b^S \in N^S: \\ (b^S, a^S) \in E^S}} \mathcal{P}_{p, (b^S, a^S), e^V}$$
$$= \mathcal{X}_{a^S, s^V} - \mathcal{X}_{a^S, t^V}, \tag{6}$$
$$\forall p \in P, \forall a^S \in N^S, \forall e^V = (s^V, t^V) \in E^V;$$

$$\sum_{\forall p \in P} \sum_{\forall b^S \in N^S} \mathcal{F}_{p, (a^S, b^S), e^V} - \sum_{\forall p \in P} \sum_{\forall b^S \in N^S} \mathcal{F}_{p, (b^S, a^S), e^V}$$
$$= \mathcal{X}_{a^S, s^V} - \mathcal{X}_{a^S, t^V}, \tag{7}$$
$$\forall a^S \in N^S, \forall e^V = (s^V, t^V) \in E^V : (a^S, b^S), (b^S, a^S) \in E^S;$$

$$\mathcal{F}_{p, e^S, e^V} \leq \mathcal{P}_{p, e^S, e^V}, \qquad \forall p \in P, \forall e^S \in E^S, \forall e^V \in E^V. \tag{8}$$

Constraints (2) and (3) guarantee, respectively, that each virtual node is embedded in exactly one physical node and that they are embedded on different physical nodes. Constraints (4) and (5) ensure that physical node and link capacities are not exceeded. The composition of a valid end-to-end path is ensured by Constraint (6). Finally, the last two constraints define the amount of bandwidth capacity allocated on each path. They ensure flow conservation on each path [Constr. (7)] and that traffic only flows on links that are selected by paths [Constr. (8)].

Penalty functions: we prioritize disjointness by penalizing path similarity. This constraint is defined by approximating a piecewise-linear function which grows with the number of paths that share a given physical link. The following expression generalizes the piecewise function:

$$\xi_{e^S,e^V} \geq w_K \cdot \sum_{p \in P} \mathcal{P}_{p,e^S,e^V} - c_K, \qquad \forall e^S \in E^S, \forall e^V \in e^V, \tag{9}$$

where constant K ($\leq |P|$) indicates how many paths of virtual link e^V overlap on physical link e^S, and $w_K = s^{K-1}$ ($s \geq 2$) weights this overlap. Constant c_K is used to select which function dominates the value of ξ_{e^S,e^V} for each sharing scenario. For example, suppose the maximum number of paths is set to four ($|P| = 4$) and a given virtual link x^V has three of its paths sharing a substrate link y^S, then ξ_{x^S,y^V} would equal $w_3 \cdot 3 - c_3$. Constant c_K is calculated by the following expression: $c_K = w_K \cdot (K-1) - [w_{K-1} \cdot (K-1) - c_{K-1}]$, $\forall e^S \in E^S, \forall e^V \in e^V, c_0 = 0, w_0 = 0$. Since all values are given beforehand, we use Equation (9) to create the set of constraints for each sharing scenario. For example, with 2 paths the following two constraints would be created:

$$\xi_{e^S,e^V} \geq \sum_{p \in P} \mathcal{P}_{p,e^S,e^V}, \qquad \forall e^S \in E^S, \forall e^V \in e^V;$$

$$\xi_{e^S,e^V} \geq s^1 \cdot \sum_{p \in P} \mathcal{P}_{p,e^S,e^V} - c_2, \qquad \forall e^S \in E^S, \forall e^V \in e^V.$$

We penalize link saturation (ϕ) by employing a classic traffic engineering function proposed by Fortz and Thorup [7]. This piecewise-linear function defines an exponential cost which is proportional to the substrate link capacity and utilization. We use this function by adding the same contraints and values defined by the authors.

Finally, to calculate differential delay we need to find the maximum and minimum delays of a given set of paths. As the order of paths is not important, we add the following constraint without altering the result of the optimization:

$$\sum_{\forall e^S \in E^S} \mathcal{P}_{p,e^S,e^V} \cdot d(e^S) \leq \sum_{\forall e^S \in E^S} \mathcal{P}_{p+1,e^S,e^V} \cdot d(e^S),$$
$$\forall p \in P' = \{1, \ldots, k\} \setminus \{k\}, \forall e^V \in E^V.$$

This constraint sorts paths in non-decreasing order of end-to-end propagation delay. Therefore, the differential delay of a virtual link δ_{e^V} can be calculated as follows:

$$\delta_{e^V} = \sum_{\forall e^S \in E^S} \mathcal{P}_{k,e^S,e^V} \cdot d(e^S) - \sum_{\forall e^S \in E^S} \mathcal{P}_{1,e^S,e^V} \cdot d(e^S),$$
$$\forall e^V \in E^V : P = \{1, \ldots, k\}.$$

Avoiding cycles: the previous set of constraints does not prevent cycles in the network. To deal with this problem, we create the auxiliary variable $\mathcal{H}_{p,n^S,e^V}$ and the following constraint to count the number of hops:

$$\mathcal{H}_{p,n^S,e^V} - |E^S| + |E^S| \cdot \mathcal{P}_{p,e^S,e^V} \leq \mathcal{H}_{p,m^S,e^V} + 1$$
$$\leq \mathcal{H}_{p,n^S,e^V} + |E^S| - |E^S| \cdot \mathcal{P}_{p,e^S,e^V}, \tag{10}$$
$$\forall p \in P, \forall e^S = (m^S, n^S) \in E^S, \forall e^V \in E^V.$$

This constraint can be read as follows: "for each node connected to links selected by $\mathcal{P}$, `next-hop = previous-hop + 1`". Thus, solutions with cycles are naturally discarded by the Mixed-Integer Program since `next-hop` ($\mathcal{H}_{p,n^S,e^V}$) and `previous-hop` ($\mathcal{H}_{p,m^S,e^V}$) would try to mutually increment, counting towards infinity.

4.2 Opportunistic Capacity Recovery

A key contribution of this paper is what we call **opportunistic recovery**. Instead of allocating backup resources, the capacity compromised by a successful DoS attack is reallocated over the available bandwidth along the set of unaffected paths. These paths can be any path that was previously selected to allocate a virtual link and remained operational after an attack. From this point onwards, we will refer to such paths as *active paths*.

Whenever a physical link becomes inaccessible, one or more paths of a virtual link may disappear. If the virtual link has no active path left, it means it was fully compromised. Otherwise, one or more active paths remain available, and any spare bandwidth remaining on these paths can be used to restore the capacity of the virtual link. The formulation of this strategy is given as follows:

Variable:
- $\mathcal{Q}_{p,e^V} \in \mathbb{R}$ within $[0,1]$: the fraction of capacity each virtual link $e^V \in \dot{E}^V$ reallocates to path p.

Input:
- $\hat{E}^S \subset E^S$: substrate links that remain available after a disruption event;
- $\dot{E}^V \subset E^V$: virtual links affected by the disruption;
- P_{e^V}: active paths of virtual link e^V after disruption;
- F_{e^V}: compromised capacity of virtual link e^V;
- $P_{p,e^S,e^V} \in \mathbb{B}$: indicates if physical link e^S is being used by active path p on virtual link e^V.

Objective: *maximize bandwidth capacity recovered after disruption events.* Therefore, function

$$\max T = \sum_{\forall e^V \in E^V} \sum_{\forall p \in P_{e^V}} \mathcal{Q}_{p,e^V}, \tag{11}$$

attempts to allocate all spare bandwidth available over the set of active paths.

Constraints:

$$\sum_{\forall p \in P_{e^V}} \mathcal{Q}_{p,e^V} \leq F_{e^V}, \qquad \forall e^V \in \dot{E}^V; \tag{12}$$

$$\sum_{\forall e^V \in E^V} \sum_{\forall p \in P_{e^V}} P_{p,e^S,e^V} \cdot \mathcal{Q}_{p,e^V} \cdot b(e^V) \leq b(e^S), \qquad \forall e^S \in \hat{E}^S. \tag{13}$$

The first constraint [Constr. (12)] guarantees that the amount of bandwidth reallocated cannot be greater than the amount compromised by an attack. The second constraint [Constr. (13)] ensures that the allocated bandwidth remains within the available capacity of physical links.

5. EVALUATION

The strategies described in the previous section have two main roles: (i) to *prevent* against the impact of attacks by allocating, efficiently, virtual links into multiple paths; and (ii) to *react* against these attacks, whenever possible, by reallocating compromised capacity over a set of active paths. This section evaluates the proposed approach by quantifying the impact of attacks over the affected virtual links – with and without our strategies – as well as the amount of accepted requests. Further, we evaluate the time needed to optimally allocate virtual network requests.

The mathematical formulation described in this paper was implemented on CPLEX 12.3. All experiments were executed on an Intel i7 with 8 cores of 2.93 GHz and 8 GB of RAM.

5.1 Evaluation Settings

The time of our experiments is organized in rounds. Each round can receive one or more virtual network requests and execute the allocation algorithm. Furthermore, the recovery algorithm is executed whenever an attack occurs. The round only finishes after concluding all steps of the allocation and/or recovery algorithms.

The network topologies were synthetically generated using BRITE with the Barabási-Albert (BA-2) network model. Next, we present the network and workload configurations used in this paper, which are in line with those in [6, 4].

Substrate. The physical network is composed of 30 nodes and 114 links on a 60x60 grid. The CPU and bandwidth capacity of nodes and links are uniformly distributed within [50,100]. The propagation delay of links is directly proportional to the distance between nodes, as generated by BRITE, normalized to the greatest value (i.e., within (0,1]).

Virtual networks. The capacity of nodes and links is uniformly distributed within [5,20] and [50,100], respectively. These values were chosen so that a single attribute would not be the main factor of impact on the experiments. The number of nodes on a virtual network request is uniformly distributed within 2 and 4. The use of such low values was necessary to perform the set of experiments as the allocation phase consumes too much CPU time. This limitation is discussed at the end of Section 5.2.

Workload. Our workload is composed of virtual network requests and denial of service attacks. The arrival rate of each request is given by a Poisson process with an average of 7 requests per 100 rounds. The duration of each request is assumed to be geometrically distributed with an average of 1000 rounds. The attacks are modeled by a Poisson process with mean of 1 attack per 100 rounds and duration of 10 rounds. As described earlier (§3.1), each attack is launched against the physical node that has the greatest bandwidth allocation on its links.

Experiments where executed during 5000 rounds. Thus, each experiment generated, approximately, 350 requests and 50 attacks. Finally, CPLEX gap to optimally was set to 1%, so solutions could be found in feasible time.

5.2 Evaluation Results

The proposed strategies help mitigating DoS attacks by preventing bandwidth loss. Fig. 1 depicts the effect of physical network disruptions with and without the use of our strategies. In all scenarios, w_Ξ, w_Φ e w_Δ were fixed in 1/3. Axis x and y represent, respectively, the number of paths to be used when embedding a virtual link and the average bandwidth loss after the attacks; therefore, the lower the bar, the better it is. Intuitively, increasing the number of paths should make the network more resilient to attacks.

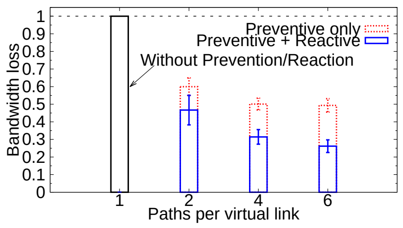


Figure 1: Bandwidth loss and bandwidth recovery on scenarios with 1, 2, 4 and 6 paths per virtual link, and $w_\Xi = w_\Phi = w_\Delta = 1/3$.

Our baseline have only one path per virtual link. In this case, when an attack occurs, all allocated bandwidth is lost and the virtual links become inoperable. The increase in the number of paths per virtual link allows a higher level of protection, since the proposed strategies work together against attacks. In Fig. 1, the preventive strategy is presented by dotted-lines (red bars), while the behavior of both strategies is presented by full lines (blue bars). Observe that, in the scenario with 6 paths per virtual link, the preventive strategy can protect approximately 50% of the capacity. As expected, results are improved when combining both strategies. Fig. 1 shows that the second strategy is able to recover part of the compromised capacity, so that we can reduce this loss to approximately 25%. Moreover, although the efficiency of our strategies increases with the number of paths, this behavior is nonlinear. This happens because the substrate network has a limited amount of disjoint paths available to embed each virtual link. We provide further evidence of this behavior on our next experiment.

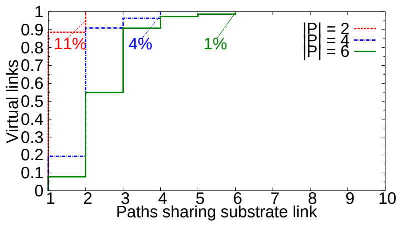
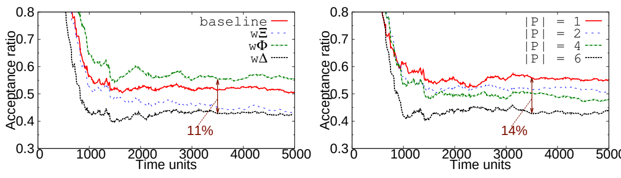


Figure 2: Amount of paths that use the same substrate link.

Multipath allocation reduces the severity of attacks to critical targets. We define as "critical target" a substrate resource, namely a node or link, that concentrates the full capacity of a virtual link by a circumstance of the physical network and/or the allocation procedure. According to this definition, if an attack on a critical target succeeds, one or more virtual links would be fully compromised. The CDF presented in Fig. 2 shows the percentage of virtual links vulnerable to such attacks. The X axis indicates how many paths of the same virtual link utilize a given physical link (i.e., 1 means that only one path uses the substrate link and there is no sharing). When considering 2 paths per virtual link ($|P| = 2$), approximately 11% of virtual links have paths that share the same physical link. By increasing the number of paths, the percentage of vulnerable virtual links decreases to 4% ($|P| = 4$), and then to 1% ($|P| = 6$). On the other hand, it is also possible to see that the percentage of virtual links with disjoint paths also decreases. The importance of disjoint paths is that if one or more paths share the same underling physical link, the efficiency of the opportunistic recovery strategy is limited by the capacity of the shared link. When virtual links are embedded on 2 paths, 89% of these virtual links have disjoint paths (i.e., do not share any substrate link). This decreases to 7% when the number of paths per virtual link increases to 6. This evidence sustains our earlier observation that the amount of disjoint paths on the substrate affects the performance of our approach.

The allocation cost is proportional to the level of resilience against attacks. Fig. 3 illustrates the acceptance ratio through time in scenarios using different parameters. These graphs should be read as follows. At first ($0 \sim 900$), none of the arriving virtual network requests are rejected since the substrate has sufficient resources to embed all of them. As time passes and more requests arrive, resources become scarce



(a) $|P| = 2$ (b) $w_\Xi = w_\Phi = w_\Delta = 1/3$

Figure 3: Impact on acceptance ratio when varying (a) parameters w_Ξ, w_Φ, and w_Δ and (b) the number of paths per virtual link.

($900 \sim 2000$) which results in some requests being rejected. Finally, the allocation process stabilizes ($2000 \sim \infty$) when the amount resources being freed is approximately the amount demanded by new requests, and acceptance ratio stays between 40 and 60%. Fig. 3(a) shows how prioritizing one factor over the others can affect the acceptance ratio of the substrate network. Our baseline curve has $w_\Xi = w_\Phi = w_\Delta = 1/3$. The other three curves have the main factor set to 0.998 and the other two to 0.001. In practice, this is the same as replacing our 3-factor objective function by one that has a single factor. When we *prioritize path disjointness* (w_Ξ), the amount of allocated substrate resources is increased and the acceptance ratio decreases. This effect happens because the allocation algorithm tends to choose longer paths to avoid sharing substrate links. Prioritizing *less saturated links* (w_Φ) or *differential delay minimization* (w_Δ) results in the best and worst acceptance ratios, respectively (with a reduction of 11% in acceptance ratio). On the one hand, increasing w_Φ helps to avoid bottlenecks, which tends to leave free access to substrate nodes. On the other, increasing w_Δ prioritizes paths with similar delays, which means longer paths with similar delays may be chosen over shorter paths with more distinct delays.

Fig. 3(b) shows the efficiency of our allocation algorithm through time when varying the number of paths per virtual link. Although increasing the number of paths provides better granularity, it tends to result in worse allocations. In the worst case, when considering 6 paths, acceptance ratio drops 14%. This effect happens because prioritizing disjointness (Ξ) and differential delay minimization (Δ) does not take any effect when employing a single path. However, the relative importance of both metrics, and their collateral effect on allocation, increases with the number of paths.

The difficulty of this problem increases exponentially, thus justifying the study of heuristics. Previous work proposes heuristics to deal with the virtual network embedding problem [6, 4]. Its similarity to the multiway separator problem [1] is a strong evidence that this problem is NP-Hard. Our virtual network embedding approach requires few milliseconds to optimally solve a 2-node Virtual Network Request (VNR), yet it demands approximately 250s to optimally solve 4-node VNR. Therefore, the natural evolution of this work consists in defining heuristics (e.g., Greedy algorithms) and meta-heuristics (e.g., Simulated Annealing) to improve the scalability of our approach.

6. CONCLUSIONS

One of the potential advantages in network virtualization is the use of isolation to withstand network attacks. Nonetheless, in this paper we argued that virtual networks are still vulnerable to DoS attacks performed by compromising physical substrate resources. Moreover, we presented a novel approach to protect against such attacks. Among our contributions, we highlight that, unlike previous work, our allocation approach does not need to set aside backup resources or recalculate end-to-end paths to reallocate virtual links. Rather, we divide our approach in two complementary strategies. The first protects against attacks by embedding virtual links into multiple, preferably disjoint paths. The second attempts to recover any compromised capacity over the set of active paths (*i.e.*, those unaffected by the attack).

Our solution improves the protection against DoS attacks at the cost of a lower acceptance ratio. Additionally, this improvement gets proportionally smaller as resilience is prioritized over allocation. Therefore, it is possible to adapt the solution to the requirements of the environment. For instance, one can chose to get the best possible resilience by sacrificing the efficiency of allocation, or else adopt a more conservative approach and still mitigate disruptions.

Regarding future work, we envision heuristics that allow our approach to be used on larger scenarios. Moreover, we are investigating the incorporation of migration to the recovery process, making networks more robust against attacks.

Acknowledgment

This work has been supported by European Commission's FP7 and CNPq Research Agency (Project SecFuNet, FP7-ICT-2011-EU-Brazil), RNP's CTIC program (Project ReVir), as well as PRONEM/FAPERGS/CNPq (Project NPRV).

7. REFERENCES

[1] D. G. Andersen. Theoretical approaches to node assignment, 2002. Available online: `http://goo.gl/rrnHz`.

[2] Belbekkouche et al. Resource discovery and allocation in network virtualization. *Comm. Surv. Tut., IEEE*, PP(99):1–15, 2012.

[3] Chen et al. Resilient virtual network service provision in network virtualization environments. In *Proc. ICPADS, IEEE*, pages 51–58, 2010.

[4] Cheng et al. Virtual network embedding through topology awareness and optimization. *Comp. Netw.*, 56(6):1797–1813, 2012.

[5] N. M. K. Chowdhury and R. Boutaba. A survey of network virtualization. *Comp. Netw.*, 54(5):862–876, 2010.

[6] Chowdhury et al. Virtual network embedding with coordinated node and link mapping. In *Proc. INFOCOM, IEEE*, pages 783–791, 2009.

[7] B. Fortz and M. Thorup. Increasing internet capacity using local search. *Comp. Optim. and Applic.*, 29:13–48, 2004.

[8] Guo et al. Shared backup network provision for virtual network embedding. In *Proc. ICC, IEEE*, pages 1–5, 2011.

[9] Houidi et al. Adaptive virtual network provisioning. In *2nd SIGCOMM VISA workshop*, pages 41–48. ACM, 2010.

[10] Huang et al. Survivable multipath provisioning with differential delay constraint in telecom mesh networks. *Trans. on Netw., IEEE/ACM*, 19(3):657–669, 2011.

[11] Khan et al. Network virtualization: a hypervisor for the internet? *Comm. Mag., IEEE*, 50(1):136–143, 2012.

[12] Rahman et al. Survivable virtual network embedding. In M. e. Crovella, editor, *NETWORKING 2010*, volume 6091 of *LNCS*, pages 40–52. Springer Berlin / Heidelberg, 2010.

[13] Yeow et al. Designing and embedding reliable virtual infrastructures. In *2nd SIGCOMM VISA workshop*, pages 33–40. ACM, 2010.

[14] Yu et al. Cost efficient design of survivable virtual infrastructure to recover from facility node failures. In *Proc. ICC, IEEE*, pages 1–6, 2011.

[15] Zhang et al. Reliable adaptive multipath provisioning with bandwidth and differential delay constraints. In *Proc. INFOCOM, IEEE*, pages 2178–2186, 2010.

APPENDIX E - SBSEG ARTICLE

- Title: Redes Virtuais Seguras: Uma Nova Abordagem de Mapeamento para Proteger contra Ataques de Disrupção na Rede Física
- Conference: XII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg 2012)
- URL: `http://sbseg2012.ppgia.pucpr.br/`
- Date: 19 – 22, november, 2012
- Location: Curitiba, PR, Brazil

Redes Virtuais Seguras: Uma Nova Abordagem de Mapeamento para Proteger contra Ataques de Disrupção na Rede Física

Rodrigo R Oliveira[1], **Leonardo R Bays**[1], **Daniel S Marcon**[1], **Miguel C Neves**[1], **Luciana S Buriol**[1], **Luciano P Gaspary**[1], **Marinho P Barcellos**[1]

[1]Universidade Federal do Rio Grande do Sul (UFRGS),
Instituto de Informática

{ruas.oliveira, lrbays, daniel.stefani, mcneves, buriol, paschoal, marinho}@inf.ufrgs.br

Resumo. *Na virtualização de redes, roteadores e enlaces virtuais são alocados sobre uma infraestrutura de rede física. Tal característica representa uma vulnerabilidade a ataques de negação de serviço na rede física, visto que um único dispositivo físico comprometido afeta todos os virtuais sobrepostos. Trabalhos anteriores propõem a reserva de recursos sobressalentes. Apesar de funcional, esse tipo de solução agrega custo ao provedor da rede física. Neste artigo, propõe-se uma abordagem para alocação de redes virtuais que explora o compromisso entre a resiliência a ataques e a eficiência na utilização de recursos. A abordagem é separada em duas estratégias, uma preventiva e uma reativa. A primeira aloca enlaces virtuais em múltiplos caminhos do substrato, enquanto a segunda tenta recuperar a capacidade dos enlaces virtuais afetada por um ataque de negação de serviço subjacente. Ambas as estratégias são formuladas como problemas de otimização. Resultados numéricos demonstram o nível de resiliência a ataques propiciado pela abordagem e o baixo custo decorrente da mesma.*

Abstract. *In network virtualization, virtual routers and links are embedded into a physical network infrastructure. Such characteristic represents a vulnerability as a compromised physical device affects all the overlaid virtual ones. Previous work proposes setting aside backup resources. Although effective, this solution aggregates cost to infrastructure providers. In this paper, we propose a virtual network allocation approach which explores the trade-off between resilience to attacks, and efficiency in resource utilization. Our approach is composed of two strategies, one preventive and the other reactive. The former allocates virtual links into multiple substrate paths, while the latter attempts to recover the capacity of virtual links affected by an underlying DoS attack. Both strategies are formulated as optimization problems. Numerical results show the level of resilience to attacks and the low cost demanded by our approach.*

1. Introdução

A virtualização de redes consiste na instanciação de múltiplas redes virtuais sobre um mesmo substrato de rede física. Esse paradigma permite o isolamento entre redes virtuais, propiciando a independência funcional entre as mesmas. Uma de suas mais promissoras vantagens é a capacidade de limitar o escopo de ataques, através da organização de uma infraestrutura em múltiplas redes virtuais, isolando o tráfego das mesmas [Khan *et al* 2012]. De maneira geral, as redes virtuais são requisitadas por diferentes "Provedores de Serviço" (*Service Providers* – SP), e alocadas sob-demanda nos recursos físicos de "Provedores de Infraestrutura" (*Infrastructure Provider* – InP) [Belbekkouche *et al* 2012]. A alocação significa designar uma parcela dos nós e enlaces físicos do substrato aos nós e enlaces virtuais sobrepostos. Dessa forma, as redes virtuais compartilham os recursos dos dispositivos da rede física [Chowdhury and Boutaba 2010]. Tal particularidade tende a aumentar a dependência de certos recursos físicos. Consequentemente, um atacante pode lançar ataques de negação de serviço às redes virtuais, comprometendo para isso roteadores e/ou enlaces do substrato físico subjacente. Especificamente, caso determinado enlace do substrato seja comprometido, todos os enlaces virtuais sobrepostos (ou seja, alocados neste) serão afetados[1].

Para lidar com esse problema, a literatura propõe estratégias que reservam determinada quantidade de recursos do substrato como sobressalentes. Essa reserva pode ser uma fração de todos os recursos da rede [Rahman *et al* 2010] ou a alocação extra de nós disjuntos [Yeow *et al* 2010], caminhos disjuntos [Chen *et al* 2010,Guo *et al* 2011] ou ambos [Yu *et al* 2011]. Apesar de eficaz, essa abordagem pode ser muito custosa ao Provedor de Infraestrutura visto que os recursos sobressalentes (a) ficam ociosos caso não haja disrupções e (b) a ocupação extra pode limitar a alocação de novas redes. Dessa forma, a aplicabilidade dessas estratégias pode ficar limitada a nichos de mercado, deixando pragmaticamente as futuras redes virtuais vulneráveis a ataques de negação de serviço.

Neste artigo é proposta uma abordagem para aumentar a resiliência das redes virtuais dos Provedores de Serviço contra ataques de DoS, sem prejudicar a alocação dos recursos físicos do Provedor de Infraestrutura. A abordagem pode ser dividida em duas estratégias complementares, formuladas como problemas de otimização linear inteira. Primeiro utiliza-se uma estratégia **preventiva** para mitigar o impacto de um ataque. Nessa estratégia os enlaces virtuais são alocados em *múltiplos caminhos preferencialmente disjuntos* para impedir que toda a capacidade dos enlaces virtuais seja afetada quando um enlace físico subjacente é comprometido. Após a ocorrência de um ataque, utiliza-se uma estratégia **reativa** para recuperar, *parcial ou integralmente*, a capacidade perdida nos enlaces virtuais afetados.

A abordagem proposta acarreta dois problemas principais: (i) a alocação com *priorização por caminhos disjuntos* levar à sobrecarga de determinados enlaces físicos, deixando pouca ou nenhuma capacidade disponível, potencialmente impedindo a alocação de novas redes que dependem daquele enlace; e (ii) o uso de múltiplos caminhos, com atrasos de propagação potencialmente distintos, causa o problema da entrega de pacotes fora de ordem. Para mitigar o primeiro problema propõe-se o balanceamento de carga entre os enlaces do substrato, evitando saturar os caminhos escolhidos. Para tratar do segundo, alocações são feitas buscando minimizar a diferença dos atrasos entre caminhos, em linha com a literatura [Zhang *et al* 2010]. Tais escolhas serão discutidas na Seção 3.

As principais contribuições deste artigo são sumarizadas a seguir:

- **Modelagem dos três fatores do problema, combinando relativamente o impacto de caminhos disjuntos, balanceamento de carga e alocação eficiente de recursos (Seção 3).** Obtém-se assim um equilíbrio entre a resiliência das redes virtuais e a eficiência de alocação de recursos.

[1]Este artigo concentra-se em ataques, mas as soluções propostas também aplicam-se a falhas.

- **Formulação das estratégias como problemas de otimização (Seção 4).** As duas estratégias propostas neste artigo, a *alocação em múltiplos caminhos* e a *redistribuição oportunística de carga*, são formuladas em programação linear inteira. As formulações geram o *resultado ótimo* de acordo com a priorização desejada.
- **Análise do compromisso gerado pela solução (Seção 5).** Experimentos demonstram que a abordagem proposta pode reduzir substancialmente o impacto de ataques de negação de serviço, porém ao custo de uma taxa de aceitação menor. Este artigo responde a seguinte questão: *como sobreviver à disrupção nas redes virtuais aumentando minimamente o custo?*

O restante do artigo está organizado da seguinte forma: a Seção 2 descreve trabalhos relacionados. A Seção 3 define o modelo de ataque e os principais conceitos relacionados à alocação de redes virtuais, ao atraso diferencial e ao objetivo. A Seção 4 apresenta as formulações em programação linear inteira das duas estratégias desenvolvidas. Por fim, os resultados da avaliação dos modelos são discutidos na Seção 5, e a Seção 6 conclui o trabalho.

2. Trabalhos relacionados

O presente trabalho se concentra em ataques de negação de serviço a redes virtualizadas através do comprometimento de enlaces físicos. O problema pode ser relacionado à pesquisa em sobrevivência em redes. Embora tal assunto não seja novo, apenas recentemente a comunidade científica começou a tratar dessa questão no contexto de redes virtuais. Esta seção discute os trabalhos relacionados, tanto em um contexto mais amplo como especificamente em redes virtualizadas.

A principal diferença entre o tratamento de disrupções nas redes virtuais para as redes convencionais (p.ex., redes ópticas, MPLS ou no planejamento de tráfego IP) é que na virtualização de redes o tráfego é gerado pelas redes virtuais, as quais podem ser instaladas e retiradas da rede sob-demanda. Em contraste, nas redes convencionais o tráfego entre os nós é conhecido e considerado como um parâmetro do modelo [Medhi 2006]. Ademais, as redes virtuais devem garantir a topologia, ou seja, os enlaces virtuais devem permanecer ativos. Por outro lado, nas redes convencionais basta manter a conectividade [Belbekkouche *et al* 2012].

De maneira geral, a resiliência das redes virtuais é provida através da alocação de recursos sobressalentes. [Rahman *et al* 2010], [Chen *et al* 2010] e [Guo *et al* 2011] estudam a *disrupção de enlaces do substrato físico*. [Rahman *et al* 2010] reservam determinado percentual de banda de cada enlace do substrato para proteger enlaces virtuais contra disrupções. Os autores calculam "k-caminhos mais curtos" entre os nós de cada enlace físico, resultando em micro-desvios. Caso um enlace físico seja interrompido, o fluxo é redirecionado pelos recursos sobressalentes desses desvios. Por sua vez, [Chen *et al* 2010] e [Guo *et al* 2011] utilizam a alocação de caminhos disjuntos, entre os mapeamentos primário e sobressalente, como estratégia para prover resiliência a disrupções.

Por outro lado, [Yeow *et al* 2010] estudam a *negação de serviço em nós da rede física*. Os autores consideram a existência de nós virtuais críticos. Para proteger esses nós, os autores oferecem redundância por meio da reserva de nós físicos sobressalentes. Apesar de não considerar disrupções em enlaces físicos, a estratégia também reserva caminhos sobressalentes para manter a topologia. Isso pode resultar em um número potencialmente grande de enlaces físicos alocados como sobressalentes, visto que cada nó redundante precisa manter a conectividade dos nós críticos mapeados.

[Yu *et al* 2011] consideram eventos de *disrupção regionalizada* no substrato. Tal evento ocorre quando uma única ação compromete múltiplos dispositivos de uma mesma localidade. Para sobreviver a esses eventos, cada nó virtual é replicado em regiões distintas. Além disso, os caminhos entre esses nós sobressalentes não podem passar pelas mesmas regiões dos primários. Essa solução, no entanto, pode restringir bastante o espaço de soluções.

As soluções anteriores fazem uso da reserva de recursos sobressalentes para oferecer resiliência. Tal abordagem pode ser muito custosa ao substrato pois os recursos sobressalentes limitam a alocação de novas redes e ficam ociosos caso não haja disrupções. Em contraste, [Houidi *et al* 2010] dispensam recursos sobressalentes e propõem o uso de multiagentes para lidar com *negação de serviço em nós e enlaces físicos*. Na proposta, os roteadores físicos devem implementar um agente que se comunica com os demais para realocar nós e enlaces virtuais de dispositivos sob ataque. Apesar de não alocar recursos sobressalentes, a estratégia precisa recalcular novos caminhos para os enlaces virtuais afetados. Essa estratégia demanda um período de convergência e pode deixar as redes virtuais inoperantes durante o mesmo.

A proposta apresentada neste artigo difere das anteriores pelas seguintes razões: i) ela não considera a alocação de recursos sobressalentes; ii) a recuperação de capacidade dos enlaces virtuais não necessita recalcular os caminhos fim-a-fim, evitando o custo de realizar esta computação na recuperação; iii) as restrições de resiliência são relaxadas de forma a oferecer uma estratégia menos custosa ao provedor de infraestrutura.

3. Definições Preliminares

3.1. Modelo de Ameaça

A principal ameaça considerada neste artigo é a disrupção de comunicação entre os nós das redes virtuais. Essa disrupção pode ocorrer por meio de ataques de negação de serviço nos roteadores ou enlaces da rede física.

Ataques a roteadores e enlaces físicos. Um ataque de negação de serviço pode ser feito por meio do acesso físico ao dispositivo, ou exploração de alguma vulnerabilidade. No primeiro caso, um atacante obtém a localização de uma fibra óptica ou roteador e interrompe seu funcionamento (p.ex., cortando a fibra ou desligando a energia). No segundo caso, o atacante explora alguma vulnerabilidade no protocolo ou *software* de controle para comprometer algum dispositivo.

Premissas. O escopo deste trabalho é delimitado pelas seguintes premissas:

- Considera-se que um atacante pode obter informação sobre a utilização de determinado dispositivo. Dessa forma, o ataque tem como alvo o nó ou enlace com maior utilização, de forma a causar o maior impacto.
- Há isolamento entre as redes virtuais. Portanto, uma rede virtual não pode interferir no funcionamento de outra ao tentar consumir mais recursos do que requisitou.
- A disrupção de um nó físico pode ser reduzida a disrupções em múltiplos enlaces físicos do substrato.

3.2. Alocação de Redes Virtuais

A alocação de redes virtuais consiste em alocar nós e enlaces virtuais em nós e caminhos do substrato físico, respectivamente. Cada nó virtual é alocado em um nó físico distinto e cada enlace virtual é alocado em *um caminho* (ou *subconjunto dos caminhos*) do substrato. Ademais, as requisições de redes virtuais são enviadas ao substrato físico *sob-demanda*, ou seja, não é possível determinar com antecedência quais redes virtuais deverão ser alocadas. Essas e as demais definições estão em linha com trabalhos anteriores de alocação [Chowdhury *et al* 2009, Cheng *et al* 2012].

Substrato. Representado por um grafo dirigido $\mathsf{G}^{\mathsf{S}}(\mathsf{N}^{\mathsf{S}}, \mathsf{E}^{\mathsf{S}}, \mathsf{A}^{\mathsf{S}})$, onde N^{S} é o conjunto de nós, E^{S} o conjunto de enlaces e A^{S} o conjunto de atributos dos nós e enlaces. Neste artigo, são considerados os seguintes atributos: *capacidade computacional* dos nós, *largura de banda* e *atraso de propagação* dos enlaces.

Requisição de rede virtual. Cada requisição é definida por um grafo dirigido $\mathsf{G}^{\mathsf{V}}(\mathsf{N}^{\mathsf{V}}, \mathsf{E}^{\mathsf{V}}, \mathsf{A}^{\mathsf{V}})$, onde N^{V}, E^{V} e A^{V} são conjuntos similares aos supracitados.

3.3. Atraso Diferencial

A divisão do tráfego de dados nem sempre pode ser realizada por caminhos com mesmo atraso fim-a-fim. Particularmente, a utilização de caminhos com diferenças de atraso pode intensificar a entrega de pacotes fora de ordem. Esse problema (definido como *problema do atraso diferencial*) acaba impactando o funcionamento geral dos protocolos de entrega confiável (p.ex., TCP). De maneira geral, pode-se realizar a multiplexação de três formas [He and Rexford 2008]: por fluxo, por rajadas (*flowlets*) ou por pacotes. A primeira forma possui baixa granularidade, visto que cada fluxo está limitado à capacidade de um único caminho. A segunda permite que o mesmo fluxo modifique o caminho de tempos em tempos, o que pode ser ineficiente em fluxos que possuam comportamento contínuo. A terceira alternativa possui a melhor granularidade, permitindo que os fluxos agreguem toda a capacidade dos caminhos.

Neste trabalho optou-se por esta última alternativa, a qual permite aos enlaces das redes virtuais utilizar toda a capacidade requisitada. Apesar dessa grande vantagem, a multiplexação por pacotes pode impor um custo adicional ao destino, seja aos roteadores de borda ou aos hospedeiros. Tal custo, em memória e processamento, é diretamente proporcional às diferenças no atraso entre pacotes. Portanto, de forma similar a trabalhos anteriores [Zhang *et al* 2010], define-se como subobjetivo de otimização a minimização do atraso diferencial.

3.4. Objetivos de Otimização

A formulação apresentada neste trabalho possui por objetivo oferecer resiliência às redes virtuais, levando em consideração o custo gerado ao substrato físico. Para isso, o modelo proposto considera três fatores principais.

Priorização por multi-caminhos disjuntos. A alocação em múltiplos caminhos permite que um enlace virtual possa resistir a uma disrupção na rede física. No entanto, tal característica só é eficiente se os caminhos selecionados são suficientemente diferentes, caso contrário, uma disrupção em um enlace físico poderia afetar a maior parte dos caminhos utilizados por determinado enlace virtual. No pior caso, uma disrupção pode afetar completamente um ou mais enlaces virtuais. Para lidar com esse problema, define-se a função $\Xi = \sum_{\forall e^{\mathsf{S}} \in \mathsf{E}^{\mathsf{S}}} \sum_{\forall e^{\mathsf{V}} \in \mathsf{E}^{\mathsf{V}}} \xi_{e^{\mathsf{S}}, e^{\mathsf{V}}}$, a qual descreve a penalidade total dada pela similaridade entre os caminhos.

Balanceamento de carga. Conforme mencionado anteriormente, a alocação de redes virtuais é feita sob-demanda. Com isso, a alocação a longo prazo pode causar a saturação de um subconjunto dos enlaces da rede física. Essa saturação, por sua vez, pode acabar gerando rejeição de novas requisições que possuam preferência por determinado (subconjunto de) enlace(s) físico(s). O balanceamento de carga é utilizado para evitar essa saturação de enlaces físicos. Para isso, define-se a função $\Phi = \sum_{\forall e^s \in E^s} \phi_{e^s}$, expressando a penalização pela saturação do substrato.

Minimização do atraso diferencial. Supondo P_{e^v} o conjunto de caminhos selecionados para alocar determinado enlace virtual, D_p como o atraso no caminho p e $\delta_{e^v} = \max_{\forall p,q \in P_{e^V}}(\mid D_p - D_q \mid)$ como a diferença entre o maior e menor atraso no conjunto de caminhos de um enlace virtual. A função $\Delta = \sum_{\forall e^v \in E^v} \delta_{e^v}$ penaliza o atraso diferencial acumulado de todos os enlaces virtuais.

4. Estratégias para Mitigar a Disrupção na Rede Física

Nesta seção são definidas as duas estratégias, preventiva e reativa, para oferecer resiliência contra ataques de negação de serviço às redes virtuais. A primeira estratégia consiste no mapeamento dos enlaces virtuais em múltiplos caminhos, de forma que o comprometimento de um ou mais enlaces físicos não afete completamente os enlaces virtuais. A segunda estratégia é utilizada quando um ataque obtém sucesso em derrubar um enlace. Essa estratégia tenta realocar a capacidade perdida nos caminhos que não foram afetados.

4.1. Mapeamento de Redes Virtuais considerando Resiliência

A estratégia utilizada para prover resiliência visa mapear cada enlace virtual das requisições em conjuntos de caminhos físicos, tal que: (i) toda a capacidade do enlace virtual seja distribuída por esses caminhos e (ii) tais caminhos tenham pouca similaridade entre si. A formulação da estratégia é dada conforme segue:

Variáveis:

- $\mathcal{X}_{n^s,n^v} \in \mathbb{B}$: indica que o nó $n^S \in N^S$ do substrato físico está sendo utilizado para mapear o nó virtual $n^V \in N^V$.
- $\mathcal{P}_{p,e^s,e^v} \in \mathbb{B}$: indica que o caminho p utiliza o enlace do substrato $e^S \in E^S$ para mapear o enlace virtual $e^V \in E^V$.
- $\mathcal{F}_{p,e^s,e^v} \in \mathbb{R}$ no intervalo $[0;1]$: indica a parcela do fluxo do enlace virtual $e^V \in E^V$ a ser alocada no enlace físico e^S para o caminho p.

Entrada:

- $c(.) \in A^*$: poder computacional de determinado nó (físico ou virtual).
- $b(.) \in A^*$: largura de banda de determinado enlace (físico ou virtual).
- $d(.) \in A^S$: atraso de determinado enlace físico.
- $P, |P|$: representam, respectivamente, o conjunto de índices dos caminhos e número máximo de caminhos por enlace virtual.

Objetivo: *minimizar o impacto causado pelos três fatores do problema*,

$$\min Z = w_\Xi \cdot \Xi + w_\Phi \cdot \Phi + w_\Delta \cdot \Delta \tag{1}$$

onde w_Φ, w_Ξ e w_Δ são pesos que representam a importância relativa de cada parte da função objetivo. A definição das funções de penalização que compõem Ξ, Φ e Δ serão apresentadas após as restrições do modelo.

Restrições:

$$\sum_{\forall n^S \in N^S} \mathcal{X}_{n^S,n^V} = 1 \qquad \forall n^V \in N^V \tag{2}$$

$$\sum_{\forall n^V \in N^V} \mathcal{X}_{n^S,n^V} \leq 1 \qquad \forall n^S \in N^S \tag{3}$$

$$\sum_{\forall n^V \in N^V} \mathcal{X}_{n^S,n^V} \cdot c(n^V) \leq c(n^S) \qquad \forall n^S \in N^S \tag{4}$$

$$\sum_{\forall p \in P} \sum_{\forall e^V \in E^V} \mathcal{F}_{p,e^S,e^V} \cdot b(e^V) \leq b(e^S) \qquad \forall e^S \in E^S \tag{5}$$

$$\sum_{\forall b^S \in N^S:(a^S,b^S) \in E^S} \mathcal{P}_{p,(a^S,b^S),e^V} - \sum_{\forall b^S \in N^S:(b^S,a^S) \in E^S} \mathcal{P}_{p,(b^S,a^S),e^V} = \mathcal{X}_{a^S,s^V} - \mathcal{X}_{a^S,t^V} \tag{6}$$

$$\forall p \in P, \forall a^S \in N^S, \forall e^V = (s^V, t^V) \in E^V$$

$$\sum_{\forall p \in P} \sum_{\forall b^S \in N^S:(a^S,b^S) \in E^S} \mathcal{F}_{p,(a^S,b^S),e^V} - \sum_{\forall p \in P} \sum_{\forall b^S \in N^S:(a^S,b^S) \in E^S} \mathcal{F}_{p,(b^S,a^S),e^V} = \mathcal{X}_{a^S,s^V} - \mathcal{X}_{a^S,t^V} \tag{7}$$

$$\forall a^S \in N^S, \forall e^V = (s^V, t^V) \in E^V$$

$$\mathcal{F}_{p,e^S,e^V} \leq \mathcal{P}_{p,e^S,e^V} \qquad \forall p \in P, \forall e^S \in E^S, \forall e^V \in E^V \tag{8}$$

As restrições (2) e (3) garantem respectivamente que todo nó virtual será alocado a exatamente um nó físico e que nós virtuais serão alocados em nós físicos diferentes. As restrições (4) e (5) asseguram que as capacidades dos nós e enlaces físicos serão respeitadas. A formação de um caminho fim-a-fim válido é representada pela restrição (6). Por fim, as duas últimas restrições definem a quantidade de banda em cada caminho. Elas asseguram a conservação de fluxo ao longo dos caminhos (7) e que o fluxo só será definido em enlaces físicos selecionados (8).

O conjunto de restrições anterior não previne a formação de laços na rede. Dessa forma, adiciona-se a variável auxiliar $\mathcal{H}_{p,a^S,e^V}$ e a restrição não linear $\mathcal{H}_{p,b^S,e^V} = \mathcal{P}_{p,(a^S,b^S),e^V} \cdot \mathcal{H}_{p,a^S,e^V} + \mathcal{P}_{p,(a^S,b^S),e^V}$. Essa restrição "conta o número de saltos" do roteador origem ao destino de cada caminho. Com isso, mapeamentos com laços serão naturalmente descartados pois não seria possível encontrar solução factível. A linearização da restrição é feita pela substituição da multiplicação de variáveis por uma variável auxiliar τ. Após, com o uso de técnicas padrões é possível atribuir relação entre as mesmas. Esse processo culminou nas seguintes restrições:

$$\mathcal{H}_{p,n^S,e^V} \leq \sum_{\substack{m^S \in N^S: \\ (m^S,n^S) \in E^S}} \sigma \cdot \mathcal{P}_{p,(m^S,n^S),e^V} \qquad \forall p \in P, \forall n^S \in N^S, \forall e^V \in E^V : \sigma \rightarrow \infty \tag{9}$$

$$\mathcal{H}_{p,n^S,e^V} = \sum_{\substack{m^S \in N^S: \\ (m^S,n^S) \in E^S}} \left(| E^S | \cdot \tau_{p,(m^S,n^S),e^V} + \mathcal{P}_{p,(m^S,n^S),e^V}\right) \qquad \forall p \in P, \forall n^S \in N^S, \forall e^V \in E^V \tag{10}$$

$$\tau_{p,e^S,e^V} \leq \frac{\mathcal{H}_{p,n^S,e^V}}{| E^S |} \qquad \forall p \in P, \forall e^S = (n^S, m^S) \in E^S, \forall e^V \in E^V \tag{11}$$

$$\tau_{p,e^S,e^V} \leq \mathcal{P}_{p,e^S,e^V} \qquad \forall p \in P, \forall e^S \in E^S, \forall e^V \in E^V \tag{12}$$

$$\tau_{p,e^S,e^V} \geq \frac{\mathcal{H}_{p,n^S,e^V}}{| E^S |} + \mathcal{P}_{p,e^S,e^V} - 1 \qquad \forall p \in P, \forall e^S = (n^S, m^S) \in E^S, \forall e^V \in E^V \tag{13}$$

$$\mathcal{H}_{p,n^S,e^V},\ \tau_{p,e^S,e^V} \geq 0 \qquad \forall p \in P, \forall n^S \in N^S, \forall e^S \in E^S, \forall e^V \in E^V \tag{14}$$

Funções de penalização: a priorização por caminhos disjuntos é dada pela penalização da similaridade de caminhos. Tal restrição é definida por uma aproximação linear de uma função que cresce exponencialmente com o compartilhamento de enlaces físicos. A expressão generalizada é dada pela seguinte formula:

$$\xi_{e^S,e^V} \geq w_K \cdot \sum_{p \in P} \mathcal{P}_{p,e^S,e^V} - c_K \qquad \forall e^S \in E^S, \forall e^V \in e^V \tag{15}$$

onde a constante K ($\leq |P|$) indica quantos caminhos do enlace virtual e^V utilizam o mesmo enlace físico e^S, e $w_K = s^{K-1}$ ($s \geq 2$) indica o peso dessa sobreposição. Por sua vez, c_K é dado pela seguinte equação: $c_K = w_K \cdot (K-1) - [w_{K-1} \cdot (K-1) - c_{K-1}], \quad \forall e^S \in E^S, \forall e^V \in e^V, c_0 = 0, w_0 = 0$. A expressão da Eq. (15) é utilizada para criar uma restrição para cada caminho. Portanto, no caso de 2 caminhos serão criadas duas restrições da seguinte forma:

$$\xi_{e^S,e^V} \geq \sum_{p \in P} \mathcal{P}_{p,e^S,e^V} \qquad \forall e^S \in E^S, \forall e^V \in e^V$$

$$\xi_{e^S,e^V} \geq s^1 \cdot \sum_{p \in P} \mathcal{P}_{p,e^S,e^V} - c_1 \qquad \forall e^S \in E^S, \forall e^V \in e^V$$

Para prover a penalidade na saturação de enlaces (ϕ) utilizou-se a função clássica de engenharia de tráfego proposta por [Fortz and Thorup 2004]. Tal função define um custo exponencial (com aproximação linear) proporcional à utilização do enlace físico e a capacidade do mesmo.

Por fim, para calcular o atraso diferencial, é preciso encontrar os atrasos mínimo e máximo ao longo dos caminhos. Como a ordem dos caminhos não é importante para a formulação, é possível adicionar a seguinte restrição sem alterar o resultado da otimização:

$$\sum_{\forall e^S \in E^S} \mathcal{P}_{p,e^S,e^V} \cdot d(e^S) \leq \sum_{\forall e^S \in E^S} \mathcal{P}_{p+1,e^S,e^V} \cdot d(e^S) \qquad \forall p \in P = \{1, \ldots, k\} \setminus \{k\}, \forall e^V \in E^V$$

a qual ordena os caminhos selecionados de menor a maior atraso. Logo, o atraso diferencial em um enlace virtual δ_{e^V} pode ser calculado pela seguinte expressão:

$$\delta_{e^V} = \sum_{\forall e^S \in E^S} \mathcal{P}_{k,e^S,e^V} \cdot d(e^S) - \sum_{\forall e^S \in E^S} \mathcal{P}_{1,e^S,e^V} \cdot d(e^S) \qquad \forall e^V \in E^V : P = \{1, \ldots, k\}$$

4.2. Recuperação Oportunística de Carga

Uma contribuição chave do presente trabalho é o que cunhamos de **recuperação oportunística**. Ao invés de reservar recursos sobressalentes, a capacidade perdida após um ataque de DoS bem sucedido é redistribuída sobre a banda disponível nos caminhos remanescentes dos enlaces virtuais afetados (denominados de *caminhos ativos*).

Quando um enlace físico torna-se inacessível, a capacidade dos enlaces virtuais sobrepostos é totalmente ou parcialmente comprometida. Caso determinado enlace virtual seja parcialmente afetado, ele ainda possuirá um conjunto de caminhos ativos no substrato. Dessa forma, é possível utilizar qualquer banda disponível nesses caminhos para tentar recuperar a capacidade do enlace virtual. A formulação da estratégia de recuperação é dada conforme segue:

Variável:

- $Q_{p,e^V} \in \mathbb{R}$ no intervalo $[0;1]$: indica a taxa de fluxo do enlace virtual $e^V \in \dot{E}^V$ utilizada no caminho p. Essa taxa é relativa apenas ao fluxo afetado, visto que o restante do fluxo do enlace virtual não é redistribuído.

Entrada:

- $\dot{E}^S \subset E^S$: conjunto de enlaces físicos disponíveis após a disrupção.
- P_{e^V}: conjunto de caminhos ativos para cada enlace virtual após a disrupção.
- $P_{p,e^S,e^V} \in \mathbb{B}$: indica que o enlace físico e^S está sendo utilizado pelo caminho ativo p do enlace virtual e^V.
- $\dot{E}^V \subset E^V$: indica o conjunto de enlaces virtuais afetados.

Objetivo: *maximizar a capacidade de recuperação após uma disrupção.* Dessa forma, a função

$$\max T = \sum_{\forall e^V \in E^V} \sum_{\forall p \in P_{e^V}} Q_{p,e^V} \tag{16}$$

visa realocar todo o tráfego afetado sobre o conjunto de caminhos ativos.

Restrições:

$$\sum_{\forall p \in P_{e^V}} Q_{p,e^V} \leq F_{e^V} \qquad \forall e^V \in \dot{E}^V \tag{17}$$

$$\sum_{\forall e^V \in E^V} \sum_{\forall p \in P_{e^V}} P_{p,e^S,e^V} \cdot Q_{p,e^V} \cdot b(e^V) \leq b(e^S) \qquad \forall e^S \in \dot{E}^S \tag{18}$$

A primeira restrição (17) impede que o modelo aloque mais banda do que o necessário (ou seja, somente aquela afetada pelo ataque). A segunda restrição (18) assegura que as capacidades de banda dos enlaces físicos serão respeitadas.

5. Avaliação

Os principais objetivos das estratégias descritas na seção anterior são (i) prevenir ataques de disrupção, alocando redes virtuais com múltiplos caminhos e de forma eficiente, e (ii) reagir perante ataques, realocando a capacidade comprometida em outros enlaces (quando possível). Esta seção avalia a abordagem proposta através da quantidade de enlaces virtuais afetados por ataques, com e sem proteção, e da quantidade de requisições aceitas na alocação determinada pela abordagem. Além disso, avalia-se o tempo necessário para realizar a alocação ótima de redes virtuais.

A formulação matemática descrita nas seções anteriores foi implementada no software CPLEX 12.3. Esse software utiliza variações do algoritmo Simplex e da técnica de *Branch-and-Bound* para encontrar a solução ótima dentro de determinada margem de erro (ou distância para a solução ótima). Todos os experimentos foram executados em um Intel Core i7 com 8 núcleos de 2,93 GHz e 8GB de memória ram. As condições dos experimentos e seus parâmetros são descritos a seguir.

5.1. Configuração dos Experimentos

Durante os experimentos o tempo foi discretizado, organizado em rodadas. Cada rodada pode receber uma ou mais requisições e executar o algoritmo de alocação para cada uma delas. Ademais, caso ocorra um evento de ataque, o algoritmo de recuperação também é executado. A rodada é encerrada somente após a conclusão de todas as etapas pertinentes à alocação e/ou recuperação.

As topologias de rede dos experimentos foram geradas sinteticamente utilizando o *software* BRITE com o modelo de redes Barabási-Albert (BA-2). As configurações das redes e da carga de trabalho são similares às definidas em trabalhos anteriores [Chowdhury *et al* 2009, Cheng *et al* 2012].

Substrato. A rede física é composta por 30 nós e 114 enlaces em uma grade de 60x60. As capacidades de processamento dos nós e de largura de banda dos enlaces são uniformemente distribuídas no intervalo [50,100][2]. O atraso dos enlaces é diretamente proporcional à distância entre os nós, conforme gerado pelo BRITE, normalizados para o maior valor (ou seja, em intervalos entre 0 e 1).

Redes virtuais. O número de nós de uma requisição de criação de rede virtual é gerado uniformemente entre 2 e 4. Conforme será discutido ao final da Subseção 5.2, o uso de valores assim baixos foi necessário para viabilizar a realização do conjunto de experimentos, visto que a etapa de alocação demanda muito tempo de processamento e memória. A capacidade dos nós e enlaces é uniformemente distribuída nos intervalos [5,20] e [50,100], respectivamente. Essas configurações de capacidade foram escolhidas de modo que um único atributo não seja o principal fator de impacto nos experimentos.

Cargas de trabalho. A carga de trabalho é composta por requisições de redes virtuais e por ataques de negação de serviço às mesmas via disrupção da rede física. A chegada de requisições é dada por um processo de Poisson com média de 7 para cada 100 rodadas. A duração de cada rede virtual segue uma distribuição geométrica com média de 1000 rodadas. Por sua vez, os ataques são modelados por um processo de Poisson com média de 1 para cada 100 rodadas. Conforme mencionado anteriormente (Seção 3.1), cada ataque é lançado contra o nó que possui a maior alocação de largura de banda em seus enlaces.

Os experimentos executaram durante 5000 rodadas. Dessa forma, durante cada experimento foram geradas, em média, 350 requisições e 50 ataques. Por fim, a margem de erro do CPLEX foi definida para 1%, de modo que soluções fossem encontradas em tempo factível.

5.2. Avaliação dos Resultados

As estratégias propostas ajudam a evitar o impacto de ataques. A abordagem proposta divide-se em duas estratégias: mapeamento em múltiplos caminhos (preventiva) e recuperação da capacidade perdida (reativa). A Fig. 1 ilustra o efeito da disrupção da rede física com e sem o uso das estratégias, em ambos os casos tendo os parâmetros w_Ξ, w_Φ e w_Δ fixados em $1/3$ (de modo que seu somatório seja 1). Os eixos x e y representam respectivamente o número de caminhos a ser usado entre dois nós e a perda média de largura de banda (proporcional) para cada enlace virtual causada pelo ataque. Intuitivamente, um número maior de caminhos deve tornar a rede mais resiliente, e quanto menor a barra, melhor.

No caso padrão, sem a abordagem proposta, há apenas 1 caminho por enlace virtual. Neste caso, quando ocorre um ataque, toda a banda alocada para os enlaces virtuais afetados é perdida e o mesmo fica completamente inoperante. Com o incremento no número de caminhos é possível utilizar as duas estratégias propostas para

[2]Em linha com trabalhos similares, mantém-se unidade no nível abstrato; a mesma poderia representar Mbps no caso de enlaces ou número de fatias de tempo no caso de nós.

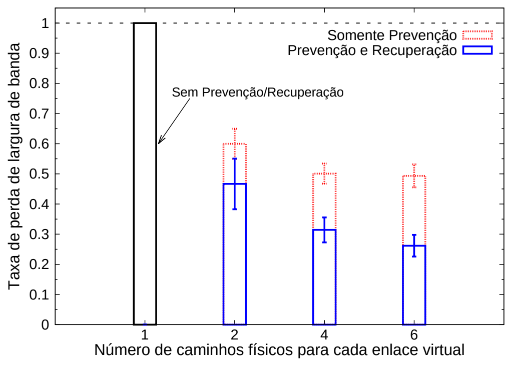


Figura 1. Perda e recuperação da largura de banda para cenários com 1, 2, 4 e 6 caminhos por enlace virtual. Neste experimento, $w_{\Xi} = w_{\Phi} = w_{\Delta} = 1/3$.

mitigar o efeito de ataques. O comportamento da estratégia preventiva é representado pelas barras vermelhas (contorno tracejado), enquanto o comportamento das duas estratégias em conjunto é representado pelas barras azuis (contorno contínuo). Considerando o cenário com 6 caminhos por enlace virtual, é possível perceber que a estratégia preventiva consegue proteger, em média, 50% da capacidade do enlace. Por sua vez, a utilização posterior da estratégia reativa para recuperar a capacidade comprometida reduz a perda para aproximadamente 25%. Como esperado, a abordagem proposta tem melhor resultado quando ambas as estratégias são utilizadas. Ademais, apesar da eficiência das estratégias aumentar com o número de caminhos utilizados, observa-se que seu comportamento não é linear. Tal é explicado pelas limitações na quantidade de caminhos disjuntos do substrato conforme evidenciado a seguir.

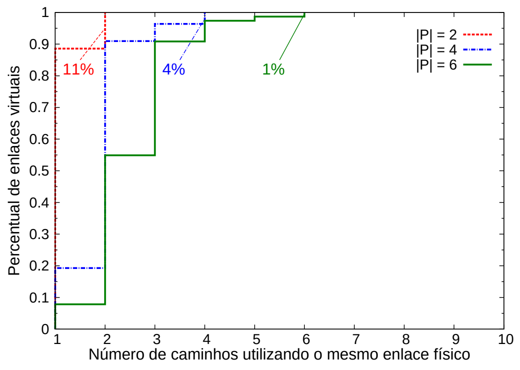
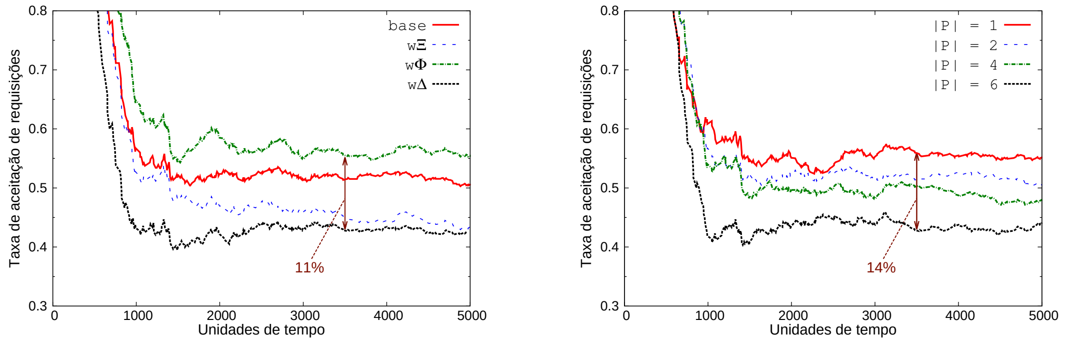


Figura 2. Nível de compartilhamento de enlaces físicos pelos caminhos alocados aos enlaces virtuais.

A alocação em múltiplos caminhos reduz a severidade de ataques a pontos críticos. Define-se um "ponto crítico" como um recurso, seja ele nó ou caminho, que por uma circunstância da rede física e/ou da alocação, acaba por concentrar toda a capacidade de um enlace virtual. Um ataque a um ponto crítico pode comprometer um ou mais enlaces virtuais por completo. A Função de Distribuição Acumulada (*Cumulative Distribution Function* – CDF) apresentada na Fig. 2 permite avaliar a quantidade de enlaces virtuais suscetíveis a esse tipo de ataque. O eixo x indica quantos caminhos de um mesmo enlace virtual utilizam determinado enlace físico (ou seja, quando for 1 não existe compartilhamento). Considerando dois caminhos por enlace virtual ($|P| = 2$), aproximadamente 11% desses enlaces possuem ambos os caminhos compartilhando o mesmo enlace físico. Ao incrementar o número de caminhos, a quantidade de enlaces virtuais suscetíveis ao ataque cai para 4% ($|P| = 4$) e então para 1% ($|P| = 6$). Por outro lado, é possível perceber que o número de caminhos disjuntos também decresce. A importância de tais caminhos está no fato de que se dois ou mais caminhos compartilham algum enlace físico, a eficiência da recuperação fica limitada à capacidade do mesmo. Quando os enlaces virtuais são mapeados em dois caminhos, 89% deles possuem caminhos disjuntos. Esse número decresce para 7% quando são usados 6 caminhos por enlace virtual. Tal evidência corrobora a afirmação anterior de que a quantidade de caminhos disjuntos no substrato pode afetar o desempenho da abordagem proposta.



(a) Número de caminhos por enlace virtual ($|P|$) fixado em 2

(b) Parâmetros $w_\Xi = w_\Phi = w_\Delta = 1/3$

Figura 3. Taxa de aceitação de acordo com a variação (a) dos parâmetros w_Ξ, w_Φ e w_Δ e (b) no número de caminhos por enlace virtual.

O custo da solução é diretamente proporcional ao nível de resiliência contra disrupção. A Fig. 3 ilustra as diferenças na taxa de aceitação de redes virtuais para diferentes parâmetros de entrada ao longo do tempo, com a taxa de aceitação sendo representada no eixo y. Em um primeiro momento ($0 \sim 900$), as requisições de rede virtual encontram um substrato ocioso com ampla disponibilidade de recursos. À medida que os recursos vão ficando escassos ($900 \sim 2000$), a taxa de aceitação diminui. Por fim, o substrato alcança estabilidade ($2000 \sim \infty$) quando a quantidade de recursos liberados pelas redes virtuais extintas é similar à quantidade demandada pelas novas requisições e 40 a 60% são aceitas. A Fig. 3(a) demonstra a diferença na taxa de aceitação causada pela escolha de prioridades distintas na alocação. O caso base (primeira curva da legenda e segunda de cima para baixo) foi utilizado para verificar a importância relativa de cada parâmetro. O aumento na *priorização por caminhos disjuntos* (parâmetro $w\Xi$, segunda curva na legenda e terceira de cima para baixo) provoca uma sobrecarga na alocação de recursos do substrato e diminui a taxa de aceitação. Tal se deve ao algoritmo de alocação, que opta por caminhos maiores para evitar a sobreposição de enlaces físicos. A escolha por *enlaces menos saturados* (parâmetro $w\Phi$) ou pela *minimização do atraso diferencial* (parâmetro $w\Delta$) acarretaram respectivamente na melhor e pior taxas de aceitação (com uma diferença de aproximadamente 11%). O aumento de $w\Phi$ ajuda a evitar a formação de gargalos, deixando o acesso mais livre aos nós da rede. Por outro lado, o aumento de $w\Delta$ prioriza a escolha de caminhos maiores com atrasos similares, ao invés de caminhos menores com maior diferença nos atrasos.

A Fig. 3(b) demonstra a eficiência do algoritmo de alocação ao longo do tempo com a variação no número de caminhos por enlace virtual. Apesar de possibilitar maior granularidade na alocação, o aumento no número de caminhos pode resultar em alocações piores. No pior caso, ao considerar 6 caminhos por enlace virtual, a taxa de aceitação decresce 14%. Isso ocorre porque ao considerar somente um único caminho, os efeitos colaterais causados pela priorização de caminhos disjuntos Ξ e pela minimização do atraso diferencial Δ são anulados. Dessa forma, o critério dominante na alocação passa a ser o balanceamento de carga Φ. No entanto, quanto maior for a quantidade de caminhos, maior será a importância relativa destes dois outros critérios.

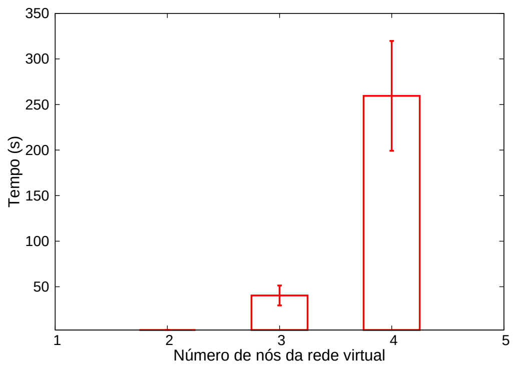


Figura 4. Tempo médio para alocar cada requisição de rede virtual.

A característica exponencial do problema de alocação justifica o estudo de heurísticas. Diversos trabalhos da literatura estudam heurísticas para lidar com o problema da alocação de redes virtuais [Chowdhury *et al* 2009, Alkmim *et al* 2011, Cheng *et al* 2012]. Sua similaridade com o *multiway separator problem* [Andersen 2002] é um forte indicio de que o problema é NP-Difícil. A Fig. 4 demonstra o tempo médio para a execução do algoritmo de alocação de redes sobre uma única requisição. Conforme pode ser observado, o problema de alocação limita a escalabilidade da solução, demandando em torno de 250s para resolver de forma ótima *uma única instância* de rede virtual com 4 nós na rede física descrita anteriormente. Dessa forma, a evolução natural deste trabalho é a definição de heurísticas para a abordagem proposta.

6. Conclusão

Apesar das potenciais vantagens de redes virtuais, como o isolamento, elas ainda estão sujeitas a ataques de negação de serviço lançados à rede física. Este trabalho apresentou uma abordagem inovadora para aumentar a proteção de redes virtuais contra ataques de DoS via rede física. Entre suas contribuições, destaca-se a que a abordagem de alocação proposta, diferentemente dos trabalhos anteriores, não precisa reservar recursos sobressalentes ou recalcular os caminhos utilizados pelos enlaces virtuais. A mesma é dividida em duas estratégias complementares. A primeira estratégia visa prevenir os ataques pelo uso de múltiplos caminhos *prioritariamente* disjuntos. A segunda visa recuperar a capacidade comprometida utilizando os caminhos ativos (ou seja, aqueles não afetados pelo ataque).

Conforme discutido na seção de avaliação, a solução proposta oferece um ganho na proteção e recuperação de capacidade, ao custo de uma menor taxa de aceitação. Ademais, esse ganho é incrementalmente menor à medida que a resiliência é priorizada sobre a alocação. Portanto, é possível adequar a solução aos requisitos do ambiente, priorizando a resiliência em detrimento da alocação ou adotando uma estratégia mais conservadora, mas ainda assim mitigar os ataques de disrupção.

Quanto aos trabalhos futuros, vislumbramos a definição de heurísticas para tornar a abordagem mais rápida, possibilitando sua aplicação em cenários maiores. Ademais, estamos investigando novas estratégias para o processo de recuperação, como a migração de nós, a qual possibilitará tornar a rede ainda mais robusta a ataques de negação de serviço.

Referências

Alkmim *et al* (2011). Optimal mapping of virtual networks. In *GLOBECOM, IEEE*, pages 1–6.

Andersen, D. G. (2002). Theoretical approaches to node assignment. Unpublished manuscript. `http://www.cs.cmu.edu/~dga/papers/andersen-assign.ps`.

Belbekkouche *et al* (2012). Resource discovery and allocation in network virtualization. *Comm. Surv. Tut., IEEE*, PP(99):1–15.

Chen *et al* (2010). Resilient virtual network service provision in network virtualization environments. In *ICPADS, IEEE*, pages 51–58.

Cheng *et al* (2012). Virtual network embedding through topology awareness and optimization. *Comput. Netw.*, 56(6):1797–1813.

Chowdhury, N. M. K. and Boutaba, R. (2010). A survey of network virtualization. *Comp. Netw.*, 54(5):862–876.

Chowdhury *et al* (2009). Virtual network embedding with coordinated node and link mapping. In *INFOCOM, IEEE*, pages 783–791.

Fortz, B. and Thorup, M. (2004). Increasing internet capacity using local search. *Comput. Optim. and Applic.*, 29:13–48.

Guo *et al* (2011). Shared backup network provision for virtual network embedding. In *ICC, IEEE*, pages 1–5.

He, J. and Rexford, J. (2008). Toward internet-wide multipath routing. *Network, IEEE*, 22(2):16–21.

Houidi *et al* (2010). Adaptive virtual network provisioning. In *2nd SIGCOMM VISA workshop, ACM*, pages 41–48.

Khan *et al* (2012). Network virtualization: a hypervisor for the internet? *Comm. Mag., IEEE*, 50(1):136–143.

Medhi, D. (2006). Network restoration. In Resende, M. G. C. and Pardalos, P. M., editors, *Handbook of Optimization in Telecomm.*, pages 801–836. Springer US.

Rahman *et al* (2010). Survivable virtual network embedding. In Crovella, M. *et al.*, editor, *NETWORKING 2010*, volume 6091 of *LNCS*, pages 40–52. Springer Berlin / Heidelberg.

Yeow *et al* (2010). Designing and embedding reliable virtual infrastructures. In *2nd SIGCOMM VISA workshop, ACM*, pages 33–40.

Yu *et al* (2011). Cost efficient design of survivable virtual infrastructure to recover from facility node failures. In *ICC, IEEE*, pages 1–6.

Zhang *et al* (2010). Reliable adaptive multipath provisioning with bandwidth and differential delay constraints. In *Proc. INFOCOM, IEEE*, pages 2178–2186.